Proceedings of the 10th International Conference on Cognitive Modeling

August 5-8, 2010

Drexel University
Philadelphia, PA

Edited by

Dario D. Salvucci & Glenn Gunzelmann

Table of Contents

Poster Abstracts

Doctoral Consortium Abstracts

Symposium Abstracts

Tutorial Abstracts

Introduction

The International Conference on Cognitive Modeling (ICCM) is the premier conference for research on computational models and computation-based theories of human behavior. ICCM is a forum for presenting, discussing, and evaluating the complete spectrum of cognitive modeling approaches, including connectionism, symbolic modeling, dynamical systems, Bayesian modeling, and cognitive architectures. ICCM includes basic and applied research, across a wide variety of domains, ranging from low-level perception and attention to higher-level problem-solving and learning. The 10th ICCM was held at Drexel University in Philadelphia, PA, on August 5-8, 2010.

All papers and abstracts in the ICCM 2010 proceedings may be cited as follows:

Doe, J., & Doe, J. (2010). This is the title of the paper. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 1-6). Philadelphia, PA: Drexel University.

Sponsors

Air Force Office of Scientific Research

Air Force Research Laboratory, Human Effectiveness Directorate

Aptima, Inc.

Cognitive Science Society

National Science Foundation

Office of Naval Research

Drexel University

Committees

Organizing Committee

Conference Chairs:	Dario Salvucci & Glenn Gunzelmann
Tutorials & Workshops:	Frank Ritter
Doctoral Consortium:	Robert St. Amant
Local Administration:	Julie Fisher & Tuyet Sithiphavong

Program Committee

Erik Altmann
Thomas Barkowski
Martin Baumann
Roman Belavkin
Thierry Bellet
Duncan Brumby
Mike Byrne
Nick Cassimatis
Balakrishnan Chandrasekaran
Richard Cooper
Garrison Cottrell
Fabio Del Missier
Wai-Tat Fu
Danilo Fum
Kevin Gluck
Fernand Gobet
Tim Halverson
Andrew Howes
Christian Janssen
Gary Jones
Mark Keane
David Kieras
Boicho Kokinov
John Laird
Peter Lane
Christian Lebiere
Richard Lewis
Yili Liu
Michael Matessa
Alain Mille
Claus Möbus
Shane Mueller
Christopher Myers
Josef Nerb
Hansjoerg Neth
David Noelle
David Peebles
Thad Polk
Marco Ragni
Frank Ritter
Ute Schmidt
Mike Schoelles
Lael Schooler
Christian Schunn
Barry Silverman
Patrick Simen
Robert St. Amant
Terry Stewart
Andrea Stocco
Ron Sun
Niels Taatgen
Greg Trafton
Hedderik van Rijn
Boris Velichkovsky
Robert West
Sharon Wood
Richard M. Young

Tutorials Committee

Erik Altmann
Mark Cohen
Jim Davies
Fabio Del Missier
Olivier Georgeon
Randolph M. Jones

Awards Committee

Erik Altmann
Wai-Tat Fu
Wayne Gray
Andrew Howes
Tiffany Jastrzembski
Shane Mueller
Terry Stewart
Leendert van Maanen
Richard M. Young

Awards

The following awards honor the best paper and poster contributions in select categories as chosen by a committee of distinguished researchers. Congratulations to our winners and honorees!

Siegel-Wolf Award for Best Applied Paper

Sponsored by Aptima, Inc.

This award, given for the best applied research paper, is named in recognition of Art Siegel and Jay Wolf, who worked on human performance models for more than 20 years at Applied Psychological Services in Wayne, PA. The winners are:

Task-Constrained Interleaving of Perceptual and Motor Processes in a Time-Critical Dual Task as Revealed Through Eye Tracking
Anthony J. Hornof & Yunfeng Zhang

The Evolution of a Goal-Directed Exploration Model: Effects of Information Scent and Go-back Utility on Successful Exploration
Leonghwee Teo & Bonnie E. John

Honorable mention goes to the following papers:

Modeling the Effects of Work Shift on Learning in a Mental Orientation and Rotation Task
Tim Halverson, Glenn Gunzelmann, L. Richard Moore Jr., & Hans Van Dongen

Exploration of Costs and Benefits of Predictive Human Performance Modeling for Design
Bonnie E. John & Tiffany S. Jastrzembski

Allen Newell Award for Best Student Paper

Sponsored by the Office of Naval Research

This award, given for the best full paper with a student as first author, is named in recognition of Allen Newell, one of the founders of the field of cognitive modeling. The winner is:

A Cognitively Bounded Rational Analysis Model of Dual-Task Performance Trade-Offs
Christian P. Janssen, Duncan P. Brumby, John Dowell, & Nick Chater

Honorable mention goes to the following papers:

A New Approach to Exploring Language Emergence as Boundedly Optimal Control in the Face of Environmental and Cognitive Constraints
Jeshua Bratman, Michael Shvartsman, Richard L. Lewis, & Satinder Singh

Rewards and Punishments in Iterated Decision Making: An Explanation for the Frequency of the Contingent Event Effect
Antonio Napoli & Danilo Fum

Neural Correlates of Temporal Credit Assignment
Matthew M. Walsh & John R. Anderson

Best Student Poster

Sponsored by the Cognitive Science Society

This award is given for the best poster presentation for a paper or abstract, submitted to the main program, with a student as first author and presenter. Committee members will visit student posters during the poster sessions and the award winner(s) will be announced on Sunday morning at the start of the 9am session.

HOW TO INVESTIGATE THE *LIVING COGNITION*: AN APPLICATION TO DYNAMIC SIMULATION OF MENTAL ACTIVITIES WHILE DRIVING

Thierry Bellet (thierry.bellet@inrets.fr), Pierre Mayenobe (pierre.mayenobe@inrets.fr), Jean-Charles Bornard (jean-charles.bornard@inrets.fr)
INRETS (LESCOT) - French National Institute on Transport and Safety Research,
25 Avenue François Mitterrand, 69675 Bron cedex, France

Abstract

This paper is dedicated to the "living cognition" issues, which concern the ability of a cognitive model to simulate humans' mental activities when dynamically interacting with the external environment. After having introduced the theoretical foundations of this approach, an integrative *COgnitive Simulation MOdel of the DRIVEr* is presented (i.e. COSMODRIVE). The central process that supports the *living cognition* in this model is the *deployment* of a *cognitive schema*, corresponding to the driver's mental representation of the driving situation as instantiated in the Working Memory. This dynamic visual-spatial mental model, defined as the driver's *situational awareness*, is used by the driver for perceptive exploration of the road scene, decision-making, anticipation and action planning, in order to interact with the road environment. This dynamic process of regulation is based on both *implicit* and *explicit* mental simulations and is illustrated through an example in the last section of the paper.

Keywords: Cognitive simulation, car driving, visual-spatial mental representation, dynamic cognition, implicit and explicit situation awareness.

1.Theoretical foundation of the living cognition

Although a familiar task of everyday life, car driving is however a complex activity that involves every levels of human cognition. Indeed, driving a car requires (i) to select relevant information from the environment, (ii) to understand the current situation and to anticipate its progression in the more or less long term, (iii) to take decisions in order to dynamically interact - via the vehicle - with the road environment and the other road users, (iv) and to manage owns resources (physical, perceptive and cognitive) in order to satisfy the time constraints of the task, inherent to the dynamic nature of the driving situation. The selective dimension of information collection is especially important as drivers cannot take in and process all the information available in the road environment. As we shall argue in this paper, this information is not selected haphazardly. It depends on the aims the drivers pursue, their short-term intentions (i.e. tactical goals, such as turn left at a crossroads) and long-term objectives (i.e. strategic goals, such as reaching their final destination within a given time), the knowledge they possess and the attentional resources allocated to the driving task. Information selection is the result of a complex process whose keystone is the driver's mental representation of the driving situation. Indeed, from their interaction with the road environment, drivers build mental models of the events and objects that surround them. These mental representations are dynamically formulated in working memory through a matching process between (i) pre-existing *operative* knowledge (Ochanine, 1977) and (ii) perceived information extracted in the external environment. They are formulated *by* and *for* the action, and they provide *interiorized models of the task* (Leplat, 2005). When driving, these representations provide 3-Dimensional (i.e. visual-spatial) models of the environment, liable to be mentally manipulated by the driver, in order to support anticipation through cognitive simulations, and thus providing expectations on future situational states. Drivers continually update these mental models as and when they carry out their activity. This dynamic process, based on both *implicit* and *explicit* mental simulations (Bellet et al., 2009), is the central focus of the "*living cognition"* (Bellet, 2010) as investigated in this paper. At a theoretical level, the living cognition is jointly based on three scientific traditions: (i) the *cybernetics* and the *human information processing* theories, (ii) the Russian *theory of activity*, and (iii) the *ecological approach of human perception*.

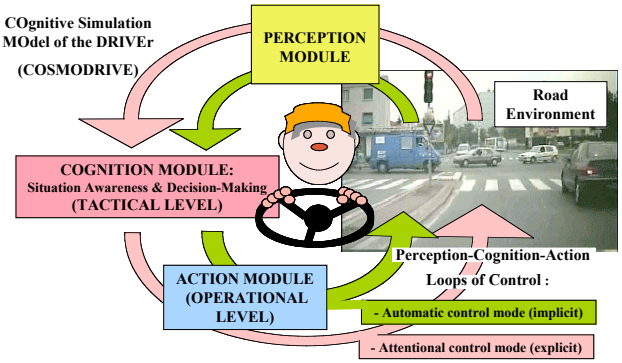


Figure 1: the car driving activity as a dual regulation loop

According to Wiener's *cybernetics theory* (1948), human can be defined as a self-adaptive system who interacts with the external environment through a feedback regulation mechanism. Humans' mental activities are then described as a black box owning information processing mechanisms, able to generate *outputs* from perceptual *inputs*, in order to adapt itself to the situation. As and when this cycle repeats itself recursively, the human cognitive system perceptually assesses the effects of its action on the environment, and then determines which new action is needed to achieve the expected state of the surroundings. This iterative process start again until this state-goal is obtained. Although cybernetics has finally introduced an epistemological break with the behaviorist approach in Psychology, the initial model proposed by Wiener was fully compatible with the Skinner's "S-R" approach, until the Pandora's black box was opened. However, with the development of the *human*

information processing theory, the internal mechanisms implemented into the black box, like mental representations elaboration, reasoning, or decision-making, became the new central topics of the cognitive sciences. Nevertheless, according to the experimental method used in laboratory for investigating cognition in well-controlled conditions, the Cybernetics "loop logic" has been progressively lost for two main reasons. First, the experimental paradigm applied in cognitive sciences requires to artificially break down human cognition into several functions to be individually investigated. Moreover, and maybe more critical from the *living cognition* point of view, in-lab investigation of human cognition are based on repetitive measures collected for similar artificial tasks, in similar conditions. Therefore, the story must re-start after each new stimulus, as if it was a totally "new story", in order to allow the scientists to rigorously control the experiment. After each S-R sequence, the task is thus completed, without any expected feedback effect. Therefore, by using the experimental method, cognitive sciences ended up losing the notion of "cycle", however so important in the cybernetics *feedback* process supporting the dynamic of the living cognition, in favor of a sequential string of processes, from perception to action.

Like Cybernetics, the Russian *Theory of Activity* considers human operators through their dynamic interactions with the external environment. But in this approach, *Activity* is the starting point and the core topic of the scientific study of human cognition, because it is argued that activity directly structures the operator's cognitive functions. The fundamental postulate of the Theory of Activity is well summarized by Smirnov (1966): *human becomes aware of the surrounding world, by acting on it, and by transforming it*. From this point of view, human is not a passive cognitive system whose undergoes the stimulus given by the external environment. S/he is an active observer, with inner intentions, able to voluntary act on the world and to modify the situation by their activity, in accordance with their own needs. Indeed, behind activity *there is always a need, which directs and regulates concrete activity of the subject in the objective environment* (Leontiev, 1977; p. 88). Such a consideration, so essential in our everyday life as psychological subjects with needs, intents and will, has been nevertheless progressively forgotten by the modern cognitive sciences, when based on the experimental paradigm. Through laboratory experiments, inner needs and spontaneous motives disappear, as well as the dynamic "life cycle" of the natural living cognition.

The same criticism against the destructive effect of experimental method when applied to cognition has been formulated by Neisser (1976), through his *ecological approach of human perception*. Neisser's work was initially based on the *direct perception* theory of Gibson (1979), who postulates that some *affordances*, corresponding to properties of the objects, are directly perceived by the organism. By contrast with the Gibson "un-cognitive" theory of perception, Neisser admits the existence of mental functions, even if he criticizes the sequential vision of the cognition dominated the human information processing theory. In a synthetic way, Neisser considers perception as a skilled and iterative process. Like the Russian theorists of the activity, he argues that human are not passive receivers of perceptual inputs, but that they are *active* in the world, in accordance with their own motives, their abilities, and their expectations. His approach describes perception as a dynamic cycle focused on the relationships between pre-existing knowledge and the human information-gathering activity. According with this *perceptive cycle*, the perceiver actively explores the surroundings, and then constructs a dynamic understanding of the current environment. The mental structure that supports such processes of perception is described as an *active schema* of the environment, which is continually modified by the new perceptual information, and which also contains anticipatory expectations. This mental schema includes a *cognitive map* of the world, and therefore directs perceptual explorations of the environment, or prepares the mind for perception of anticipated events. It can be consequently considered as a kind of *control structure* of the perceptive processes.

2. An integrative model of the car driver

In this section, we would like to present a comprehensive model of the human driver, so-called COSMODRIVE (for COgnitive Simulation MOdel of the DRIVEr, Bellet et al., 1999, 2010), that combines in an integrative way the different theoretical approaches presented above. Several driver models have been developed during the last decades, even if the most of them are focused human's *performance* more than on *cognitive simulation* (for a discussion on this issue, see Bellet et al., 2007). One of the most advanced one is surely the driver model developed by Salvucci (2006), that is based on the ACT-R cognitive architecture (Anderson and al., 2004). Like COSMODRIVE, this model provides an integrative approach of the driver's cognition, by considering 3 components of (i) *control*, (ii) *monitoring*, (iii) and *decision making*. Cognitive abilities at the *monitoring* level are conceptually close to our approach of *mental representation* simulation, even if they are different from the computational point of view (ACT-R *chuncks* in *declarative memory* versus visual-spatial [3D] and dynamic mental models in COSMODRIVE). Nevertheless, the aim of this paper is not to theoretically discuss on driver models, but only to provide an illustrative example of the *living cognition*, applied to a very familiar task. The figure 2 provides a synthetic overview of the cognitive architecture of COSMODRIVE. The heart of the model are the drivers' mental representations of the driving environment, corresponding to the driver's *Situation Awareness* according to Endsley (1995) definition of this concept: *the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*. These mental models are built in working memory. At the *tactical level* (Michon, 1985), they provide an ego-centered and a goal-oriented understanding of the traffic situation, including

anticipations of the future changes of the current driving situation, liable to be mentally investigated by the driver at an explicit level. At the operational level, which generally corresponds to the driver's *implicit awareness* of the situation, driving activity is implemented through operative know-how for vehicle lateral and longitudinal controls (Bellet et al., 2009). This dichotomy between implicit and explicit cognition is well established in scientific literature, for example, with the distinction proposed by Schneider and Schiffrin (1977) between *controlled processes*, which require cognitive resources and which can only be performed sequentially, and *automatic processes*, which can be performed in parallel without any attentional effort. In the same way, Rasmussen (1986) distinguishes different levels of activity control according to whether the behaviors implemented rely on (i) integrated sensorial-motor reflexes (*Skill-based behaviors*), (ii) decision rules for managing familiar situations (*Rule-based behaviors*), or (iii) generic knowledge activated in new situations for which the driver doesn't have any experience (*Knowledge-based behaviors*).

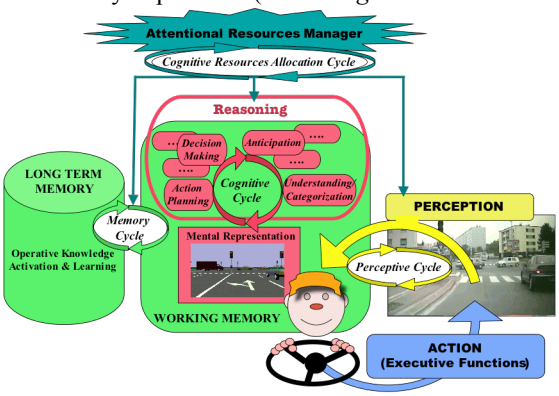


Figure 2: Cognitive architecture of COSMODRIVE

Four dynamic cycles regulate the internal functioning of the model. The *perceptive cycle* supports the human perception functions, allowing the driver to actively explore the road environment, according to their current needs and objectives (top down *perceptive exploration* process) and to integrate new information into their mental models (bottom up *cognitive integration* process). The *memory cycle* plays a central role for pre-existing knowledge activation (based on *categorization* and *matching* processes permitting to fit knowledge with the reality, Bellet et al., 2007) as well as in terms of new knowledge acquisition. The *cognitive cycle* corresponds to a set of *cognitive agents* (like mental representation elaboration, understanding, anticipation, decision-making, or action planning) which collectively handled the internal mental representations, in order to take appropriate decision and then, to act into the current environment. Lastly, the *cognitive resources allocation cycle* is in charge to dynamically regulate and control the life cycle of the driver's cognitive system, in accordance with the attentional resources that are currently available.

The central structure supporting to the living cognition in this cognitive architecture is the *working memory*. From this point of view, this architecture is directly inspired by the ACT-R theory (Anderson et al., 2006). However, the working memory of COSMODRIVE merges both *procedural* and *declarative* memories, and comes more from the *operational memory* concept of Zintchenko than from the Baddeley's *working memory* model (1986). For Zinchenko (1966), the operational memory is a structure whose main function is to *serve the real needs of the activity*. Thus, it is a transitory rather than permanent memory. However, it should be distinguished from a short-term buffer limited in storage capacities, in so far as the information it contains remains available for as long as the task is performing (for several hours in some cases).

Through COMSODRIVE approach, car driving is modeling as a dynamic process of interaction between the driver and the environment through a dual iterative regulation loop, supporting the living cognition. In accordance with the *Cybernetics theory*, human activity is defined here as an continuous loop of regulation between (i) *inputs*, coming from the road environment, and (ii) *outputs*, corresponding to the driver's behaviors implemented into the real world via the car, which generate (iii) *feedbacks,* in the form of a new inputs, requiring new adaptation from the driver. From this general point of view, the first iteration of the Perception-Decision-Action regulation loop corresponds to the moment when the driver starts up the engine, and the last iteration comes when the driver reaches the final trip destination, and stops the car. In accordance with the *Human information processing theory*, human is not described here as a closed black box, but as a set of perceptive, cognitive and behavioral functions allowing the driver to dynamically regulate their interactions with the surrounding environment. In terms of cognitive activities, mental representation of the driving situation plays a key-role in the cognitive system functioning. This mental model, based on perceptive information extracted into the road environment, corresponds to the driver's awareness of the driving situation, and therefore determines directly all their decision-making concerning the relevant adaptive behaviors to be carried out in the current driving context. In accordance with the Russian *theory of activity*, this mental representation is based on operative knowledge practically learnt "*in situation*". Moreover, the driving task is performed by using an *artifact* (i.e. the vehicle), and the driving situation is directly *transformed* by the human operator's activity (e.g. car position on the road depending of the driver's action on the vehicle controls), as well as the situation *modifies* the driver's cognitive states (in terms of mental representation updating, for example, or new operative knowledge learning). Lastly, in accordance with the *ecological theory of Neisser* (1976), driver's perception in figure 2 is based on a dynamic *perceptive cycle* when (i) an active schema directs gathering-information activity (i.e. top down processes) and (ii) focus driver's attention on information currently available in the environment. Then (iii), this active schema provides a mental model that is continuously updated by dynamic integrating the new pieces of information collected into the road scene.

3. Computational and dynamic simulation of the driver's mental activities while driving

By considering this theoretical background, the COSMODRIVE model is composed of three main functional modules (i.e. the *Perception*, the *Cognition*, and the *Action* modules) in order to drive a virtual *Car* into a virtual *Environment* through two synchronized "Perception-Cognition-Action" regulation loops (Bellet et al., 2010): an attentional control mode (mainly focused on Rasmussen's rule-based behaviors, and simulated through *Driving Schemas*, and an automatic control loop (corresponding to the skill-based behaviors simulated through the *Envelope Zones* concept and the *Pure-Pursuit Point* method).

3.1 Modeling the explicit cognition: the *Driving Schemas*

Based on both the Piaget's concept of *operative scheme* and the Minsky (1975) *frames theory*, *driving schema* is a computational formalism defined in order to implement *operative driving knowledge* a the tactical level of COSMODRIVE (Bellet et al., 1999). They correspond to prototypical empirical situations, actions and events, learnt by the driver from practical experience.

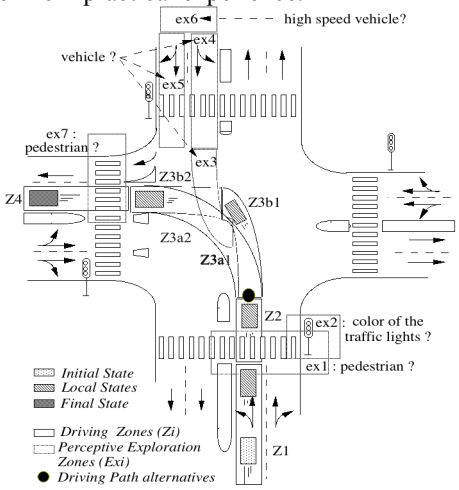


Figure 3: The *Driving Schemas* formalism

From a formal point of view (Figure 3), a *Driving Schema* is composed of (i) a functional model of road *Infrastructure*, (ii) a *Tactical Goal* (e.g. turn left), (iii) a sequence of *States* and (iv) a set of *Zones*. Two types of *Zone* are distinguished: *Driving Zones* (Z_i), corresponding to the driving path of the vehicle as it progresses through the crossroads, and the *Perceptive Exploration Zones* (ex_i), in which the driver seeks information (e.g. potential events liable to occur). Each driving zone is linked to *Actions* to be implemented (e.g. braking or accelerating, in view to reach a given state at the end of the zone), the *Conditions* of performing these actions, and the perceptive exploration zones that permit checking these conditions (e.g. color of traffic lights, presence of other road users). A *State* is defined by a vehicle position and speed. The different sequences of the driving zones make up the *Driving Paths* that progress from the initial to the final state (achievement of the tactical goal).

Once activated in working memory and instantiated with the road scene, the active driving schema becomes the *tactical mental representation* of the driver, which will be continually updated as and when s/he progresses into the current environment. Tactical representation corresponds to the driver's explicit awareness of the driving situation and provides a mental model of the road functionally structured, according to the tactical goal pursued by the driver in this particular context (e.g. turn on the left).

3.2 Modeling the implicit cognition: the *Envelope-Zones* and *Pure Pursuit Point* regulation strategies

At the operational level (corresponding to the automatic control loop presented in fig. 1), COSMODRIVE regulation strategy is based on two implicit regulation mechanisms: the *envelope zones* and the *pure pursuit point*. From a theoretical point of view (Bellet et al., 2007), the concept of envelope zones recalls two classical theories in psychology: the notion of *body image* proposed by Schilder (1950), and the theory of *proxemics* defined by Hall (1966), relating to the distance keeping in social interactions with other humans. Regarding car-driving activity, envelope zones also refer to the notion of safety margins. At this last level, COSMODRIVE model approach (Fig.4) is more particularly based on Kontaratos' work (1974), and distinguishes a *safety zone*, a *threat zone*, and a *danger zone* in which no other road user should enter (if this occurs, the driver automatically activates an emergency reaction).

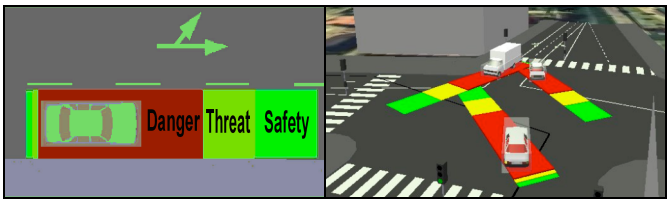


Figure 4: COSMODRIVE "Envelope-Zones" model

The envelope zones correspond to the portion of the path of driving schema to be occupied by the vehicle in the near future. Moreover, as an "hidden dimension" of the social cognition, as suggested by Hall's theory (1966), these proxemics zones are also mentally projected to other road users, and are then used to dynamically interact with them, as well as to anticipate and manage collision risks. This "virtual skin" is permanently active while driving, as an implicit awareness of our expected allocated space for moving. As with the Schilder's body schema, it belongs to a highly integrated cognitive level (i.e. implicit regulation loop), but at the same time favors the emergence of critical events in the driver's explicit awareness. Therefore, the envelope zones play a central role in the regulation of social as well as physical interactions with other road users under normal driving conditions (e .g. inter-vehicle distance keeping), and in the risk assessment of path conflicts and their management if a critical situation occurs (commitment of emergency reactions).

The second hidden dimension of the implicit cognition implemented at the operational level of COSMODRIVE is

the *Pure Pursuit Point* method. This method was initially introduced for modeling in a simplified way the lateral and the longitudinal controls of an automatic car along a trajectory (Amidi, 1990), and has been adapted by Sukthankar (1997), and then Mayenobe (2004), for driver's situational awareness modeling. Mathematically, the pure-pursuit point is defined as the intersection of the desired vehicle path and a circle of radius centered at the vehicle's rear axle midpoint (assuming front wheel steer). Intuitively, this point describes the steering curvature that would bring the vehicle to the desired lateral offset after traveling a distance of approximately l. Thus the position of the pure-pursuit point maps directly onto a recommended steering curvature: $k = -2x/l$, where k is the curvature (reciprocal of steering radius), x is the relative lateral offset to the pure-pursuit point in vehicle coordinates, and l is a parameter known as the look-ahead distance. According to this definition, the operational control of the car by COSMODRIVE can be seen as process of permanently keeping the Pursuit Point in the driving path, to a given speed assigned with each segment of the current tactical schema, as instantiated in working memory.

4. The emerging living cognition

By using the functional architecture and the cognitive agents of COSMODRIVE described in figure 2, (ii) the *driving schemas* as operative knowledge activated and then dynamically updated in the form of a functional mental representation matched with the road scene, and (iii) the operational skills corresponding to the pure-pursuit point and the envelopes zones regulation process, it becomes thus possible to dynamically simulate of the driver's "living cognition". The central process that supports the living cognition is the *deployment* of the active driving schema, as instantiated in Working Memory through the current mental representation. This deployment consists in moving the car along a *driving path* (cf. fig. 3), by successively traveling through the different driving zones of the schema, from the initial state (i.e. Z1) until reaching the tactical goal (i.e. Z4). This deployment process may occurs at two levels: (i) at the representational level (*explicit* and *implicit* mental simulations of the future activity to be carried out), when the drivers anticipate and project themselves mentally in the future, (ii) and through the activity itself, during the effective implementation of the schema while driving the car. This twofold deployment is not performed by a specific process in COSMODRIVE. It is an *emergent* collective product, resulting from the combined effect of several cognitive processes (like *anticipation* or *decision-making*), and merged with the computations based on the *envelope zones* and the *pursuit point* regulation laws. As a result, the deployment process generates a particular *instance* of the active schema execution, composed of a temporal sequence of mental representations, causally interlinked, and corresponding to the driving situation as it is progressively understood and anticipated, then experienced, and lastly *acted* by the driver, along the driving path progression.

The figure 5 provides an example of COSMODRIVE simulation results, permitting to visualize the mental representation evolution of a novice driver (who has the intention to turn on the left), while approaching of an urban crossroads with traffic lights. In a first time (i.e. first left view, corresponding to the driver's mental representation at a distance of 30 meters of the traffic lights), the driver's situation awareness is centered on the near traffic and on the traffic lights color, that directly determine the short-term activity to be implemented. Then, as s/he progresses towards the crossroads, the driver's attention is gradually focused on the ahead area, and the traffic flow occurring in the intersection center is progressively integrated into the driver's mental representation (i.e. second left view, at a distance of 10 meters of the traffic lights).

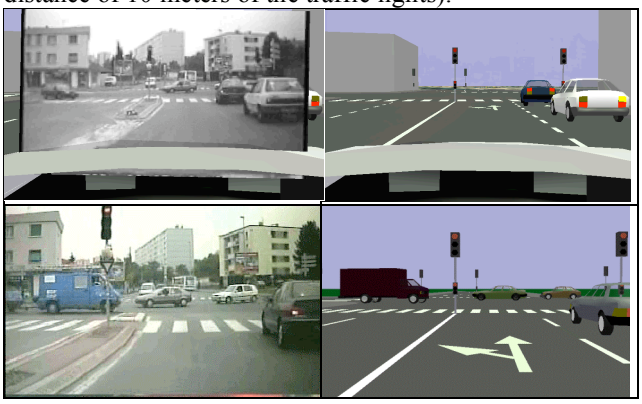


Figure 5: virtual simulation of a driver's mental models

The advantage of the driving schema formalism as defined in COSMODRIVE is to combine declarative and procedural knowledge in the unified computational structure. When associated with the operational regulation processes linked with the *envelope zones* and the *pursuit point* strategies, it is then possible to use such driving schemas as a structure of control for both monitoring the operative activity, as well as for supervising the mental derivation of the "schema deployment", as this process is implemented by the human cognitive system in order to anticipate future situational status, or to mentally explore the potential effects of an action before applied it. In accordance with the activity theories, these cognitive structures guarantee a continuum between the different levels of awareness (implicit *versus* explicit) and the activity control (tactical *versus* operational), thereby taking full account of the embedding of operative know-how (i.e. the level of implementation) in the explicit and decisional regulation loop of the activity.

5. Conclusion: "*in silico veritas*"

By considering the challenge of the *living cognition* study, it is needed to apprehend the dynamic functioning of the human cognitive system in interaction with the environment where s/he is currently immersed. Thus, computational models able to virtually simulate the human mental activities on computer are required. One of the key issues of the living cognition is mental representations simulation, that are

dynamically elaborated and continually updated in the working memory of the human operator before (i.e. action planning) and during the activity, when practically carried out. Indeed, mental representations and operative activity are intimately connected. In the same way as the human activity fuels itself directly with mental representations, the operator's mental representations are also fuelled "by" the activity, and "for" the activity, according to a double deployment process: cognitive and representational, on the one hand, and sensorial-motor and executive, on the other.

The key mental structure supporting both drivers' mental representations and their activity are *driving schemas*. From a metaphorical standpoint, such schemas can be compared to a strand of DNA. They "genetically" contain all the potential behavioral alternatives that allow the driver to act within a generic class of situations. Nonetheless, only a tiny part of these "genotypic potentialities" will finally express themselves in the current situation – with respect to the constraints and specific characteristics of reality – during the cognitive (i.e. mental deployment), and then executive implementation of this schema (i.e. effective activity carried out to drive the car). And it is only through this dynamic process of deployment of operative mental representations, involving a collective effort of several cognitive processes, that certain of intrinsic properties of the *living cognition* will emerge. From this point of view, the scientific investigation of the living cognition cannot forego the use of computer simulation of the human mental activities, without taking the risk of being largely incomplete.

Acknowledgments

The research is currently supported by the European Commission Seventh Framework Program (FP7/2007-2013), in the frame of ISi-PADAS Project.

References

Amidi O. (1990). *Integrated mobile robot control*, (Technical Report CMU-RI-TR-90-17). Pittsburgh, PA: Carnegie Mellon University, the Robotics Institute.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y . (2004). An integrated theory of the mind. *Psychological Review*, 111, (4), 1036-1060.

Baddeley A.D. (1986). *Working Memory*. Clarendon Press.

Bellet, T. (2010). Analysis, modeling and simulation of human operator's mental activities. In G. A. Boy (Ed), *Handbook of Human-Machine Interaction*, Ashgate.

Bellet, T., Bailly-Asuni, B., Mayenobe, P., & Banet, A. (2009). A theoretical and methodological framework for studying and modelling drivers' mental representations. *Safety Science*, 47,1205–1221.

Bellet, T., Bailly, B., Mayenobe, P., & Georgeon, O. (2007). Cognitive modelling and computational simulation of drivers mental activities. In P. Cacciabue (Ed.), *Modelling Driver Behaviour in Automotive Environment: Critical Issues in Driver Interactions with Intelligent Transport Systems*, 315-343, Springer Verlag.

Bellet, T., Mayenobe, P., Bornard, J.C., Gruyer, D., & Mathern, B. (2010). COSMO-SIVIC: a first step towards a virtual platform for Human Centred Design of driving assistances. *Proceedings of the 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Valenciennes, France

Bellet, T., Tattegrain-Veste, H. (1999). A framework for Representing Driving Knowledge. *International Journal of Cognitive Ergonomics,* 3 (1), 37-49.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. Human Factors 37, 32-64.

Gibson, J.J. (1979). *The Ecological Approach to Visual Perception*. Boston, Mass.: Houghton Mifflin.

Hall, E.T. (1966). *The hidden dimension*. New York: Doubleday.

Kontaratos, N.A. (1974). A system analysis of the problem of road casualties in the United States. *Accident Analysis and Prevention*, 6, 223-241.

Leontiev, A. (1977). *Activity and Consciousness*, on-line available at: http://www.marxists.org/archive/404.htm

Leplat, J. (1985). Les représentations fonctionnelles dans le travail. *Psychologie Française*, 30 (3/4), 269-275.

Mayenobe, P. (2004). *Perception de l'environnement pour une gestion contextualisée de la coopération Homme-Machine*. PhD Thesis, University of Clermont-Ferrand.

Michon, J.A. (1985). A critical view of driver behavior models : what do we know, what should we do ?. In: Evans, L., Schwing, R.C. (Eds), *Human behavior and traffic safety*. New York: Plenum Press.

Minsky, M. (1975). A Framework for Representing Knowledge. In P.H. Winston (Ed.), *The Psychology of Computer Vision*. New York: Mc Graw-Hill.

Ochanine V.A. (1977). *Concept of operative image in engineering and general psychology*. In B.F. Lomov, V.F., Rubakhin & V.F. Venda (Eds), *Engineering Psychology*. Moscow: Science Publisher.

Rasmussen J. (1986). *Information processing and human-machine interaction: an approach to cognitive engineering*. Amsterdam: North Holland.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, 48, 362-380.

Schilder, P. (1950). *The image and appearance of the human body*. New York: International Universities Press.

Schneider W., & Shiffrin R.M. (1977). Controlled and automatic human information processing I: Detection, search and attention. *Psychological Review*, 84, 1-88.

Smirnov A. (1966). La mémoire et l'activité. In A. Leontiev, A. Luria & A Smirnov (Eds), *Recherches Psychologiques en URSS*. Moscow: Editions du Progrès.

Sukthankar R. (1997). *Situation Awareness for Tactical Driving*, Phd thesis, Carnegie Mellon University.

Wiener N. (1948). *Cybernetics or control and communication in the animal and the machine*. Cambridge: MIT Press.

Zintchenko P. (1966). Quelques problèmes de psychologie de la mémoire. In A. Leontiev, A. Luria & A Smirnov (Eds), *Recherches Psychologiques en URSS*. Moscow: Editions du Progrès.

A New Approach to Exploring Language Emergence as Boundedly Optimal Control in the Face of Environmental and Cognitive Constraints

Jeshua Bratman[1] (jeshua@umich.edu)
Division of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109

Michael Shvartsman[1] (mshvarts@umich.edu)
Richard L. Lewis (rickl@umich.edu)
Department of Psychology, University of Michigan, Ann Arbor, MI 48109

Satinder Singh (baveja@umich.edu)
Division of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109

Abstract

Computational experiments have been used extensively to study language emergence by simulating the evolution of language over generations of interacting agents. Much of this work has focused on understanding the mechanisms of how language might have evolved. We propose a complementary approach helpful in understanding why specific properties of language might have emerged as an adaptive response to joint pressures from the environment and constraints on an agent's cognitive architecture. The approach suggests that linguistic systems can be described as boundedly optimal policies in multi-agent dynamic control problems defined by specific environments, agent computational structures, and task-oriented (vs. communication oriented) rewards. We illustrate the approach with a set of computational experiments.

Keywords: language emergence, bounded optimality, cognitive architecture, reinforcement learning, adaptive control

Introduction

The goal of this paper is to begin exploring a new approach to understanding the emergence of language. The primary scientific aim is understanding how pressures from the *environment* and constraints on the agent's *cognitive architecture* jointly lead to the emergence of specific properties of linguistic communication as optimal policies for obtaining well-defined long-term task- or environment-related reward.

Taking this perspective allows us to abstract away from the question of *how* language evolved and systematically explore constraints explaining *why* language appeared in the form that it has. We hypothesize that specific language-like properties (for instance, compositionality and systematic reliance on surface cues such as order) can in part be explained as bounded optimal solutions to control problems faced by computationally limited agents in environments exerting specific pressures. We propose investigating language through such environments in which we can formulate control problems for two or more bounded agents. If the optimal policies for these agents exhibit certain linguistic properties, then we can begin to define a mapping from the original pressures and agent constraints to the properties exhibited.

Finding solutions to these control problems computationally can be accomplished through various means such as reinforcement learning, game-theoretic analysis, or evolutionary algorithms. Thus, the approach allows us to step away from assumptions about specific mechanisms of learning or evolution, and focus on the joint relationship of agent structure and environment to derived linguistic systems. A feature of this approach that distinguishes it from related efforts is the focus on deriving control for internal cognitive processes and external actions generally rather than communication systems specifically, with communication processes emerging only if they are part of the optimal policy.

This paper proceeds as follows: first, we review related work on language emergence and discuss ways in which our approach complements this work. Next, we move to an example (the "Treasure Box Domain") designed to illustrate the approach by exploring constraints leading to the emergence of structured utterances — here the systematic use of serial order and allocation of lexical items to aspects of the environment. Finally, we show how this domain, and the approach in general, can be extended to investigate more sophisticated phenomena and propose future directions of inquiry.

Related Work

Research into the origins of language has a rich and controversial history. Chomsky addressed it in his early work on generative grammar, prompting a longstanding debate on the extent to which language is a biological adaptation arrived at via natural selection (Chomsky, 1968; Pinker & Bloom, 1990; for a more recent treatment, see Hauser, Chomsky, & Fitch, 2002; Pinker & Jackendoff, 2005; Fitch, Hauser, & Chomsky, 2005; Jackendoff & Pinker, 2005). Chomsky's (Chomksy, 2010) own recent approach to the question attempts to minimize—in fact, nearly eliminate—the role of language-specific biological adaptation. A more recent line of research by Nowak and colleagues (Nowak, Krakauer, & Dress, 1999; Nowak & Krakauer, 1999; Nowak, Plotkin, & Jansen, 2000; Nowak, Komarova, & Niyogi, 2002), establishes a mathematical framework used to explore the evolution of language from the standpoint of computational learning theory and evolutionary game theory. This work also provides evidence for coding constraints that may have resulted in increased fitness for agents capable of multi-symbol utterances.

[1]The first two authors contributed equally to this paper.

Several recent computational experiments explore the notion that cultural adaptation and domain-general cognition may be sufficient for the emergence of language (Beckner et al., 2009, also see Christiansen & Chater, 2008; Steels, 1998; De Beule, 2008; Gong, Minett, Ke, Holland, & Wang, 2005). This work shows a number of features emerging from repeated interactions of pairs of computational agents in a population playing a language game. In a way, this work implicitly frames language emergence as a function of environment, agent, and learning mechanism. Our work attempts to remove the last of these and more explicitly address what aspects of environment and agent architecture are important—potentially leading to a more deeply explanatory account.

The questions we are interested in are in part orthogonal to these debates: we are not making claims about either domain-specificity or the mechanisms of learning or evolution, but rather the interplay of cognitive constraints and environmental pressures that lead to the emergence of particular language features as adaptive. By leaving the mechanism of adaptation unspecified, our approach is relevant to researchers working in both biological and cultural frameworks.

Our work also departs from the approaches above in that it does not create a pressure for language by explicitly rewarding cooperation or communication of a particular type. This approach considers communication not as an end-goal but rather as the means to obtain some primary reward such as sustenance, shelter or reproduction. This may give us a principled way to examine and sharpen what it is about language which directly contributes to effective behavior.

Environmental Pressures & Agent Constraints

Natural environments comprise extremely complicated sets of pressures acting on agents. A key part of the work in this approach is identifying tractable sets of specific pressures that are independently motivated by the study of the environments of early hominids or humans and that might plausibly be important in the emergence of language. It is not our intent in this initial exploration to undertake this identification systematically, but we propose here a few plausible candidates as starting points that suffice to illustrate the approach.

Many environments naturally limit agent's ability to observe and act. For example human beings can only manipulate small pieces of the natural world. Furthermore, knowledge and ability to act is not usually distributed uniformly among agents, making information sharing between agents potentially useful. The nature of tasks that must be performed by agents may limit how immediately information can be utilized, requiring memory and independent action. A related pressure is limitation on the lexicon size available to the agents for communication. This could require generalization and furthermore may be a natural consequence of coding constraints on noisy information transmission (see Nowak et al., 1999, for a complete discussion). Another important pressure might be temporal: environment dynamics might require speed or brevity in communication.

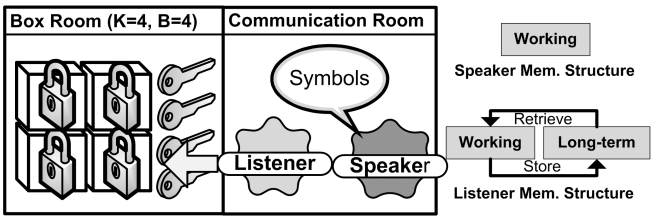


Figure 1: Treasure box domain.

Identifying structural constraints on agents is a second major requirement for this approach. These constraints may be independent of learning mechanisms and describe computational and physical capabilities of an agent. Our interests initially are in cognitive and perceptual constraints, such as limited attention and short-term memory. In the experiments below we adopt highly idealized versions of such constraints, but we always define computationally complete agents that can condition their control of internal and external processes on an internal state that combines memory and perception.

One concern about this approach is the prospect that pressures in the real world and human cognitive capabilities are so complex that our proposed analysis is impossible. However, this is an empirical question. It could very well be that careful investigation will yield simple features or ones that can be idealized while retaining their important aspects. It could very well be that careful investigation will yield simple features or ones which could be idealized while keeping their important aspects. It may also be possible to separate and explain specific language properties on a large scale.

Example: Treasure Box Domain

To demonstrate this approach to understanding language emergence we designed a set of experiments in which particular kinds of communication may emerge as optimal (or approximately optimal) behavior in a simple domain populated by two computationally limited agents. We describe next the structure of this domain and then discuss why it is of potential interest for our purposes—why we expect interesting linguistic systems to emerge.

Environment and agent structure

Figure 1 shows the Treasure Box domain. There are two agents, SPEAKER and LISTENER, who share the goal of opening a locked treasure box. These agents are in an environment containing two rooms: a first room, *communication room*, in which LISTENER can hear symbols uttered by SPEAKER and a second room, *box room*, in which there are B different boxes and K keys. At any one time, only one particular box contains treasure and can only be opened by one particular key. To solve this problem, LISTENER must go into *box room* and choose the correct box and key. However, LISTENER knows neither which box contains treasure nor which key opens it. The second agent, SPEAKER, knows the correct box and key, but cannot leave the *communication room* and therefore cannot open the box itself. Instead, SPEAKER

can communicate with LISTENER by uttering symbols from a lexicon of size S which LISTENER observes while in *communication room*.

When SPEAKER utters a symbol it is placed into LISTENER's immediate perception: a buffer holding a single symbol (working memory). In addition to the working memory store, LISTENER has a second memory location to hold a single symbol (long-term memory), the value of which cannot be observed without retrieving it. LISTENER can move a symbol from the working memory store into long-term memory and vice-versa (memory encoding and retrieval), but can only observe the symbol in working memory. The agent does, however, know whether long-term memory contains information. SPEAKER remembers the last symbol uttered in an observable working memory.

Speaker. This agent observes: (1) The box containing treasure; (2) the key which opens that box; and (3) last symbol it uttered. It can act by either (1) waiting or; (2) uttering a single symbol out of a limited set of size S.

Listener. This agent observes: (1) the room it is in; (2) whether it holds a key; (3) whether it holds a box; (4) whether its long-term memory contains information; and (5) the contents of its working memory. It can act by (1) moving to the box room; (2) encoding a symbol from working memory into long-term memory; (3) retrieving a symbol from long-term memory into working memory (4) picking up a specific key; or (5) picking up a specific box.

Dynamics. The domain is structured as an episodic task where each episode ends when LISTENER picks up both a box and a key (at which point the key is automatically used to open the box). If the key is correct and the box and the box contains treasure then *both* agents will receive a positive reward (of $+1$); otherwise no reward is received and a new episode begins. At the beginning of an episode the box containing treasure and the key that opens it are chosen randomly, LISTENER is returned to *communication room* holding neither key nor box, and both agents' memories are cleared.

Learning algorithm. Although the specifics of the learning mechanism are not the focus, we needed a method for discovering good agent behavior. Both agents use the ε-greedy Sarsa(λ) algorithm (Sutton & Barto, 1998). This algorithm learns by estimating state-action values $Q(s,a)$ that represent the best expected discounted sum of rewards over an episode that can be gained by following action a from state s and then the best policy thereafter (we initialize the Q values to 0). At each step actions are chosen greedily based on the current Q function except with a probability of ε when a random action is chosen instead (yielding exploration). We use a low exploration rate of $\varepsilon = 0.01$ across our experiments. After action a_t in state s_t at time t, the algorithm updates the Q value for all state-action pairs (s,a) according to their eligibility $e_t(s,a)$ as follows earlier actions by

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha\delta_t e_t(s,a),\ \forall s \in S, \forall a \in A$$

where before the update $e_t(s_t,a_t)$ is set to 1.0 and the eligibility for every other state-action is decreased by a multiplicative factor of γ,λ (we used $\lambda = 0.8$ for all of our experiments); the more recently a state-action pair is visited the higher its eligibility and the more credit or blame it gets for the temporal difference error $\delta_t = r_{t+1} + \gamma Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t)$ which is the the current estimated value of the resulting (s_{t+1},a_{t+1}) plus the reward r_t immediately gained minus the predicted value of the pair (s_t,a_t). The discount factor γ describes how much less future reward is valued compared to immediate reward; we used $\gamma = 0.8$ for all our experiments. The step-size parameter α controls how fast the algorithm incorporates new experience, we use $\alpha = 0.03$ in all of our experiments.

Why this domain is of potential linguistic interest

Without any communication the best LISTENER can do is to open an arbitrary box with an arbitrary key. Given KB possible box-key combinations the probability of success at each episode is $\frac{1}{KB}$. To improve beyond this, a communicative policy is required wherein SPEAKER informs LISTENER of the correct box and/or key in some way.

Different environmental pressures and agent constraints make different behaviors optimal. For example, we can explore how varying the size of the available lexicon alters behavior. If there are enough symbols ($S \geq KB$), then a single symbol suffices to describe each box-key combination. If there are at least $K+B$ but fewer than KB symbols, then two symbols are required but each box and each key could be given a unique symbol removing the need for symbol order. Finally, with $S = max(K,B)$ the meaning of symbols will have to be shared between boxes and keys, so order may be important. In all cases these interpretation of the symbols must be learned by both agents.

We can explore the effects of changing other constraints as well, such as agents' memory or environment structure. For example, if LISTENER can store two symbols in working memory, then consistent symbol order may not matter. If the environment is no longer divided into two rooms (so communication and box opening can occur simultaneously) symbol order might still matter, but the LISTENER may not need to encode anything into long-term memory, instead acting based on the contents of its working memory at every step—in effect becoming a situated instruction-taker.

Linguistic Properties of Emergent Policies

We conducted three sets of experiments (eight individual experiments) to demonstrate how environmental pressures and agent constraints jointly effect communication properties; the experiment structure and results are summarized in Table 1. In all experiments the number of boxes and keys is equal $K = B = 4$. The first set is the domain originally described with two separate rooms where LISTENER has a working memory of one symbol and a long-term memory of one symbol. The second set modifies the agent constraints by giving the LISTENER two symbols in working memory (no long-term memory). The third set changes the environmental pressures by removing the room separator.

Table 1: Summary of three sets of experiments and policies learned. See text for detailed description.

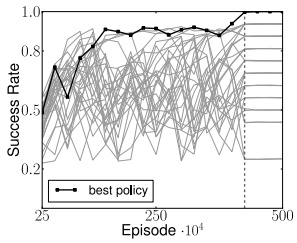
ENVIRONMENT	AGENT MEMORY	LEXICON SIZE (S)	PROPERTIES OF EMERGENT LINGUISTIC SYSTEM
Two Rooms	one symbol working memory + one symbol long-term memory	3	Association and systematic order, where in addition single symbols uttered in isolation denote specific box-key combinations. Can only achieve 75% success.
		4	Association and systematic symbol order. SPEAKER first describes the box, then the key (see Figure 2b).
		8	Highly context-dependent and idiosyncratic symbol meanings. For example *key 2* is represented by *symbol 4* if uttered before box, but *symbol 5* after.
		16	Each symbol denotes a box-key combination. For example symbol 5 means *key 1* and *box 1*.
Two rooms	two symbol working memory (no long-term memory)	3	Similar to case with 3 symbols above.
		4	Complex lexical forms. Describes entire box-key combination with two symbols which can be observed simultaneously by LISTENER effectively creating a 2-symbol length word (see Figure 3b).
One room	one symbol working memory + one symbol long-term memory	3	Symbols act as direct orders to LISTENER, but otherwise policy is similar to the cases of 3 symbols above.
		4	Association and symbol order, but no storing or retrieving from long-term memory is necessary because LISTENER can act immediately upon hearing a symbol (see Figure 4b).

Experiment set 1: Exploring constraints on the lexicon. We explore four different lexicon sizes: $S = 16$, $S = 8$, $S = 4$, and $S = 3$. Figure 2 shows 30 independent learning trajectories for each value of S. The high variance is due to the nature of the learning algorithm which may not converge for both agents every trial (or may get stuck on a less-than-optimal policy)—but what we are interested in are the best policies learned (because the mechanism used can be improved significantly beyond our initial implementation of Sarsa(λ) with fixed parameters across all experiments).

The first four rows of Table 1 summarize the results. Here we will discuss the resulting policies in more detail. For 16 available symbols, as expected, a different symbol is associated with each box-key combination and the agents arrive at perfect performance. With eight symbols, again the best performing policies use two-symbol utterances for each box-key combination, but not always in the same order (i.e. for some combinations keys are uttered first and in other boxes are uttered first). For the case of four symbols, the best performing policies communicate box and key in a particular order, with each symbol able to refer to either box or key (see Figure 2b). Of particular interest is that the the agents settle on a consistent order across box-key combinations, but this order might be different over seperate experiments: the linear position is necessary but the specific order is not. Finally, for the case of only three symbols the agents again learn a policy where linear symbol order matters. Curiously, this alone should only afford success in 56% of combinations; some policies however achieved 75% success. The policy succeeds in the additional box-key combinations by associating each with a single symbol uttered in isolation. That is, with limitations in symbol size utterance length becomes informative in addition to positional information.

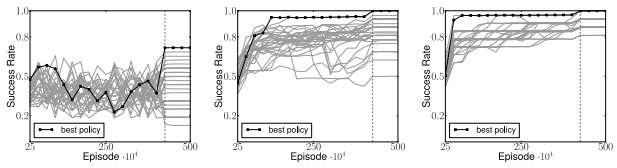
As we can see, this method of systematically altering only a single constraint (lexicon size) yields broad variation in linguistic properties even in this extremely simple domain, including the denotation of symbols and the use of order information. The case of three and four symbols suggests that limited memory (paired with environmental pressures) leads to the systematic use of symbol order in optimal performance, especially when the lexicon size is limited.

Experiment set 2: Modified agent constraints. Here our aim is to explore further what specific constraints led to the systematic use of order in Experiment 1. We alter the constraints on the agents by allowing the LISTENER two symbols in working memory instead of one (and no long-term memory). All the other dynamics of the Treasure Box Domain are kept constant. The actions of *store* and *retrieve* have new



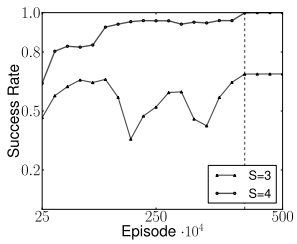
Timesteps	Listener				Speaker	
	Held	WM	LTM	Action	M	Action
	none	*e*	*e*	Wait	*e*	Utter 3
	none	3	*e*	Encode	3	Utter 2
	none	2	3	Move	2	Utter 2
	none	2	3	Get_K1	2	Utter 2
	K	2	3	Retrieve	2	Utter 2
	K	3	*e*	Get_B1	2	Utter 2

(a) $S = 4$ (b) Sample of policy[2] for $S = 4$.



(c) $S = 3$ (d) $S = 8$ (e) $S = 16$

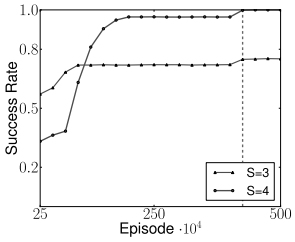
Figure 2: Experiment set 1: Exploring constraints on the lexicon. Each figure shows 30 learning curves in the Treasure Box Domain with $B = 4$ and $K = 4$. Success rate at each point is the average success rate over all episodes since previous point. Dotted line marks where learning and exploration are disabled. Best policy is highlighted and described in table 1. Figure (b) is a sample policy for $S = 4$ showing the significance of symbol order. In this case, agents have learned to associate string "3,2" with key 1, box 1, as can be seen in the rightmost column where SPEAKER utters first symbol "3" then symbol "2".



Timesteps	Listener				Speaker	
	Held	WM	LTM	Action	M	Action
	none	*e*	*e*	Wait	*e*	Utter 1
	none	1	*e*	Encode	1	Utter 4
	none	4	1	Move	4	Utter 2
	none	4	1	Get_K1	2	Utter 4
	K	4	1	Get_B1	4	Utter 2

(a) Best policies. (b) Sample of policy[2] for $S = 4$.

Figure 3: Experiment set 2: Modified agent constraints; LISTENER has two working memory locations. Left figure shows learning curves for best policies for $S = 3$ and $S = 4$. Right figure is a sample policy for $S = 4$ showing that the LISTENER can act according to the length-2 string in working memory: LISTENER's last two actions are box and key pickups without a retrieval in between, unlike the policy in figure 2.



Timesteps	Listener				Speaker	
	Held	WM	LTM	Action	M	Action
	none	*e*	*e*	Wait	1	Utter 1
	none	1	*e*	Get_K1	1	Utter 2
	K	2	*e*	Get_B1	2	Utter 1

(a) Best policies. (b) Sample of policy[2] for $S = 4$.

Figure 4: Experiment set 3: Modified environmental pressures: no room separator. Left figure shows learning curves for best policies for $S = 3$ and $S = 4$. Right figure is a policy sample for $S = 4$. The absence of a room barrier allows symbols to act as direct orders: the "utter 1" action by SPEAKER is followed by LISTENER's "get key 1" on the next time step.

semantics now: moving symbols between the two working memory locations. Figure 3 shows the best trial for each case in this experiment (for lexicon size of 3 and of 4). With 4 symbols in the lexicon, pairs of symbols can be used to describe each box-key combination. This is possible because unlike Experiment 1 both symbols are visible to the LISTENER (when both stored in memory) and thus there is no need for an association of order of symbol with object type (key or box). What is perhaps surprising about this result is that the more flexible agent structure in this experiment yields a simpler communication system, whereas the putatively more sophisticated linguistic system in Experiment 1 emerges as an adaptive response to the more computationally limited agent structure.

Experiment set 3: Modified environmental pressures. Here we alter the environmental constraints by removing the separator between the communication room and the box room. This modification relieves the pressure imposed by delay between communication and utilization effectively removing the need to remember information. Instead LISTENER can act immediately from SPEAKER's instructions. Figure 4 shows the best trial $S = 3$ and $S = 4$. For the case of 4 symbols, SPEAKER's utterances act as immediate instructions to LISTENER. Word order still matters, but when a particular symbol is uttered first it may correspond to a different object (box-key) than if uttered second. Furthermore, the second symbol uttered can have different meaning depending on the context. For example if LISTENER has already chosen a box, the second symbol will be associated with a key.

[2]Example policies show actions for the case key = 1 and box = 1. Each row is one time step; *e* means empty memory location. For readability, we are showing the contents of LISTENER's long-term memory and omitting current room. LISTENER does not have a "wait" action, but instead uses an action which has no effect (e.g. "pick up a key" while in the *communication room*). The SPEAKER's utterances do not impact LISTENER after it changes rooms so these actions are unimportant.

Conclusions and Looking Ahead

We have described and illustrated a novel approach to language emergence hypothesizing that specific properties of language may be understood as features of boundedly optimal policies to control problems imposed on computationally limited agents. What makes the approach distinctive is its emphasis on the shaping of linguistic systems by the joint constraints of agent and environment structure, and the emergence of such systems as the solution to the problem of how to optimally control both cognitive and physical actions in service of task goals (rather than communication goals). This means that there is no associative learning component or any other learning mechanism beyond the reinforcement learning algorithm described above. Any associations between symbols and objects or actions are arrived at not because the agents are explicitly trying to understand each other or arrive at shared symbol-meaning mappings, but rather implicitly as joint solutions to the control problem.

Our initial experiments yielded two key results. First, we have shown that even simple environments and agent architectures give rise to linguistic systems with interesting properties, including systematically structured utterances and flexible use of limited lexical resources. Second, we have shown that changes in environmental pressures or agent constraints may yield dramatic changes in optimal communication structure. Some constraints and pressures yield communication with systematic symbol order, other constraints yield policies that break the association between single symbols and single objects in the environment. The changes to environment and agent may seem small, raising the question of how a robust communication system can emerge, but in the context of the environment we explored the modifications are quite large. We expect small changes in a complex environment would not drastically alter the resulting communication systems. Furthermore, the fact that the communication system is strongly shaped by specific constraints of the cognitive architecture is also unproblematic, because we expect such constraints to be relatively stable across conspecifics. Indeed, to the extent that language is shaped by such constraints, this is good news for the cognitive scientist, because their detailed nature is likely to be more accessible that the relevant details of the shaping environments.

Our results suggest that there is promise in developing a broad systematic framework for studying language emergence by identifying mappings between pressures, constraints, and language properties independent of questions regarding the mechanisms of evolution or adaptation. Promising future avenues include investigating the emergence of compositional mechanisms like recursion, categorical features including distinctions between nouns and verbs, or more sophisticated uses of language for representation of internal mental states.

Acknowledgments: This work was supported by NSF grant IIS 0905146. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsors.

References

Beckner, C., Ellis, N. C., Blythe, R., Holland, J., Bybee, J., Ke, J., et al. (2009). Language Is a Complex Adaptive System: Position Paper. *Language Learning*, *59*, 1–26.

Chomksy, N. (2010). Some simple evo devo theses: How true might they be for language? In Y. H. Larskon R. K. Deprez V. (Ed.), *The evolution of human language: Biolinguistic perspectives.* Cambridge: Cambridge University Press.

Chomsky, N. (1968). *Language and mind*. New York: Harcourt, Brace & World.

Christiansen, M., & Chater, N. (2008). Language as shaped by the brain. *Behavioral and Brain Sciences*, *31*(05), 489–509.

De Beule, J. (2008). The emergence of compositionality, hierarchy and recursion in peer-to-peer interactions. In *Proceedings of the 7th international conference on the evolution of language* (pp. 75–82). World Scientific Pub Co Inc.

Fitch, W. T., Hauser, M. D., & Chomsky, N. (2005). The evolution of the language faculty: clarifications and implications. *Cognition*, *97*(2), 179–210; discussion 211–25.

Gong, T., Minett, J. W., Ke, J., Holland, J. H., & Wang, W. S.-Y. (2005, July). Coevolution of lexicon and syntax from a simulation perspective. *Complexity*, *10*(6), 50–62.

Hauser, M. D., Chomsky, N., & Fitch, W. T. (2002). The faculty of language: What is it, who has it, and how did it evolve? *Science*, *298*(5598), 1569-1579.

Jackendoff, R., & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of language (Reply to Fitch, Hauser, and Chomsky). *Cognition*, *97*, 211-225.

Nowak, M., Komarova, N., & Niyogi, P. (2002). Computational and evolutionary aspects of language. *Nature*, *417*(6889), 611–617.

Nowak, M., & Krakauer, D. (1999). The evolution of language. *PNAS*, *96*(14), 8028.

Nowak, M., Krakauer, D., & Dress, A. (1999). An error limit for the evolution of language. *Proceedings of the Royal Society*, *266*(1433), 2131.

Nowak, M., Plotkin, J., & Jansen, V. (2000). The evolution of syntactic communication. *Nature*, *404*(6777), 495–498.

Pinker, S., & Bloom, P. (1990). Natural language and natural selection. *Behavioral and Brain Sciences*, *13*(4), 707–784.

Pinker, S., & Jackendoff, R. (2005). The Faculty of Language: What's Special About it? *Cognition*, *95*(2), 201–36.

Steels, L. (1998). Synthesising the origins of language and meaning using co-evolution, self-organisation and level formation. In *Approaches to the evolution of language: Social and cognitive bases* (pp. 384–404). Edinburgh University Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

Linguistic Spatial Gestures

Leonard A. Breslow (len.breslow@nrl.navy.mil)
Anthony M. Harrison (anthony.harrison@nrl.navy.mil)
J. Gregory Trafton (Trafton@itd.nrl.navy.mil)
NCARAI, Naval Research Lab., Code 5515
Washington, DC 20375 USA

Abstract

We model the gestures accompanying spoken descriptions of spatial information and propose a conception of spatial gestures that differs from previous proposals by making a distinction between gestures used for thinking (cognitive gestures) and gestures used to help express predetermined ideas (linguistic gestures), and positing a tighter integration between gesture and language production in the latter than most previous researchers.

Keywords: gesture; spatial reasoning; language production

Introduction

Symbolic speech-accompanying gesture, representing spatial information, has lately been an area of active research (Alibali, 2005). Of particular interest is the relationship between gesture and the language it accompanies. In considering this relationship, we think it is useful to distinguish between gestures that help us determine *what* to communicate/express (*cognitive gestures*) and gestures that help us to express what we have determined to say, that is, gestures concerned with *how* to communicate (*linguistic gestures*). While these two functions clearly overlap in certain cases, we consider the distinction useful. Specifically, we argue that, in general, cognitive gestures lead language, whereas language leads gesture in the case of linguistic gestures.

Cognitive gesture leads language indirectly by facilitating thinking, thereby helping us determine what to say. Thus, they are used in situations with competing conceptual representations (Kita & Davies, 2009), high conceptual load (Melinger & Kita, 2007), mental rotation tasks (Chu & Kita, 2008), expert and novice scientific thinking (Trafton et al., 2006), and problem solving (Lozano & Tversky, 2006), among others. Such gestures are relatively independent of language, often expressing information different from that expressed in the accompanying language, and sometimes cognitively more advanced than the latter, e.g., in development (Alibali & Goldin-Meadow, 1993) or in problem-solving performance (Lozano & Tversky, 2006). While cognitive gestures sometimes aid communication (Lozano & Tversky, 2006), they are relatively independent of communication, as evidenced by their use when solving problems silently in solitude (Chu & Kita, 2008; Lozano & Tversky, 2006).

In contrast to cognitive gestures, linguistic gestures are more strongly tied to language and dependent upon language. They convey little or no information beyond what is expressed in the accompanying language (Beattie & Shovelton, 1999; So, Kita, & Goldin-Meadow, 2009), except where the respective roles of gesture and language are predetermined as in deixis ("Look at that!") or in language referring to gesture ("It was *this* big."). Neurological as well as behavioral evidence suggests the absence of priming of words by gestures in comprehension (Bernardis & Caramelli, 2007) or production (Beattie & Coughlan, 1999; Bernardis, Salillas, & Caramelli, 2008). On the contrary, language primes gesture comprehension (Bernardis & Caramelli, 2007) and cross-linguistic studies demonstrate that the grammatical organization of speech is predictive of the sequence and nature of symbolic gesturing (Kita & Ozyurek, 2003).

Also in contrast to cognitive gestures, linguistic gestures are typically associated with communication, as evidenced by the great reduction in gesturing when the listener cannot see the speaker (Alibali, Heath, & Myers, 2001) and the absence of gesturing outside of communication (e.g., in silence or solitude). However, we do not claim that linguistic gestures always *facilitate* communication, since people gesture even when speaking on the telephone (de Ruiter, 1995).

Note that the outward form of both cognitive and linguistic gestures may appear very similar – they are iconic gestures typically tied to a spatial representation of what is being thought or said. The types of gestures may be distinguished by the degree to which the gesturer has difficulty determining the spatial ideas he/she wishes to express, which may vary by population (e.g., child vs. adult) as well as by situation (e.g., problem-solving vs. simple description).

We will focus in the remainder of this paper on linguistic gestures. One question that researchers have considered is the extent to which the perceptual information being described by the speaker inputs directly into the generation of gestures, without the intermediary of language processing. Some argue that direct perception accounts for the few features of gestures that are not conveyed in the accompanying language (Kita & Ozyurek, 2003). Other theories (de Ruiter, 2007; Hostetter & Alibali, 2008) attempt to account for gesture solely on the basis of perception or imagery. Both types of theory are challenged in explaining the process by which perceptual features are selected for inclusion in gestural representation.

We propose a model of linguistic gestures that posits a tighter integration between gesture and language than most previous models (as does McNeill, 1992) by adopting a broader view of language representation than typically used.

Our approach draws on a recent linguistic theory proposed by Ray Jackendoff (2002), according to which language representation includes some irreducibly spatial components. It also draws on the construction grammar approach of Goldberg (1995), according to which linguistic structures containing both semantic and syntactic components are central to language processing. Combining these two approaches, we hypothesize that people select a construction before retrieving words and gestures. The construction provides an abstract plan for speaking that includes the semantic-syntactic information used in the retrieval of both words and spatial representations at appropriate places in the utterance. The spatial representations are the basis of symbolic gestures and so this approach helps to identify where specific gestures will occur. Following Jackendoff, we hypothesize that the spatial representations are abstract in nature (Avraamides et al.,2004). We propose that these abstract representations may be instantiated either as internal mental images or externally as gestures.

This account predicts that the information conveyed in linguistic gesture will be tightly tied to the accompanying language, since both language and gesture derive from the same construction. This helps to explain why linguistic gestures provide little information not included in the accompanying language. What little extra information is included in gesture is information required for the instantiation of an abstract spatial schema (a spatial element of a linguistic construction) in a particular situation. For instance, a gesture representing an observed leftward movement is usually performed in a leftward direction (Kita & Ozyurek, 2003), since a linear gesture must have *some* direction. But the gestural reproduction of the stimulus is limited to what is necessary to instantiate an abstract spatial schema as a physical hand movement. Thus, this account provides a mechanism for selecting perceptual features for inclusion in gestural representation, in contrast to unconstrained perceptual accounts (de Ruiter, 2007; Hostetter & Alibali, 2008). This account also helps to explain the observed temporal synchrony between gestures and utterances of similar meaning (McNeill, 1998).

Modeling Language

We evaluate our conception of linguistic spatial gesture by modeling the findings reported by Kita and Ozyurek (Kita & Ozyurek, 2003). Native speakers of English, Japanese, and Turkish were shown a cartoon and asked to describe it to another person. In one scene, a cat (Sylvester) jumps out the window of an apartment building, grabs onto a hanging rope and swings across the street to another building. In another scene, the cat, after swallowing a bowling ball, rolls down the street. English speakers described both path (down/across the street) and manner of locomotion (swing or roll) in a single clause, such as (with clauses marked by square brackets):

English-Swing: [The cat swings across the street.]

English-Roll: [The cat rolls down the street.]

In contrast, speakers of Japanese and Turkish (hereafter J/T) described path and manner in two separate clauses, paraphrased roughly as:

J/T-Swing: [[The cat goes across the street], [ ]]
J/T-Roll: [[The cat goes down the street], [as he rolls]]

Note that J/T lack an appropriate equivalent to "swings" in this context, an unusual lacuna in both these languages, and so the manner is not described verbally, but is often depicted by a gesture following the spoken clause, the position where a dependent clause describing manner normally occurs, as in the J/T-Roll sentence.

The clausal structure of the four sentences, above, corresponds to linguistic *constructions* as characterized by Goldberg (1995). A linguistic construction is a semantic-syntax pair that also specifies the mapping between semantics and syntax. While her theory focuses primarily on clausal constructions, Goldberg considers the construction framework to be applicable to all levels of the language down to words. Thus, the J/T description of the roll event consists of two constructions nested within a larger construction, as shown in J/T-Roll, above.

Table 1 outlines a simplified English intransitive motion construction, characterizing the semantic and syntactic components of the clause in English-Roll, adapted from Goldberg (1995).

Table 1: A simplified intransitive motion construction.

Semantics	THEME	MOVE	GOAL
Lexical items	"He"	"rolls"	"down the street"
Syntax	subject	verb	oblique prep. phrase

We omit many details. A construction has semantic content beyond that indicated by standard semantic categories, such as those shown here; for example, this construction denotes movement along a path. For Goldberg, the verb has a centrality not depicted here and constructions include rules for mapping from semantics to syntax that we omit. Note that the lexical items are not part of the construction, but instead are added to the construction in the course of its application.

We adopt a simplified process model for language production based on constructions, consisting of the following sequence:

1. **Construction retrieval/instantiation.** A construction is selected based on the match of its semantic components to the situation, in the process of which those semantic components are instantiated.
2. **Lexical retrieval.** Lexical items (e.g., words) are retrieved for each semantic component in turn (from

left to right in Table 1) based on the semantics-syntax mapping specified by the construction as well as by its semantic content.

In the course of the first, construction retrieval, step, semantic components in the construction are instantiated with concepts and/or spatial representations. Following Jackendoff (2002), we hypothesize that some semantic categories are instantiated with irreducibly spatial representations. In fact, Jackendoff argues that the semantics of the MOVE component in an intransitive motion construction is exclusively spatial in nature. Note in English, the MOVE component represents the *manner* of movement (e.g., swinging, rolling). In contrast, this manner of movement component is absent from the intransitive motion constructions in J/T; instead, the manner of movement is represented by a separate dependent clause following the intransitive motion clause.

We hypothesize that the instantiated spatial components of constructions at all levels (multi-clausal, clausal, lexemes), resulting from step 1, constitute the basis for gesturing during speech.

Modeling Gesture

Kita and Ozyurek (2003) categorize the manual gestures found to accompany utterances English/J-T-Swing/Roll, above, into one of three types:

1. *Manner only*: e.g., a circular motion for rolling.
2. *Trajectory only*: e.g., a straight-line motion from left to right.
3. *Conflated*: depicting both manner and trajectory, e.g., a looping left-to-right movement for rolling.

As a manner-only gesture is not possible for denoting swinging, only trajectory and conflated gestures accompanied the swing utterances. In general, the authors found that the language groups differed in their gestures in a manner corresponding to the structure of their utterances: English speakers often made conflated gestures only, whereas J/T speakers more often made manner only and trajectory only gestures. Note that the language groups did not differ in their overall production of conflated gestures, but in the tendency to produce *only* conflated gestures, which was more common in English. Based on these findings, the authors proposed that the production of gestures is influenced by the structure of language in the planning stage of speech production.

Kita and Ozyurek also noted that among all language speakers, the direction of gestures (e.g., left to right) generally corresponded to the direction observed in the cartoon, but was never mentioned in the utterances. On this basis, they posited a separate line of influence of perception on gesture, unrelated to language. In contrast to this, we propose a unified account of gesture and language production.

We hypothesize that the spatial components of constructions at all levels (discourse, multi-clausal, clausal, lexemes) constitute the basis for gesturing during speech. We explain the selection of spatial features of an event for gestural representation in terms of the requirement to instantiate an abstract spatial representation to produce both speech and gesture. Since a translation gesture must have *some* direction, the reproduction of the observed direction is simply part of this instantiation process.

Although Kita and Ozyurek did not report the correspondence between gesture and language in a fine-grained manner, we have inferred from their reported data the correspondence outlined in Tables 2 and 3. Note that there is no manner clause for Swing descriptions available to J/T speakers. We make certain assumptions based on the common observation that symbolic gestures co-occur with like-meaning language (McNeill, 1998). Thus, manner-only and conflated (manner+trajectory) gestures accompany manner language (the verb in English, the adverbial post-clause in J/T), while trajectory-only and conflated gestures accompany path language (the prepositional phrase in English, the intransitive motion clause in J/T). The relative frequency of the two possible gestures for the two respective language segments of interest (path vs. manner language) is the focus of our model.

Table 2. Language and accompanying gestures during Roll description observed and predicted by model.

		% Ss observed	**Model**
English			
Language	**Gesture**		
Manner verb	Conflated	66	51
Manner verb	Manner only	13	18
Path phrase	Conflated	53	68
Path phrase	Trajectory only	39	28
Japanese / Turkish			
Language	**Gesture**		
Manner clause	Conflated	59	76
Manner clause	Manner only	40	16
Path clause	Conflated	25	31
Path clause	Trajectory only	67	66

Table 3. Language and accompanying gestures during Swing description observed and predicted by model.

		% Ss observed	Model
English			
Language	**Gesture**		
Manner verb	Conflated	88	93
Path phrase	Conflated	81	88
Path phrase	Trajectory only	7	3
Japanese/Turkish			
Language	**Gesture**		
[Manner clause position]	Conflated	75	88
Path clause	Conflated	37	30
Path clause	Trajectory only	63	70

The construction approach provides a useful framework for understanding both planning and online production of speech. In the present context, it offers an explanation of how ongoing speech can be influenced by elements of the speech plan that are executed before and after the currently executed (spoken) element, an explanation that can be extended to gesture. Specifically, we hypothesize that spatial semantic components within the same construction will have a greater influence on one another (via priming, etc.) than those in separate constructions. Further, this influence will be greater the lower the shared construction is in the construction hierarchy, since spatial representations are more concrete and less abstract lower in the hierarchy. Thus, conflated gestures, representing both trajectory and manner, are proportionally more common during the path language in English than in J/T because the path language in English shares the same construction as the manner language, in contrast to J/T where manner language is in a separate low-level construction. However, conflated gestures do occur sometimes in J/T because the respective clauses describing path and manner are contained within the same higher-level construction.

Similarly, clause structure can help to explain how an executed gesture influences the selection of a subsequent gesture. In English, the type of gesture selected to express manner has a great influence on the subsequent gesture selected to express path, whereas in J/T there is no apparent influence of the selection of path-describing gesture on the subsequent manner-describing gesture. This finding is explained by the occurrence of the two gestures within a single clause in English, but in separate clauses in J/T.

Model of Gesture and Language

The models of gesture and language production were developed within ACT-R (Anderson et al.,2004). ACT-R is a hybrid symbolic/sub-symbolic production-based system. ACT-R consists of a number of modules, buffers, and a central pattern matcher. Since ACT-R is not well-suited to represent structured representations, such as nested linguistic constructions, we attempt to capture the retrieval of spatial representations using ACT-R's partial matching capability. Specifically, the relative similarity of pairs of related spatial representations is modulated to reflect their proximity in the construction hierarchy, as is their capability to prime one another.

To represent space, we have developed a version of ACT-R, ACT-R/E, that utilizes a spatial theory called Specialized Egocentrically Coordinated Spaces (SECS) (Harrison & Schunn, 2003). SECS provides two egocentric spatial modules, which are responsible for the encoding and transformation of representations in service of navigation (configural) and manipulation (manipulative). Our model currently includes configural spatial representations.

Non-default ACT-R parameter settings are listed in Table 4. Manner chunk similarity refers to the associative similarity between the manner chunk in a language construction and an imaginal or gestural spatial representation. Similarly for path chunk similarity. Note that similarities are greater in English than in J/T, reflecting the increased priming by a linguistic construction lower in the construction hierarchy compared to a higher-level construction. Overall, for both language groups, there was a higher rate of conflated gestures for the swing description than for the roll description, possibly due to the smaller number of gesture types available for swing (i.e., the absence of a manner-only gesture). This may explain the need for weaker manner chunk similarities for the Roll models relative to the Swing models. The reduction of base level learning rates in English relative to J/T reflects the priming of later gesture selection by the previously-selected gesture in English, unlike in J/T.

Table 4. Non-default ACT-R parameter settings.

Parameter	English swing	J/T swing	English roll	J/T roll
Enable partial matching	true	true	true	true
Activation Noise	0.3	0.3	0.3	0.3
Retrieval threshold	-6.0	-6.0	-6.0	-6.0
Base level learning rate	0.2	0.9	0.5	0.9
Manner chunk similarity	-0.1	-1.0	-0.9	-3.0
Path chunk similarity	-0.2	-1.0	-0.1	-1.0

In describing the Roll situation, the English-language model first retrieves, and instantiates the semantics of, an intransitive motion clause construction, based on the observed event (see Table 1.) The instantiated construction forms the plan for all further retrievals, gestures, and utterances for the clause. First, the model retrieves and utters the first clause argument, the THEME (e.g., "the cat). Next it retrieves a manner verb corresponding to the MOVE argument. The verb contains a spatial representation that strongly primes a *manner* gesture representation, but the clause construction itself carries path-following meaning and so contains a spatial representation that weakly primes a *trajectory* gesture representation—weakly, because the clause construction is a higher-level construction than the verb. Although the priming of a trajectory gesture is weaker than that of a manner gesture in English, it is stronger than the priming of the corresponding "non-matching" gestures in J/T, because those gestures are primed by a still higher-level construction. As a result, English speakers more often retrieved both manner and trajectory gesture representations, fusing them into a conflated gesture. However, when only the manner gesture representation is retrieved, then a manner-only gesture will be performed. The manner verb is then uttered together with the selected gesture.

Next, path description language (spec. a prepositional phrase) is retrieved based on the instantiated GOAL. Once retrieved, this path phrase's spatial trajectory representation strongly primes a trajectory gesture. At the same time, the construction's MOVE representation *weakly* primes the manner gesture, weakly because it is at a higher level than the path language. Also, if the manner gesture was retrieved and performed earlier with the verb, that earlier retrieval makes an additional contribution to its activation, making it more likely to be retrieved again; no such priming occurs in J/T because the two successive gestures occur in separate constructions. If the manner representation is retrieved together with the trajectory representation, then the GOAL utterance is accompanied by a conflated gesture. If only a trajectory representation is retrieved, then it is accompanied by a trajectory-only gesture.

The J/T models function in a similar manner to this illustration, differing primarily in the nested structure of its constructions.

Given that individual variability is typically quite high for gesturing, the predictions of our model are rather similar to the observed pattern of behavior (Tables 2 and 3) and were all within the 95% confidence interval. r^2 was .63 for Roll and .98 for Swing.

Discussion

We have introduced the contrast between cognitive and linguistic spatial symbolic gestures in hopes of resolving apparently conflicting evidence in the literature. Cognitive gestures help us to determine *what* to say in a spatially complex domain, while linguistic gestures help us to express what we have determined to say. Obviously these two types of gesturing may be intermixed in a given situation, but certain experimental situations clearly encourage one type of gesturing over the other for a given population.

With regard to linguistic gestures, we hypothesize that gestures are generated on the basis of spatial components within linguistic representations (Jackendoff, 2002). The grammatical framework we adopt is that of constructions (Goldberg, 1995) in which lexical items, clauses, and more complex linguistic expressions may all be viewed as constructions, i.e., semantic-syntactic pairings whose semantic content, we hypothesize, includes abstract spatial components. The spatial semantic content at all levels of the construction hierarchy constitutes the basis for gesturing.

From this viewpoint, linguistic gestures are largely constrained by language generation. Specifically, perceptual information is incorporated in gesture during the course of instantiating linguistic structures. This obviates the need to hypothesize a separate, independent source of perceptual input into gesturing, together with the problems such a hypothesis entails: of explaining that mechanism and, especially, of explaining the selection of perceptual features to represent gesturally. As the information conveyed in gesture is largely limited to that conveyed in language, it would appear inappropriate to posit an unconstrained source of perceptual input into gesture production.

From our perspective, linguistic gesture and language are intimately related. Our model is an explicit computational / process account of McNeill's proposal that gesture and speech arise from a single process of utterance formation (McNeil, 1992, p. 29-30).

Although not addressed in this model, many factors modulate the rate of gesturing, such as social stimulation (Alibali et al., 2001), exposure to perceptual vs. verbal information (Hostetter & Hopkins, 2002), etc. The idea of an activation threshold governing the elicitation of gesturing, proposed by Hostetter and Alibali (2008), may be useful in explaining the expression of spatial representations externally in gesture rather than internally in imagery.

Acknowledgments

This work was supported by the Office of Naval Research under job order number N0001409WX20173. The views and conclusions contained in this document should not be interpreted as necessarily representing official policies, either expressed or implied, of the U.S. Navy.

References

Alibali, M. W. (2005). Gesture in Spatial Cognition: Expressing, Communicating, and Thinking About Spatial Information. *Spatial Cognition and Computation, 5*(4), 307–331.

Alibali, M. W., & Goldin-Meadow, S. (1993). Transitions in learning: What the hands reveal about a child's state of mind. *Cognitive Psychology, 25*, 468-523.

Alibali, M. W., Heath, D. C., & Myers, H. J. (2001). Effects of visibility between speaker and listener on gesture

production: Some gestures are meant to be seen. *Journal of Memory and Language, 44*, 169-188.

Anderson, J. R., Bothell, D., Byrne, M. D., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.

Avraamides, M. N., Loomis, J. M., Klatzky, R. L., & Golledge, R. G. (2004). Functional Equivalence of Spatial Representations Derived From Vision and Language: Evidence From Allocentric Judgments. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 30*(4), 801-814.

Beattie, G., & Coughlan, J. (1999). An experimental investigation of the role of iconic gestures in lexical access using the tip-of-the-tongue phenomenon. *British Journal of Psychology, 90*, 35-56.

Beattie, G., & Shovelton, H. (1999). Do iconic hand gestures really contribute anything to the semantic information conveyed by speech? An experimental investigation. . *Semiotica, 123*, 1-30.

Bernardis, P., & Caramelli, N. (2007). Semantic priming between words and iconic gestures. In S. Vosniadou, D. Kayser & A. Protopapas (Eds.), *Proceedings of the Second European Cognitive Science Conference* (pp. 614-619). Delphi, Greece: Lawrence Erlbaum Associates.

Bernardis, P., Salillas, E., & Caramelli, N. (2008). Behavioural and neurophysio-logical evidence of semantic interaction between iconic gestures and words. *Cognitive Neuropsychology, 25*(7-9), 1114-1128.

Chu, M., & Kita, S. (2008). Spontaneous Gestures During Mental Rotation Tasks: Insights Into the Microdevelopment of the Motor Strategy. *Journal of Experimental Psychology: General 137* (4), 706–723.

de Ruiter, J. P. (1995). Why do people gesture at the telephone? In M. Biemans & M. Woutersen (Eds.), *Proceedings of the Center for Language Studies: Opening, Academic Year 95-96* (pp. 49-55). Nijmegen: University of Nijmegen.

de Ruiter, J. P. (2007). Postcards from the mind: The relationship between speech, imagistic gesture, and thought. *Gesture, 7*(1), 21-38.

Goldberg, A. E. (1995). *Constructions: A construction grammar approach to argument structure*. Chicago: University of Chicago Press.

Harrison, A. M., & Schunn, C. D. (2003). *ACT-R/S: Look Ma, No "Cognitive-map"!* Paper presented at the International Conference on Cognitive Modeling.

Hostetter, A. B., & Alibali, M. W. (2008). Visible embodiment: Gestures as simulated action. *Psychonomic Bulletin & Review, 15*(3), 495-.

Hostetter, A. B., & Hopkins, W. D. (2002). The effect of thought structure on the production of lexical movements. *Brain and Language 82*, 22–29.

Jackendoff, R. (2002). *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford: Oxford University Press.

Kita, S., & Davies, T. S. (2009). Competing conceptual representations trigger co-speech representational gestures. *Language and Cognitive Processes, 24*(5), 761-775.

Kita, S., & Ozyurek, A. (2003). What does cross-linguistic variation in semantic coordination of speech and gesture reveal? Evidence for an interface representation of spatial thinking and speaking. *Journal of Memory and Language, 48*, 16-32.

Lozano, S. C., & Tversky, B. (2006). Communicative gestures facilitate problem solving for both communicators and recipients. *Journal of Memory and Language, 55*, 47–63.

McNeill, D. (1992). *Hand and mind: What gestures reveal about thought*. Chicago: University of Chicago Press.

McNeill, D. (1998). Models of Speaking (To Their Amazement) Meet Speech-Synchronized Gestures. In D. McNeill (Ed.), *Language and gesture: window into thought and action.* Hillsdale, NJ: Lawrence Erlbaum Assoc.

Melinger, A., & Kita, S. (2007). Conceptualisation load triggers gesture production. *Language and Cognitive Processes, 22*(4), 473-500.

So, W. C., Kita, S., & Goldin-Meadow, S. (2009). Using the Hands to Identify Who Does What to Whom: Gesture and Speech Go Hand-in-Hand. *Cognitive Science, 33*, 115-125.

Trafton, J. G., Trickett, S. B., Stitzlein, C. A., Saner, L., Schunn, C. D., & Kirschenbaum, S. S. (2006). The relationship between spatial transformations and iconic gestures. *Spatial Cognition and Computation, 6*(1), 1-29.

When to Switch? Understanding How Performance Tradeoffs Shape Dual-Task Strategy

Duncan P. Brumby (Brumby@cs.ucl.ac.uk)
Nina del Rosario (Nina.Del@gmail.com)
Christian P. Janssen (C.Janssen@ucl.ac.uk)
UCL Interaction Centre
University College London
London WC1E 6BT UK

Abstract

A novel dual-task paradigm was used to investigate how people adapt their task interleaving behavior to meet a specific performance objective. The study required participants to encode and enter a series of route instructions from a secondary display while driving a simulated vehicle. Explicit instructions were given to give greater priority to either safe driving or rapid completion of the secondary navigation task. Results showed that participants met the required task objective by varying the frequency and duration of visits to the secondary task display, and by also varying the amount of time given up to steering control in between visits. We explain these data using a framework for modeling driver distraction effects. The model predicted the observed shift in task performance between the two focus conditions and also the observed change in task interleaving strategy. Taken together these results support the idea that people can strategically control the allocation of attention in multitask settings to meet specific performance criteria.

Keywords: Multitasking, cognitive modeling.

Introduction

Consider for a moment a driver following a set of written directions to reach an unfamiliar destination. As the driver approaches a junction, they might want to consult their directions, and in doing so must consider the risks of taking their eyes off the road ahead. A safe driver, given the opportunity, might pull over to study their directions, or if this is not possible, they might choose to make many brief glances to the instructions. A risky driver, on the other hand, may choose to look away from the road for prolonged periods to study the directions in detail. In this way, the frequency and duration of attention shifts between tasks is determined by the relative importance of each task, and also a judgment of safe and acceptable behavior.

It is well known that in many multitasking situations, such as the one sketched above, constraints on the human cognitive architecture limit the extent to which tasks are performed in parallel (Meyer & Kieras, 1997). How people control the allocation of resources to multiple concurrent tasks is a topic of considerable theoretical and practical interest (e.g., Navon, & Gopher, 1979; Norman & Bobrow, 1975; Salvucci & Taatgen, 2008; Wickens, 2002).

One important application of multitasking theory has been to understand driver distraction. Driving is a safety critical task performed by millions of people on a daily basis, and with the growing ubiquity of mobile and in-car devices there are concerns about the deleterious effects of driver distraction. In this area, many studies have investigated the impact of cell phone dialing on driving performance. Typical results show that drivers tend to dial chunks of digits at a time, returning their attention to driving in between each chunk (Brumby, Salvucci & Howes, 2009; Salvucci, 2005). This pattern of task interleaving might reflect the fact that the dialing task has a strong representational structure that is difficult to disrupt, and this could be used to guide decisions about when to switch attention between tasks (Salvucci, 2005). But how might people decide how to interleave tasks in situations where there are no natural cues to guide this decision?

Salvucci and Taatgen's (2008) threaded cognition theory assumes that relatively complex multitasking behavior can emerge from a simple bottom-up process without the need for any explicit top-down control structures. The theory assumes that the cognitive system processes task threads using a least-recently-processed scheduling heuristic. While this theory offers a parsimonious account of multitasking behavior, it is not clear how this account allows the cognitive system to make strategic decisions to favor one task over another. Indeed, a large body of empirical work demonstrated that people can make explicit decisions about how to allocate attention to different tasks in multitask settings by prioritizing performance on one task over another (e.g., Brumby et al., 2009; Horrey et al., 2006; Gopher et al., 1982; Gopher, 1993; Wang et al., 2007).

One possibility for how people might adapt their dual-task strategy to meet a specific task objective is that they monitor the amount of time that has elapsed since they last checked on the more important task. Kushleyeva, Salvucci, and Lee (2005) found that when participants were required to monitor a safety-critical dynamic task, they adapted their monitoring behavior to changes in the temporal demands of the task. This suggests that the safer driver in the example above might simply set a lower threshold for the amount of time that they are prepared to take their eyes off the road, and in doing so, will interleave attention between tasks more frequently.

Another possibility is that people select strategies to meet a desired dual-task performance tradeoff objective. Brumby, Salvucci, and Howes (2009) have shown that in the case of manually dialing a standard US telephone number while driving, dialing three or four digits at a time is a particularly efficient strategy because any more interleaving incurs additional time costs without significant improvement in lane keeping, and any less interleaving sacrifices safety. To

demonstrate this claim, Brumby et al. derived performance predictions for a range of dual-task strategies using a computational model. This approach of explicitly considering the performance tradeoffs involved for choosing between various dual-task allocation strategies is similar to that of defining a Performance Operating Characteristic (Norman & Bobrow, 1975; Navon & Gopher, 1979). The analysis by Brumby et al. showed that one limitation of the dialing-while-driving paradigm is that interleaving at the natural subtask boundaries of this task often corresponds with the most efficient dual-task interleaving strategy, in terms of completing the secondary dialing task in a relatively safe and timely manner.

In this paper, we investigate multitasking behavior using a novel dual-task paradigm. The paradigm, developed by Del Rosario (2009), requires participants to look at a secondary display to encode and enter a series of route instructions while driving a simulated vehicle. The benefit of this paradigm, over the classic dialing-while-driving paradigm, is that it does not have an external representational structure that can be used to guide decisions about when to interleave. Thus, participants are free to interleave the tasks how they like.

We use this paradigm to investigate how people adapt their dual-task interleaving behavior to meet varying performance objectives. In particular, we manipulate the experimental instructions and feedback given to participants to encourage either safe driving or rapid completion of the secondary navigation task. We consider how this change in task objective affects task performance and also the decision about when to interleave attention between tasks. Finally, we seek to apply Brumby, Salvucci, and Howes' (2009) model of how people interleave cell phone dialing and driving to this novel dual-task paradigm. An important question is whether the model will generalize to this new task setup, and if so, whether it will predict how people choose to interleave in each condition.

Experiment

Method

Participants. Sixteen participants (five female) took part in the study. Participants were unpaid volunteers, aged between 21- and 42-years (*M*=28.3 years). All had a valid driver's license and at least two years of driving experience.

Materials. The experiment used a dual-task setup in which participants had to complete a secondary navigation task while driving a simulated vehicle. Figure 1 shows how the two task displays were arranged.

For the driving task, participants were required to navigate the center lane of a three-way highway environment. The simulation environment was displayed on a 30-inch monitor and controlled by a Logitech G25 Racing Wheel. Participants were only required to steer the vehicle to maintain a central lane position. The vehicle's speed was held at a constant 55 miles/h (88.5 km/h). To reinforce safe lane keeping, safety cones were placed at either side of the

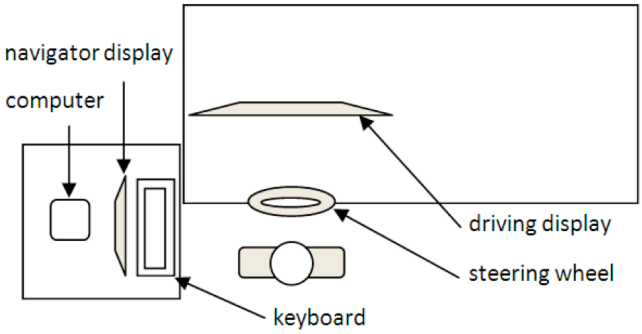


Figure 1. A schematic representation of how the driving and navigation displays were arranged.

driver's central lane. Noise was added to the vehicle dynamics, causing the vehicle to gradually drift about in the lane. This meant that the participant had to actively control and monitor the vehicle's lateral position and heading to maintain a central lane position.

For the navigation task, participants had to look at and enter a sequence of ten directions (lefts or rights). The to-be-entered sequence was randomly generated with the constraint that five left and five right directions were included and that there were no more than three consecutive repeating directions. The sequence of commands was represented either graphically (<=) or textually ("Left"), and was presented as a single vertical list on a 17-inch monitor positioned to the left of the participant (see, Figure 1).

The experiment was designed so that participants would be forced to sequentially interleave their attention between the two tasks. This was achieved by allowing only one of the task displays to be visible at any one time. By default the driving display was visible and the navigator display was blanked out. Participants activated the navigator display by moving their left hand from the steering wheel and using it to hold down the space bar on the keyboard in front of the navigator display. While the space bar was depressed the navigator display was presented and the driving display was blanked out. This meant that participants could not monitor the vehicle's position in the lane while encoding instructions for the navigation task. After viewing the instructions on the navigator display, participants had to return their hand to the steering wheel to use the left and a right paddle controls positioned under the steering wheel to enter the route instructions from memory.

Entry errors on the navigation task were associated with a time cost. If an input error occurred (e.g., a left paddle action was performed when a right action was required), the trial was terminated and the participant was instructed that they had to repeat the trial with a new list of instructions.

Design. A 2x2x2 (task-focus x representation x visual cue) mixed design was used, where task-focus was the between-subjects factor. To manipulate task priority, participants were instructed to either focus on completing the secondary navigation task as quickly as possible (the navigation-focus condition) or to focus on keeping the car as close as possible to lane center (the steering-focus condition).

Features of the secondary navigation task were manipulated as within-subjects factors. The route instructions were presented in a graphical or a textual format. In addition, a salient visual cue, indicating the current position in the list, was either present or absent.

The main dependent variables of interest were the time taken to complete the secondary navigation task and the impact that completing this task had on driving performance. The driving simulator logged the lateral distance of the vehicle from the center of the lane at a rate of 200 Hz. Driving performance was indexed by calculating the root mean square error (RMSE) of these lateral deviation samples over the period of time that the participant was working on the secondary navigation task. In addition, we were also interested in how participants chose to interleave the two tasks. To index task interleaving we consider the number and duration of each secondary task visit, as well as the time in between two visits.

Procedure. Participants were randomly assigned to one of the focus conditions, with the exception that effort was made to balance gender across conditions. Participants were given an opportunity to practice both the navigation and driving task separately. Once familiar with each task, participants completed four blocks of dual-task trials, one for each of the route representation and visual cue conditions. Trials were grouped by condition, and the order was counter-balanced across participants. For each block, participants were required to complete 10 error-free trials, up to a maximum of 15 trials per block. This dissuaded participants from making errors on the secondary navigation task.

Experimental instructions were given to encourage participants to prioritize either safe driving (steering-focus) or rapid completion of the navigation task (navigation-focus). To reinforce these instructions participants received feedback at the completion of every trial on their performance on the relevant variable. Specifically, participants in the steering-focus condition received feedback about the vehicle's RMSE lateral deviation, while participants in the navigation-focus condition received feedback on total trial time.

Results and Discussion

Due to space limitations we do not report data on how task performance was affected by manipulating features of the navigation task (see, Del Rosario, 2009, for details). Instead, we focus our analysis on how varying the instructions given to participants to prioritize one task over the other affected performance and decisions of how to interleave tasks. The primary dependent measures of interest were the time taken to complete the secondary navigation task and the lateral deviation of the vehicle from the center of the lane. We consider four separate measures to index task interleaving strategy: the number of visits to the navigator display per trial, the average duration of each visit, the number of navigation task items entered following each visit, and the average time between visits.

Figure 2 shows task time for the navigation task plotted against average RMSE lateral deviation for the driving task. There is a clear effect of task objective on how participants

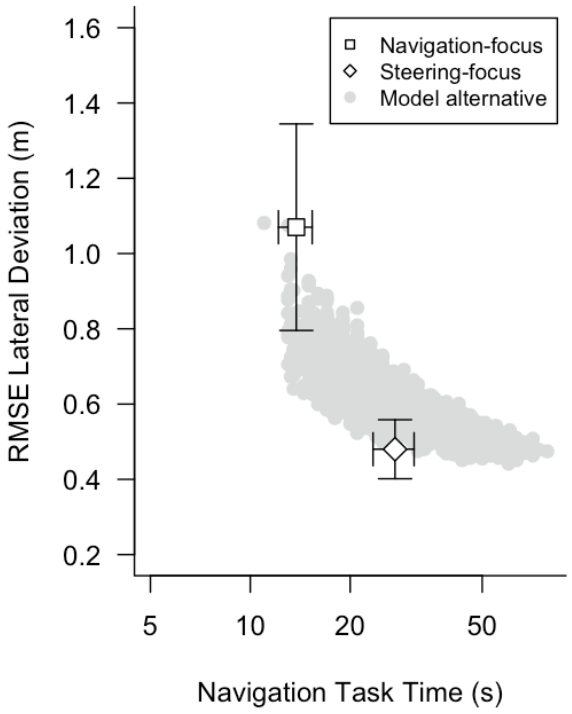


Figure 2. Data plot showing task time and RMSE lateral deviation across for varying task objectives. Error bars on human data points represent 95% confidence intervals. Model data points show performance predictions for different task interleaving strategies.

traded performance between the two tasks, in that, participants that were instructed to prioritize the navigation task completed it relatively quickly (M=13.76s, SD=2.31s), but in doing so had poor lateral control of the vehicle (M=1.07m, SD=0.41m). Conversely, participants that were instructed to prioritize safe driving completed the navigation task relatively slowly (M=27.30s, SD=5.57s) but were better able to maintain lateral control of the vehicle (M=0.48m, SD=0.10m). A 2x2x2 mixed factorial ANOVA found a significant effect of task objective on task time, $F(1,14)=40.26$, $p<.001$, $MSE=72.76$, and RMSE lateral deviation, $F(1,14)=15.87$, $p<.001$, $MSE=.35$.

We were also interested in participants' interleaving strategy, which was indexed by considering when participants choose to access the navigation task display. The data presented in Figure 3 show that the reason why participants in the steering-focus condition were better able to maintain lateral control of the vehicle than participants in the navigation-focus condition was because they made more visits to the navigation display (4.5 visits vs. 3.3 visits), $F(1,14)=3.67$, $p=.07$, $MSE=6.49$, entered fewer items following each visit (2.4 items vs. 3.4 items), $F(1,14)=5.19$, $p=.04$, $MSE=3.23$, and gave up more time to steering control between visits to the secondary display (5.34s vs. 2.57s), $F(1,14)=21.05$, $p<.001$, $MSE=6.25$.

The results of the study show that participants in the steering-focus condition interleaved more frequently and

spent more time in between glances to the secondary display stabilizing the vehicle than participants in the navigation-focus condition. However, it is not immediately clear why participants adapted their strategy in the way that they did. Changing the task priority lead to only a single extra item, on average, being encoded and entered following each visit to the secondary display. In contrast, participants spent nearly twice as long in between visits to the navigation display in the steering-focus condition. But why did participants opt to spend more time between visits rather than interleave much more frequently? To explain the observed pattern of task interleaving we apply a modeling framework developed to explain behavior in a dialing-while-driving paradigm (Brumby et al., 2007, 2009).

Model

Our modeling approach focuses on deriving performance predictions for various strategies for completing the navigation task while driving. The model represents basic task operators (i.e., encoding a single instruction from the navigation display, or performing a steering control update) as discrete processing units that are limited by a serial bottleneck. Within this framework, we systematically consider every possible dual-task strategy that could have been adopted. Specifically, given that the navigation task required participants to enter 10 route instructions, we can consider at least $2^9 = 512$ different task interleaving strategies (i.e., where strategies differ in terms of whether after encoding an item, another item is encoded or attention is returned to driving). For each of these strategies we also consider varying the amount of time that is given up to steering control in between visits to the secondary display.

We assume that glancing at the navigation display interferes with steering control. We estimate core parameters for the navigation task directly from the data. With these parameters fixed, we derive performance predictions for various dual-task interleaving strategies using a pre-existing model of steering control processes. For each strategy we derive predictions for critical performance metrics, namely, task time and lane keeping performance. The aim of this analysis is to explain the observed shift in dual-task performance between conditions, and also the precise change in low-level task interleaving behavior.

Navigation task. The navigation task is modeled at the granularity of the time taken to encode and enter route instructions. We estimate the time taken to perform these basic activities from the empirical data. Specifically, we estimate the time taken to:

- Shift attention from one task to the other
- Encode an item from the navigation display
- Input an instruction using the paddles

The time to switch attention from the secondary display to the driving task can be approximated by considering the average time between the release of the space bar (signaling the end of a visit) and the first paddle action being performed after the visit. Analysis shows that the average time between these events was approximately 1 second. A limitation of this measure as index of the cost of switching attention between tasks is that it assumes that the participant immediately commenced entering the instructions prior to returning their hand to the steering wheel.

We can approximate the time needed to encode a single route instruction by assuming that the number of items entered after a visit corresponds to the number of items that were encoded during that visit. Taking the average visit duration, we can calculate the average encoding rate to be approximately 500ms per item (i.e., in the navigation-focus condition, visits were on average 1.67s long and 3.4 items were entered after each visit). This assumes that participants never encoded items that were later forgotten or simply not entered. We shall revisit the implications of this assumption in the general discussion.

Finally, to estimate the time taken to input an instruction using the paddle, we consider the average time between two consecutive paddle entries. This yields an average time interval of 250ms between each paddle event. We assume that participants were able to perform steering updates while using the paddle to enter the route instructions, and that all instructions were entered before the next visit occurred. With these basic parameters set we can consider how this task might have interfered with driving performance.

Driving task. We use a simple mathematical model, taken from Brumby, Salvucci, and Howes (2009), which describes how people tend to adjust the heading of a vehicle based on its position in the lane. The model captures the basic idea that as the vehicle strays closer to the lane boundary, drivers react by making sharper corrective steering movements, which in turn, increase the lateral velocity of the vehicle, returning it to a central lane position more rapidly. The model assumes that discrete steering control updates are performed once every 250ms, which adjust the lateral velocity of the vehicle as follows:

$$Velocity = 0.2617 \times LD^2 + 0.0233 \times LD - 0.022 \quad (1)$$

where, *LD* represents lateral deviation from lane center, and there is an upper bound on velocity of 1.7m/s. In between steering updates, external factors can influence the vehicle's heading. To model this, we permute the vehicle's heading every 50 milliseconds with a value drawn from a Gaussian distribution with a mean of zero and a standard deviation of 0.09. Next we describe how this model is used to derive predictions of changes in a simulated vehicle's lateral deviation over time given discrete periods of driver attention and inattention.

For each of the 512 different strategies, we consider alternatives that give more or less time up to steering control in between visits to the navigation display. Specifically, we consider steering periods of between 250ms and 5000ms, at intervals of 250ms. This combined with the number of basic task interleaving strategies considered yields a fairly large set of 6,644 alternatives. For each, 50 simulations were run and performance averaged.

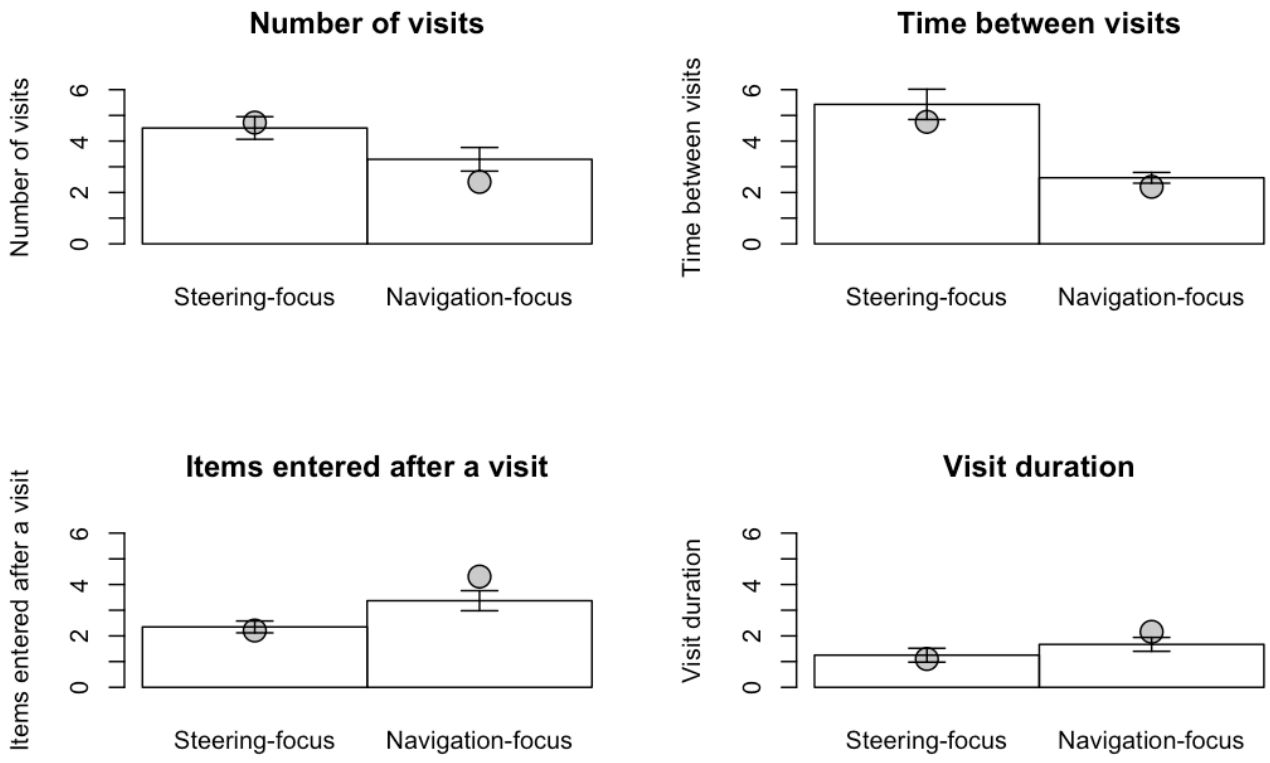


Figure 3. Data and model predictions for various navigation task measures. Bar charts show human data, with error bars representing standard errors of the mean. Circular data points represent model predictions for each priority condition. The values are the means for model alternatives that fall within the Confidence Interval in Figure 2 (see text for details).

Model Results

Figure 2 shows the predicted RMSE lateral deviation and task time for each of the 6,644 strategies that were evaluated along with the human data for each priority condition. The model predicts a clear dual-task performance tradeoff between strategies that complete the navigation task quickly and have relatively poor lane keeping performance, and those that complete the navigation task more slowly giving relatively better driving performance.

The shape of the tradeoff curve predicted by the model is noteworthy. There is a clear tipping point where improvements in lane keeping performance become smaller with increased task time. The human data for the steering-focus condition lie at this tipping point in the tradeoff curve, suggesting that participant adapted their strategy to meet the performance objective of minimizing lateral deviation while completing the secondary task in a reasonable amount of time (note that time is represented on a logarithmic scale). In contrast, data from the navigation-focus condition lie at close to the leftmost extreme of the strategy space, where faster performance is associated with poor lane keeping.

Figure 2 shows that there are many different strategies that fall within the predicted performance bounds of the human data for each condition. To get a better sense of how this performance tradeoff was achieved, we consider how these strategies allocated attention between the tasks. Specifically, we consider for each condition the subset of strategies that fall within the 95% confidence interval (CI) of the human data for each condition.

For the navigation-focus condition there were 34 strategies that fell within the CIs of the human data, while for the steering-focus condition there were 307 strategies that fell within the CIs of the human data. For each of these best-fitting strategies we define the same four measures of task interleaving behavior used in the analysis of the human data (i.e., the number of visits to the navigator display per trial, the average duration of each visit, the number of navigation task items entered following each visit, and the average time between visits). For each measure, we calculate the mean across the subset of best fitting strategies for each condition. In doing so, we get a better sense of how the best fitting strategies for each condition differed, and can compare these indexes of behavior to the human data.

Figure 3 shows these mean model predictions along with the corresponding human data for each condition. The fit of the model to these low-level task interleaving measures is remarkable, in that the model explains why participants in the steering-focus condition would have chosen to double the time between visits and encode one extra item per visit in order to reach the tipping point in the tradeoff curve.

General Discussion

A novel dual-task paradigm was used to investigate how people adapt their behavior to meet a specific performance objective. In the study, participants were required to encode and enter a series of route instructions while driving a simulated vehicle. Explicit instructions were given to participants to give greater priority to either safe driving or rapid completion of the navigation task. Results showed that participants met the required task objective by varying the number and duration of visits to the navigation display, and by also varying the amount of time given up to steering control between visits. These findings support the idea that people can strategically allocate attention in multitask settings (e.g., Brumby et al., 2009; Horrey et al., 2006; Gopher et al, 1982; Gopher, 1993; Wang et al., 2007).

We explain participants' decisions about how to allocate attention using an existing framework for modeling driver distraction effects (Brumby et al., 2007, 2009). The model represents basic task operators as discrete processing units that are limited by a serial bottleneck. To apply the model to this new dual-task context, a handful of parameters for the navigation task had to be estimated from the data (i.e., the time taken to encode a single instruction from the navigation display, shift attention back to road, and enter that instruction). With these basic timing estimates fixed, we model the effects of various allocation policies for attending to the secondary navigation display for critical task performance metrics.

The modeling results help explain the observed shift in task performance between the two focus conditions. The model predicts a classic dual-task performance tradeoff between safer driving and shorter task time. Interestingly, the tradeoff curve has a clear tipping point, after which improvements in lane keeping performance become relatively small with increased time investment. Human performance data from the steering-focus condition lie close to this tipping point, and remarkably the modeled strategies in this region of the strategy space corresponded with those adopted by participants.

However, the model did not explain data from the navigation-focus condition as well. Specifically, it under-predicted the number of visits made to the secondary display and over-predicted the number of items entered after each visit (see, Figure 3). The likely explanation for this departure is that the model assumes a perfect and limitless memory, which could enter all ten of the route instructions after a single visit. This is clearly an implausible assumption given the known limits on memory. This aspect of the model could be informed by considering how many items participants would copy over in a single-task setting. Alternatively, we could build on existing work that has modeled memory retrieval processes in similar tasks. For instance, Gray et al.'s (2006) work on modeling the impact of memory constraints in the Blocks World paradigm.

Moreover, because of space limits we could not present an analysis of how features of the navigation task affected performance. Del Rosario (2009) reports that participants could encode textual information faster than graphical information. Future work should point out how the model might explain any shift in strategy based upon changes in time take to encode an item from the display.

In summary, we have used a novel dual-task paradigm to demonstrate that people can strategically allocate attention in multitask settings. A model was used to explain why particular strategies might have been favored in terms of the shape of the performance tradeoff between safer driving and shorter task time.

Acknowledgments

This work was supported by EPSRC grant EP/G043507/1. The study reported here was undertaken as part of the second author's Master's dissertation work. We thank Dario Salvucci for providing the code for the driving simulator used here. We are grateful to three anonymous reviewers for their valuable comments on this work.

References

Brumby, D.P., Howes, A., & Salvucci, D.D. (2007). A cognitive constraint model of dual-task trade-offs in a highly dynamic driving task. In *Proc. SIGCHI Conference on Human Factors in Computing Systems, CHI'07* (pp. 233-242). New York, NY: ACM Press.

Brumby, D.P., Salvucci, D.D., & Howes, A. (2009). Focus on driving: How cognitive constraints shape the adaptation of strategy when dialing while driving. In *Proc. SIGCHI Conference on Human Factors in Computing Systems, CHI'09* (pp. 1629-1638). New York, NY: ACM Press.

Del Rosario, N. (2009). *Reasons to Switch: The Effects of Priority and Information Presentation on Dual-Task Interleaving Strategies.* Master's dissertation, University College London, London, UK.

Gopher, D., Brickner, M., & Navon, D. (1982). Different difficulty manipulations interact differently with task emphasis: Evidence for multiple resources. *Journal of Experimental Psychology: Human, Perception, & Performance, 8,* 146–157.

Gopher, D. (1993). The skill of attention control: Acquisition and execution of attention strategies. In D.E. Meyer & S. Kornblum (Eds.), *Attention and Performance XIV: Synergies in Experimental Psychology, Artificial Intelligence, and Cognitive Neuroscience* (pp. 299–322). Cambridge: MA: MIT Press.

Gray, W.D., Sims, C.R., Fu, W.-T., & Schoelles, M.J. (2006). The soft constraints hypothesis: A rational analysis approach to resource allocation for interactive behavior. *Psychological Review, 113*, 461-482.

Horrey, W.J., Wickens, C.D., & Consalus, K.P. (2006). Modeling drivers' visual attention allocation while interacting with in-vehicle technologies. *Journal of Experimental Psychology: Applied, 12,* 67-78.

Navon, D., & Gopher, D. (1979). On the economy of the human-processing system. *Psychological Review, 86,* 214-255.

Norman, D.A., & Bobrow, D.G. (1975). On data-limited and resource-limited processes. *Cognitive Psychology, 7,* 44-64.

Salvucci, D.D. (2005). A multitasking general executive for compound continuous tasks. *Cognitive Science, 29*, 457-492.

Salvucci, D.D., & Taatgen, N.A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review, 115*, 101-130.

Wickens, C. D. (2002). Multiple resources and performance prediction. *Theoretical Issues in Ergonomics Science, 3,* 159-177.

Wang, D.D., Proctor, R.W., & Pick, D.F. (2007). Acquisition and transfer of attention allocation strategies in a multiple-task world environment. *Human Factors, 49*, 995–1004.

Nomination and Prioritization of Goals in a Cognitive Architecture

Dongkyu Choi
Department of Aeronautics and Astronautics
Stanford University, Stanford, CA 94305
dongkyuc@stanford.edu

Abstract

Goals play an important role in human cognition. Different aspects of human mind influence the generation of goals they pursue, and the goals guide their behavior. In psychology, researchers made significant efforts to study goals and their origin, and cognitive architectures include various facilities to handle goals of artificial agents. One such architecture, ICARUS, supports goal-driven behaviors while maintaining reactivity, and the top-level goals play an important role by guiding the behavior of ICARUS agents. However, the architecture does not cover the origin of its goals or the management of them, and this imposes restrictions like limited autonomy in ICARUS. In this paper, we extend the architecture to provide the capability to manage top-level goals using the notion of long-term, general goals. We show some illustrative examples in an urban driving domain, and discuss related and future work in this direction.

Introduction and Motivation

Goals play an important role in human cognition. People have ideas on what they want to do or what they should do, and these give rise to many different goals. Such goals, in turn, guide people's behavior by restricting the space of possible actions to take. Traditionally, psychologists put significant efforts on the study of this process (Simon, 1967; Sloman, 1987; Gray & Braver, 2002 to name a few). As computational frameworks for models of cognition, most cognitive architectures (Newell, 1990), too, have some supports for goals. At the very least, these architectures allow the specification of goals or subgoals that guide the artificial agent's behavior. But some architectures provide more, including nomination and prioritization of goals. For instance, CLARION (Sun, 2007) has drive and goal mechanisms that correspond to psychological accounts of goal nomination. In Soar (Laird et al., 1986), the top-level operators can act as reactive goals and there are rules that govern their nomination as goals.

Another cognitive architecture, ICARUS (Langley & Choi, 2006), operates in a goal-directed fashion, and uses multiple top-level goals. However, the architecture lacks any mechanism to add, delete, or reorder such goals, limiting its capabilities significantly. In this paper, we present the ICARUS architecture with a new goal management mechanism that is reactive to the environment. We extended the existing architectural distinction between long-term knowledge and short-term structures to goals by introducing general goal descriptions associated with their own relevance conditions. The system instantiates these goals with respect to the current situation of the world and nominates them as its own top-level goals to guide its behavior. The extended architecture also has a new ability to prioritize its nominated top-level goals by modulating their priority values with continuous degrees of match for the relevance conditions.

In the subsequent sections, we briefly review the ICARUS architecture and explain the extension for nomination and prioritization of goals in detail. Then we provide some illustrative examples in an urban driving domain. After that, we conclude after a discussion on related and future work.

Review of the ICARUS Architecture

ICARUS shares its basic features with other cognitive architectures like Soar (Laird et al., 1986) and ACT-R (Anderson, 1993). It makes commitments to its representation of knowledge, memory structures, and mechanisms for inference, execution, and learning. The system provides a computational framework for intelligent agents, which stays constant across different domains. In this section, we review the basic capabilities of the architecture before we continue our discussion on nomination and prioritization of goals. We start with ICARUS' representation of knowledge and memories that support this, and then cover the architecture's inference and execution processes. Throughout this section, we show examples from an urban driving domain, which we also use for demonstration purposes in a later section.

Representation and Memories

The ICARUS architecture distinguishes conceptual and procedural knowledge. Its *concepts* describe various aspects of the environment, whereas its *skills* define procedures that are known to achieve corresponding concepts when executed until completion. ICARUS also distinguishes long-term knowledge and short-term structures. Long-term knowledge includes general descriptions of the environment and procedures. The architecture instantiates them and gets short-term structures relevant to the current situation.

The distinctions along these two directions result in four main memories in ICARUS. Its long-term conceptual memory stores general definitions of concepts that use variablized objects and their attributes to describe situations. A long-term skill memory houses variablized skills that define general procedures to achieve certain concepts, namely their goals. When the system instantiates these general concepts and skills, it deposits them in the corresponding short-term memories. A short-term conceptual memory stores instantiated concepts, which the system believes to be true in the current situation. A short-term skill memory holds instantiated skills, along with their corresponding goals. For this reason, we often call the short-term memories as the *belief memory* and the *goal memory*, respectively.

Table 1 shows some sample concepts in an urban driving domain. The first two concepts are *primitive*, and they include only perceptual matching conditions that ground on object information from the environment in the `:percepts` and `:tests` fields. On the other hand, the last concept is *non-primitive*, since it refers to other concepts in the `:relations` field. This hierarchical organization of concepts allows multiple levels of abstraction, and facilitates the description of complex situations in the world. Meanwhile, Table 2 provides some examples of skills in this domain. In a similar fashion to their conceptual counterparts, there are primitive and non-primitive skills. The first skill shown is primitive, and it consists of perceptual matching conditions, preconditions, and a direct reference to an immediate action in the world. The other two skills, however, are non-primitive, and they provide subgoal decompositions instead of references to actions. In the next section, we cover ICARUS' processes that work over these knowledge structures stored in its memories.

Table 1: Some sample ICARUS concepts for the urban driving domain.

```
((yellow-line ?line)
 :percepts  ((lane-line ?line color YELLOW)))

((at-turning-speed ?self)
 :percepts  ((self ?self speed ?speed))
 :tests     ((>= ?speed 15)
             (<= ?speed 20)))

((ready-for-right-turn ?self)
 :relations ((in-rightmost-lane ?self ?l1 ?l2)
             (at-turning-speed ?self)))
```

Table 2: Some sample ICARUS skills for the urban driving domain.

```
((in-intersection-for-rt ?self ?int ?c ?tg)
 :percepts ((self ?self)
            (street ?c)
            (street ?tg)
            (intersection ?int))
 :start    ((on-street ?self ?c)
            (ready-for-right-turn ?self))
 :actions  ((*cruise)))

((on-street ?self ?tg)
 :percepts ((self ?self)
            (street ?st)
            (street ?tg)
            (intersection ?int))
 :start    ((intersection-ahead ?self ?int ?tg))
 :subgoals ((ready-for-right-turn ?self)
            (in-intersection-for-rt ?self ?int ?st ?tg)
            (on-street ?self ?tg)))

((ready-for-right-turn ?self)
 :percepts ((self ?self))
 :subgoals ((in-rightmost-lane ?self ?l1 ?l2)
            (at-turning-speed ?self)))
```

Inference and Execution

The ICARUS architecture operates in distinct cycles. On each cycle, the system invokes a series of processes including the inference of the current belief state and the execution of skill paths relevant to the situation. ICARUS receives sensory data from the environment at the beginning of each cycle. Based on the perceptual information, the system infers its belief state, namely, all the concept instances that are true in the current state. It starts with primitive concepts at the lowest level and moves up the hierarchy to non-primitive ones. ICARUS performs this process on every cycle, and therefore, any naive approach to the belief inference is susceptible to the combinatorial effect found in domains with many objects. In response, there have been several efforts to alleviate this problem including Asgharbeygi et al. (2005).

When the system finishes inferring its belief state, it attempts to execute its skills accordingly. ICARUS retrieves skills that are relevant to its top-level goals, and finds one or more executable paths through the hierarchy that start from these skills. A skill path is executable when all the skill instances on it are executable, from top to bottom. Although a path might include a single primitive skill that achieves an ICARUS agent's top-level goal, a skill path usually starts with a non-primitive skill for a top-level goal and continues down several levels until it reaches a primitive skill at the bottom. The primitive skill includes some actions the system needs to perform in the environment. The ICARUS architecture takes these actions and applies them to make changes in its surroundings. Then the system repeats the processes based on the updated sensory data. In the following section, we continue our discussion on the architecture in the context of the new extension.

Reactive Goal Management

As seen in the previous section, the ICARUS architecture has a goal memory that stores information on its top-level goals and subgoals along with their corresponding skill instances. Most contents of the memory are very specific and short-lived, and they change as the agent moves along its path toward achieving its goals. But the top-level goals themselves did not change in this memory. It was as if a godly entity gave the agent a set of goals it should always pursue, which does not change over time.

This, however, is not very reasonable. When people are pursuing some goals of their own, they do get distracted from the environment, and sometimes more urgent matters come up and they should deal with them first. To support this kind of behavior, the top-level goals change dynamically in the extended architecture, rather than staying constant throughout the course of execution. The system has a new goal nomination process that generates top-level goals for its agent on each cycle. The nominated goals from this process are based on the generalized descriptions stored in a new long-term goal memory. In this new memory, we can program both general and domain-specific rules for the nomination of goals. These rules collectively represent a basic form of motivational structure in ICARUS.

Once the architecture finishes nominating goals that are relevant to the current situation, it prioritizes them before start executing for the goals. The programmer assigns a default priority value to each general goal, and ICARUS modulates this value based on a continuous measure for relevancy of the

goal. The architecture computes the degree of match for the relevance conditions of a goal whenever possible, and uses this continuous matching value during the goal prioritization process. This continuous degree of match represents the degree of relevance for the goal in the current situation, and anything less than the complete relevance will reduce the priority value accordingly. In the subsequent sections, we explain the new representation and processes in detail.

Representation

Perhaps the best way to describe the new representation is through examples. Table 3 shows some sample goals stored in the long-term goal memory. Each element takes the form of a <*conditions, goal*> pair that specifies the generalized goal and the conditions under which it is relevant. The relevance conditions stored in `:nominate` fields are templates for concepts that the system can match against its beliefs, and the goals are concepts that use some common variables that appear in the relevance conditions. The relation between this long-term goal memory and the existing short-term goal memory is similar to those between long-term concept and skill memories and their respective counterparts. This is a feature that has an architectural significance, which shows the unified nature of ICARUS.

Table 3: Some sample <*relevance conditions, generalized goal*> pairs stored in ICARUS' long-term goal memory.

```
((stopped-and-clear ME ?ped)
    :nominate ((pedestrian-ahead ME ?ped))
    :priority 10)
((clear ME ?car)
    :nominate ((vehicle-ahead ME ?car))
    :priority 5)
((cruising-in-lane ME ?line1 ?line2)
    :nominate nil
    :priority 1)
```

The elements of ICARUS' long-term goal memory also have priority values associated with them, which represent the relative importance of the goals compared to others in the memory. Users predefine the goals and their associated priority values, providing a default prioritization measure. This corresponds to the general idea people seem to have on what is more important and what is less so. For instance, most people agree that saving one's life has priority over saving his or her possessions. Many people will also save a child before saving an adult if caught in an accident. There are many examples like these, and we consider the default priorities assigned to generalized goals in ICARUS' long-term goal memory as representing this behavior. Next, we continue our discussion on the new processes that use this memory.

Nomination Process

When the ICARUS architecture finds a match for any relevance condition stored in its long-term goal memory, it instantiates the corresponding goal accordingly. The system then stores the instantiated goal in its short-term goal memory. When this nomination process is complete, the system has a series of top-level goals, which guide the behavior during the particular cycle.

The nomination process starts after the architecture infers its belief state based on the perceptual information from the environment. The system goes through each <*relevance condition, generalized goal*> pair stored in the long-term goal memory, and makes attempts to match the relevance conditions against the current state. Whenever its attempt is successful, ICARUS instantiates the corresponding goal with the variable bindings it has found from the match. This also means that the retraction of goals happens without any additional mechanisms. If a currently nominated goal loses its relevance in the subsequent cycles, the system no longer nominates the goal, effectively retracting it from the short-term goal memory. During this retraction, however, ICARUS stores some information on the previous nomination, and uses it at a later time if the same goal instance is nominated again.

Figure 1 shows a simple situation that involves the nomination and retraction of a goal. Initially, there is nothing in front of the agent's car (shown as a green box) moving upwards in the figure. Therefore, it has a single goal to get to its target location. Then a pedestrian, *ped1* (shown as a yellow smily face), suddenly starts to jaywalk the street in front of the agent's car and this causes a concept instance, *(pedestrian-ahead me ped1)*, to match in the state. In response, the system generates the corresponding goal, *(stopped me)*, and now it has two goals as shown in the second column. When the pedestrian moves away, the relevance condition disappears and the goal is retracted. The agent has a single goal again, as shown in the last column.

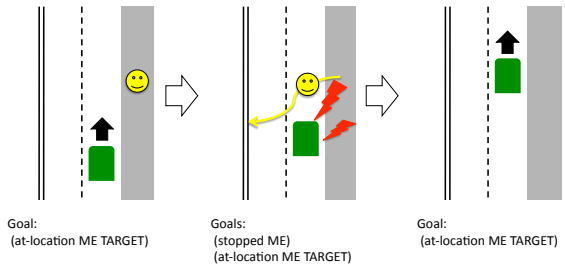


Figure 1: An example of goal nomination process in an urban driving domain.

Prioritization Process

Once ICARUS completes the nomination process, it attempts to reorder the currently nominated goals to prioritize them under the given circumstances. Since all the top-level goals have default priority values associated with them and ICARUS orders the goals according to these values, we need a mechanism to modulate these fixed values based on the current situation of the world. This modulation will then give goals with lower default priorities a chance to overtake higher-priority

ones. Our approach uses the continuous matching of concepts, more specifically, the relevance conditions associated with each goal.

As shown in the previous section, ICARUS' concepts include perceptual matching conditions. Especially, some primitive concepts have numeric tests in their bodies that often involve continuous variables. We take such variables as the source of continuous matching. For example, consider a concept that includes a numeric test on a variable, *?var*, as in $0 <?var < 10$. ICARUS normally checks if the value of the variable is within the specified range, and returns true (1) if it is larger than 0 and smaller than 10, but returns false (0) otherwise. But if we make the boundaries of the tests smoother as shown in Figure 2, we can get some partial matches between 0 and 1 when the variable falls right outside of the specified region.

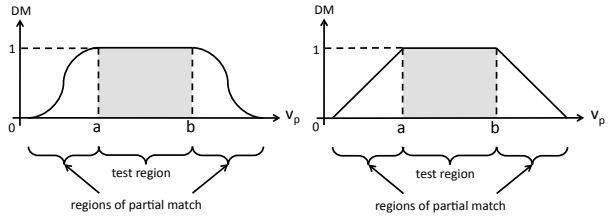


Figure 2: Curves applied to the boundaries of numeric tests for continuous matching.

When the relevance conditions associated with ICARUS' goals include a primitive concept, we can get the degree of match between zero and one using this mechanism. This value will then represent how relevant the associated goal is, and we can use it to modulate the default priority value of the goal. In this manner, a very relevant goal with a low default priority can overtake a barely relevant goal with a high default priority. We believe this explains people's behavior in extreme conditions like when people are extremely hungry or thirsty. In such cases, people will probably drink fluids with a bad smell that they would normally reject.

Illustrative Examples

With the extensions described so far, we believe the ICARUS architecture provides a reasonable account of goal management. Testing this hypothesis, however, is not of the standard affair. As is often the case in the evaluation of cognitive architectures, capabilities like the goal management are innately at a very high-level. We want to show performance improvements we can get from the extended system over the previous one, but doing so using several quantitative measures is not immediately possible in this case, and those results will not be quite representative either. Instead, we can demonstrate the qualitative behavior of the extended system and confirm that it is far more aligned with our intuition about human cognition than the previous system. Cassimatis et al. (2008) suggested that models of higher-order cognition should be evaluated in three aspects: their ability compared to humans, the breadth of situations they cover, and the parsimony of their mechanisms.

In this section, we challenge the original and the extended systems with two scenarios. By comparing the two systems, we show the advantages of the goal management in various aspects like programmability and human-like behavior. Often the original system is not capable of demonstrating the desired behavior at all, while the extended system can easily simulate it.

Scenario 1: Cruiser

Imagine that you are driving a sports car cruising down the street. You notice a car slowing down and stopping in front of you, and you swerve around the car by changing your lane. After a while, a group of careless pedestrians jump out to the road all of a sudden and jaywalk the street. Startled, but decisively you make a move to avoid hitting the pedestrians and continue your cruise down the road. Unless you are driving exclusively on freeways, this kind of situation should sound very familiar.

In the previous version of the ICARUS architecture, we would program this behavior by giving the system two goals, *(stopped-and-all-clear me)* and *(cruising-in-lane me ?line1 ?line2)* in this order. The system gives higher priority to the first goal than the second one, so it correctly focuses its attention to maintaining a safe distance from pedestrians before worrying about cruising on the street. However, we find several issues with this program. In addition to the fact that the system will have the first goal regardless of whether it is relevant or not, a more notable problem is that the first goal does not mention any specific pedestrian, and that the system will need to pick a pedestrian dynamically within the skills for this goal. This means that the system can cover for only one pedestrian at a time. We will probably program it so that the closest pedestrian from the ICARUS agent's position gets the attention, but no matter what we do, the system has no way to consider any other pedestrians.

On the other hand, using the extended system with the goal nomination capability, we would program three long-term goals like, *(stopped-and-clear me ?ped)* with the nomination condition *(pedestrian-ahead me ?ped)*, *(clear me ?car)* with the nomination condition *(vehicle-ahead me ?car)*, and *(cruising-in-lane me ?line1 ?line2)* with a null nomination condition. Table 4 shows ICARUS concepts and skills for the extended system that we wrote this way. The first advantage of this system over the previous one is that the agent has only the relevant set of goals at any given moment, much like people would. But what is more important in this particular case is that, the ICARUS agent can consider each instance of the goals separately. For instance, if there are multiple pedestrians jaywalking the street in front of the agent's car, multiple instances of the generalized goal, *(stopped-and-clear me ?ped)* will be deposited into the system's short-term goal memory, and the system will be able to consider all of them in the order of their corresponding priorities. By doing so, the system can take an action for the highest priority goal

and continue to the subsequent ones if resources are available. It is also notable that the system no longer requires a complicated goal concept. Instead, all the individual cases of different pedestrians are instantiated from a generalized goal description, and deposited into the system's short-term goal memory.

Table 4: ICARUS concepts and skills for the Cruiser scenario using the extended architecture.

```
((stopped-and-clear ?self ?obj)
 :percepts  ((self ?self))
 :relations ((stopped ?self)
             (clear ?self ?obj)))

((clear ?self ?obj)
 :percepts  ((self ?self)
             (pedestrian ?obj))
 :relations ((not (pedestrian-ahead ?self ?obj))))

((clear ?self ?obj)
 :percepts  ((self ?self)
             (car ?obj))
 :relations ((not (vehicle-ahead ?self ?obj))))
```

```
((stopped-and-clear ?self ?obj)
 :percepts ((self ?self))
 :actions  ((*brake 1000)))

((clear ?self ?obj)
 :percepts ((self ?self))
 :start     ((in-leftmost-lane ?self ?line1 ?line2))
 :subgoals ((in-rightmost-lane ?self ?line3 ?line4)))

((clear ?self ?obj)
 :percepts ((self ?self))
 :start     ((in-rightmost-lane ?self ?line1 ?line2))
 :subgoals ((in-leftmost-lane ?self ?line3 ?line4)))
```

Let us analyze a typical run with this system. The agent starts in the leftmost lane of a street segment. There are several other cars in that stretch of the street, and the first one, *c6120* is far ahead of the agent in the same lane. For the first 10 cycles, the agent has a single goal, *(cruising-in-lane me ?line1 ?line2)* that is always nominated. On cycle 11, as the ICARUS agent gets closer to the car, *c6120*, it detects that the car is blocking its way and the predicate, *(vehicle-ahead me c6120)*, becomes true in the state. So, the system nominates *(clear me c6120)* as its goal. On the next cycle, ICARUS retrieves a skill for the first goal with the same name, *clear*, and the skill leads to an action, *(*steer 35)*. While the agent is changing its lane to the right, it notices on cycle 13 that its speed is below the predefined cruising speed, and the second goal *cruising-in-lane* is unsatisfied. The agent now executes *(*gas 20)* concurrently with *(*steer 35)* to adjust its speed. It continues steering to the right while it performs the speed adjustments as needed until cycle 20, but then it notices that it is in the target lane, and starts aligning itself in that lane. By this time, the agent successfully avoided the blocking vehicle, and the concept instance, *(vehicle-ahead me c6120)*, is no longer true. So the goal, *(clear me c6120)*, that was triggered by this concept instance disappears.

Scenario 2: Ambulance

Now, to make the task more complicated, let us think about driving an emergency vehicle, say, an ambulance. We sometimes see that an ambulance is moving quite normally, waiting for pedestrians to pass, observing the speed limit, and even stopping for red lights, although it has its lights and siren on. Yet some other times we see an ambulance speeding by almost like one driven by a reckless driver, blinking every single light it has equipped on and making a very loud sound. We can guess that the difference is on the severity of the problem at their destinations, or onboard, and this factor affects the behavior of the drivers.

Modeling this behavior in the previous version of ICARUS is close to impossible, unless the programmer is patient enough to write concepts and skills for all possible cases there are. Even then, the space of search will be so large that the performance will be below what is required during the execution. However, the extended system supports this behavior easily, with some generalized goals encoded in its long-term memory, coupled with their corresponding triggers. Table 5 shows the new concepts that we added for this scenario.

Table 5: ICARUS concepts and skills for the Ambulance scenario using the extended architecture.

```
((emergency ?self)
 :percepts ((self ?self status ?status level ?level))
 :tests ((equal ?status 'emergency)
         (= ?level 10))
 :pivot (?value))

((not-emergency ?self)
 :percepts  ((self ?self))
 :relations ((not (emergency ?self))))
```

To handle the task to get to the hospital with the proper urgency based on the current situation, we encode the goal, *(okay-to-go ME ?signal)* with priority 2, to have nomination conditions, *(signal-ahead me ?signal)* and *(not-emergency me)*. This goal is what forces the agent to observe traffic signals when there is no emergency. But when the emergency strikes and the degree of match for the concept *(emergency me)* starts to increase from zero, that for the concept *(non-emergency me)* starts to decrease from one accordingly. When this happens, the relevance of the above goal drops with them, eventually making the architecture focus on the other goal of getting to the hospital first.

Now we will show how the system behaves during a typical run. In a similar fashion as before, the agent starts out by accelerating itself to reach its cruising speed. On cycle 7, it finds a car blocking its path, and starts steering to the right to clear the car. With occasional accelerations to maintain its speed, it continues steering to the right. On cycle 13, it notices that it is in the target lane, and starts to cruise there. But it soon finds another car, and clear it in a similar manner, but this time to the left lane, and finishes the move by cycle 21. The agent then sees a traffic signal that is red, and brakes to stop. During the wait, its emergency level changes to 8, which, in turn, changes the degree of match for the concept instance, *(emergency me)* to 0.8. The negation of this instance, *(not-emergency me)*, therefore, gets its degree of match at 0.2. This is a nomination condition for one of the current goals, *(okay-to-go me c27224)*. Hence the system modulates the priority value of the goal to be 0.4 ($= 2 \times 0.2$). This causes the goal to be less important than the default goal, *(cruising-in-lane me ?line1 ?line2)* that has the priority of 1.

Therefore, the system now stops observing traffic signals, and starts cruising even with the red traffic light. Later on cycle 95 when it reaches the next intersection, however, the emergency level is back to 3, and the modulated priority value for *(not-emergency me)* becomes 0.7. This once again puts the goal to observe traffic signals before the default goal of cruising, and the system starts observing signals again.

The two programs shown above, one for the original architecture and the other for the extended architecture, both result in equivalent behaviors at the high level. However, the two systems still have differences at lower level for basic driving maneuvers, and the extended system shows much smoother driving behavior. What is important to note in this scenario is that the goal nomination capability leads to a much simpler program that is more intuitive and reasonable to us.

Related and Future Work

Our work has been heavily influenced by related work in the psychology literature. One can find a fair amount of research related to motivation and goal selection there. Typically, these also cover the topic of emotion. Simon (1967) recognized that the central nervous system, despite being a serial information processor, serves multiple needs in an organism surrounded by unpredictable situations. He suggested that two mechanisms, a goal-terminating mechanism and an interruption mechanism, would satisfy this requirements. Simon further described the relationship among interruption, motivation, and emotion, and outlined an information-processing system that covers these as wells as learning in relation to them. More recently, Sloman (1987; 2002) suggested that any system with priority in beliefs and actions naturally have emotions. He argued that goals often conflict with each other, and systems must have a mechanism to resolve such conflicts. The author proposed that motivators can serve this purpose.

As mentioned earlier in this paper, there are also some related work in the architectural perspective. CLARION (Sun, 2007) and Soar (Laird et al., 1986) architectures possess their own accounts of goal management. The former is more psychologically positioned, providing interactions between drives and goals. The latter has a rule-based mechanism to nominate its top-level operators as its goals, which resembles the conditionalized goals ICARUS has. Unlike ICARUS, however, the Soar architecture proposes a single goal at a time, removing the need for prioritization or the advantage of interactions among multiple goals.

Although the current work is an important first step toward a cognitive architecture with the full capability for goal management, it still ignores a vast amount of psychological accounts on human motivation and goal handling. First of all, people can change priorities among different goal in a flexible manner, depending on the current situation. We have a way to model this behavior, and hope to report in this direction in a near future. More broadly, we should explain where the long-term knowledge about goals comes from. It is very likely that we will deal with even higher-level cognitions like motivations, emotion, and obligations. We expect the the evidences in the social psychology literature will help us in the modeling process.

Conclusions

In this paper, we introduced an extension to the ICARUS architecture for reactive goal management. We first conceived the idea in the architectural perspective, but the extension makes close connections to previous work in psychology and other related fields. The extended framework supports the nomination, retraction, and prioritization of goals based on the current belief state. We have demonstrated in an urban driving domain that the extension leads to simpler programs while supporting new behaviors that connects to the context better than the original architecture.

References

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum.

Asgharbeygi, N., Nejati, N., Langley, P., & Arai, S. (2005). Guiding inference through relational reinforcement learning. In *Proceedings of the fifteenth international conference on inductive logic programming* (pp. 20–37). Bonn, Germany: Springer Verlag.

Cassimatis, N. L., Bello, P., & Langley, P. (2008). Ability, breadth, and parsimony in computational models of higher-order cognition. *Cognitive Science*, *32*, 1304–1322.

Gray, J. R., & Braver, T. S. (2002). Integration of emotion and cognitive control: A neurocomputational hypothesis of dynamic goal regulation. In S. C. Moore & M. Oaksford (Eds.), *Emotional cognition: From brain to behaviour* (pp. 289–316). Philadelphia, PA: John Benjamins Publishing Company.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in soar: The anatomy of a general learning mechanism. *Machine Learning*, *1*, 11–46.

Langley, P., & Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings of the twenty-first national conference on artificial intelligence.* Boston: AAAI Press.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Simon, H. A. (1967). Motivational and emotional controls of cognition. *Psychological Review*, *74*(1), 29–39.

Sloman, A. (1987). Motives, mechanisms, and emotions. *Cognition & Emotion*, *1*(3), 217–233.

Sloman, A. (2002). How many separately evolved emotional beasties live within us? In R. Trappl, P. Petta, & S. Payr (Eds.), *Emotions in humans and artifacts* (pp. 35–114). MIT Press.

Sun, R. (2007). The motivational and metacognitive control in CLARION. In W. Gray (Ed.), *Modeling integrated cognitive systems.* New York, NY: Oxford University Press.

Modelling the Correlation Between Two Putative Inhibition Tasks: A Simulation Approach

Richard P. Cooper (R.Cooper@bbk.ac.uk) and Eddy J. Davelaar (E.Davelaar@bbk.ac.uk)
Department of Psychological Science, Birkbeck, University of London
Malet Street, London WC1E 7HX, UK

Abstract

Behavioural studies of individual differences have shown mild but significant correlations in performance on tasks that require the withholding of a response to a prepotent stimulus, i.e., on so-called *response inhibition* tasks. Several computational models of response inhibition tasks have been developed, but the dominant models of such tasks have been produced in isolation of each other. Consequently they fail to present a coherent unitary picture of response inhibition. In this paper we consider two established interactive activation models of distinct response inhibition tasks – the stop signal task and the Stroop task – and explore potential mechanisms within those models that might underlie the observed behavioural correlation. Only one plausible account of the correlation emerges: that it results from shared mechanisms of attentional bias. This account does not map onto the classical concept of response inhibition. It is concluded that either the accepted models are flawed or that the concept of response inhibition as applied to these tasks is misleading (and hence counterproductive). More generally the work may be taken to support an architectural approach to modelling, albeit at the level of interactive activation models, rather than the more traditional production system models.

Keywords: Executive processes; cognitive control; response inhibition; individual differences; Stroop task; Stop signal task.

Introduction

The construct of "response inhibition" is frequently invoked when attempting to explain behaviours in tasks or situations that demand the withholding of a strongly prepotent response. Response inhibition is held to be a separable task-general executive or cognitive control function, the efficacy of which varies across individuals.

In the laboratory response inhibition is standardly explored in variants of the stop signal task (Logan & Cowan, 1984). This is a form of simple reaction time task in which subjects are normally required to respond as quickly and accurately as possible. However, on a small number of trials a compound stimulus is presented and on these trials and these trials only the subject is required to withhold their response. Such trials are referred to as "stop trials". Typically the compound stimulus consists of a standard stimulus that might occur on any normal trial followed almost immediately by an auditory beep. Stop trials are rare in comparison to normal "go trials". This and the need to respond on go trials as rapidly as possible ensures that the go response is prepotent. Performance is measured in terms of the number or proportion of stop trials on which a response is (incorrectly) produced. This measure varies reliably between subjects. Good response inhibitors produce few stop responses, while poor response inhibitors produce many.

There is substantial behavioural and neuroscience evidence, as well as good theoretical reasons, for supposing that response inhibition is a task-general control function. From the theoretical perspective, response inhibition fits clearly within the supervisory system/contention scheduling framework of the control of thought and action of Norman and Shallice (1986). On this influential account, a system for the control of routine or well-learned behaviours, *contention scheduling*, is modulated by a deliberative system, *the supervisory system*, when routine behaviour is inappropriate and must be overridden. Contention scheduling is appropriate for generating the prepotent response, whatever the situation. If this is not appropriate, as in stop trials of the stop signal task, the supervisory system must override contention scheduling. A plausible way for this to be operationalised is in terms of response inhibition acting as a sub-function of the supervisory system.

From a neuropsychological perspective, patients have been reported who are well-characterised in terms of a deficit in response inhibition. Thus, utilisation behaviour patients tend to exhibit behaviours that are driven largely by environmental contingencies rather than their stated intentions (Lhermitte, 1983). Alien hand patients show similar problems, but they are restricted to one hand (Goldberg et al., 1981). Both deficits may be seen as arising from a failure in response inhibition.

One source of behavioural evidence for the task-general nature of response inhibition comes from a large individual differences study of Miyake et al. (2000). In this study, 137 subjects were each tested on a total of 14 tasks. Performance on 3 of these tasks was argued, on a priori grounds, to specifically require response inhibition. Subsequent factor analysis of subject performance across the tasks supported this view, with performance on the response inhibition tasks being related to a single factor that differentiated those tasks from others in the study, which were held to primarily tap other executive functions (namely the functions of *set-shifting* and *memory monitoring and updating*).

The three response inhibition tasks of Miyake et al. (2000) were a) a forced-choice decision variant of the stop signal task, b) the Stroop colour naming task, and c) an anti-saccade task. Our focus in this paper is on the first two, and so we described these in detail. In the stop signal task, subjects were required to indicate with a button press whether a (visually presented) word was an animal or a non-animal. The first block of 48 trials were all "go" trials.

These were used to establish a mean response time for each subject. One quarter of the trials in the second block (of 192 trials) were stop trials. In these trials, a beep was sounded shortly after presentation of the word (225ms prior to the subject's mean response time, as determined in block 1), and subjects were required to withhold their response. The dependent measure was the proportion of stop trials on which a response was given. In the well-known Stroop colour naming task, subjects were presented with a "word" written in one of six colours. They were required to name the colour of the stimulus word. On neutral trials the word was a string of asterisk symbols, while on incongruent trials it was the name of another colour. The dependent variable was the difference in mean response times for incongruent and neutral trials.

For our purposes, the critical result of this individual differences study was mild but significant positive correlations ($r \approx 0.20$) between performance on the stop signal task and the Stroop task (and in fact between all pairs of response inhibition tasks). In general, these correlations were stronger than those between any single response inhibition task and any of the non-response inhibition tasks explored in the study. However, while the study is impressive in its scale, interpretation of the results is limited because Miyake et al. fail to provide process accounts of the various tasks. While it is perhaps unreasonable to expect such models of all 14 tasks, the absence of process models leaves unexplained the mechanism that is, on the account proposed by Miyake and colleagues, shared by the response inhibition tasks. Similarly, it leaves open the issue of why that function is *not* significantly involved in successful performance of the other tasks considered in the study.

The purpose of the work presented here is to begin to address this limitation by exploring potential common mechanisms within established process models of two of Miyake et al.'s response inhibition tasks. We focus on models of the stop signal task and the Stroop task because there are established models of each task (due to Boucher et al., 2007, and Cohen & Huston, 1994, respectively) that bear some correspondence. This correspondence offers the possibility of relating the models to each other and thereby identifying a shared response inhibition mechanism. For such a mechanism to be explanatorily adequate, it must be parameterisable, with the observed behavioural correlations between tasks arising in part from variation in a shared parameter. To foreshadow, simulation findings derived from reimplementations of the existing published models suggest that directly shared parameters fail to yield the required correlation in performance. However, an appropriate correlation is forthcoming if attentional biasing mechanisms are yoked. Unfortunately, attentional biasing is not normally related conceptually to response inhibition. We conclude that either a) response inhibition is not the mechanism underlying the behavioural correlation in these tasks, or b) one or both of the accepted models requires updating. These simulation results extend those of a complementary analytic study (Davelaar & Cooper, 2010).

The Stop Signal Task

The Model

Early work with the stop signal task demonstrated that behaviour on the task could be well accounted for by a race model consisting of two stochastic processes, a "go" process which is slow to activate but has a head start, and a "stop" process which is faster to activate but starts late (Logan & Cowan, 1984). Successful performance on a stop trial requires that the stop process reach threshold before the go process. Boucher et al. (2007) note that despite this model's strengths, it is inconsistent with neural evidence of interaction between stop and go processes. They present the *interactive race model*, an update of the original model in which the stop and go processes compete through mutual lateral inhibition. The model, as applied to Miyake et al.'s semantic categorisation variant of the stop signal task, is shown in Figure 1.

The model is extremely simple, consisting of just three nodes: one for each response and one for the stop process. Processing in the model is cyclic with each node operating as a leaky competing accumulator (Usher & McClelland, 2001). On each cycle, the activation of a node is increased by an amount proportional to its external input, less an amount proportional to the activation of its competitors (corresponding to lateral inhibition), less an amount proportional to its current activation (its leakage), plus normally distributed random noise. Parameters control the contributions of the various sources to this accumulation. For default behaviour we assume ballpark parameters scaled from those of Boucher et al. to give a response threshold of 1.0. Thus, we assume lateral inhibition, β, of 0.025 between all pairs of nodes, leakage of 0.0 (i.e., the accumulators do not leak), and the standard deviation of noise, σ, of 0.025 units per cycle.

In addition, it is assumed that on any trial external input to one of the response nodes (*animal* or *non-animal*) is provided by a semantic categorisation process (which is not modelled). The level of input is controlled by the parameter μ_{go}, set to 0.005 units per cycle by default. It is assumed that the other response node receives zero external input. On stop trials it is assumed that at some point during the trial

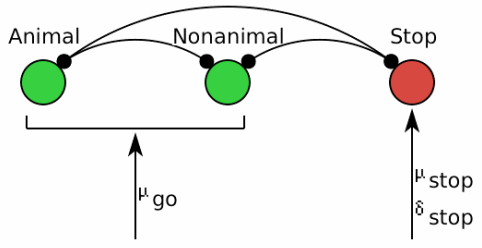


Figure 1: The interactive race model of the stop signal task. On any one trial, either the animal or the nonanimal node receives activation from a semantic categorisation process. On "stop" trials, the stop node also receives activation, though this activation is delayed relative to the activation from the semantic categorisation process.

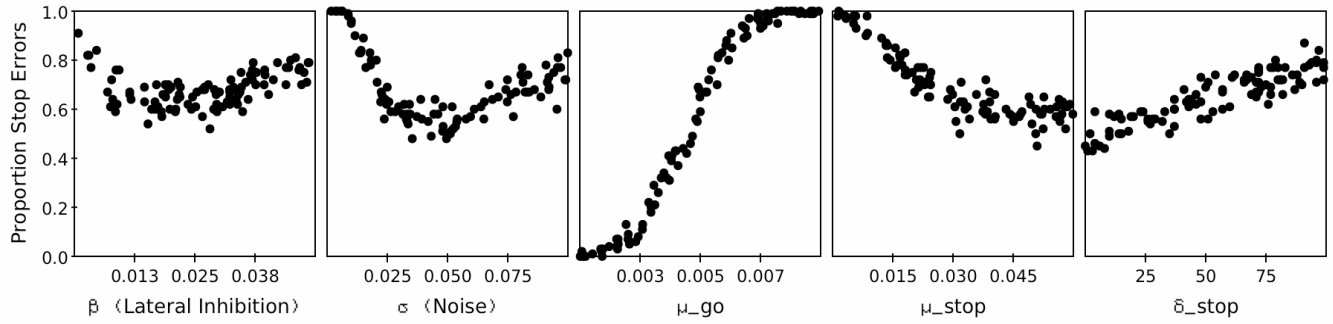


Figure 2: Effects of varying key parameters on the proportion of stop errors produced by the interactive race model of the stop signal task.

external input is provided to the stop node. The level of this input is μ_{stop}, set to 0.030 units per cycle by default. Finally, we assume that the delay between input to the response nodes and input to the stop node is 250 cycles. This delay is the sum of the actual delay between presentation of word and stop stimuli, *SSD*, and the time to initiate the stop process, δ_{stop}. With these parameters, the model performs as desired – on go trials its response accuracy is approximately 99% (with noise and lateral inhibition occasionally leading to error) while on stop trials it fails to stop on approximately 65% of occasions. This compares well with mean subject performance of 67% as reported by Miyake et al. (2000).

Simulation Results

An initial set of simulation studies was performed to determine the relation between the model's performance and the key parameters that could reasonably be argued to vary across individuals, that is: μ_{go}, μ_{stop}, β (lateral inhibition), σ (standard deviation of noise) and δ_{stop}.[1] Each parameter was varied about the default value (with the other four parameters fixed at default values) to determine the effect of that parameter on the proportion of stop errors. Figure 2 summarises the results, based on 100 blocks per parameter, each of 100 trials.

As can be seen from the figure, there is a slight non-monotonic relation between β (lateral inhibition) and the model's performance, with fewer stop errors at intermediate values. Similarly there is a non-monotonic relation between σ (noise) and stop errors. Perhaps surprisingly, when noise is very low there are more stop errors than when noise is at moderate values. This is because noise may delay the model's decision, causing it to respond more slowly on some trials (but more quickly on others). On stop trials when noise acts against the go process this gives the stop process more time to affect behaviour. There is an optimal value for noise, however, and if it is too high successful stopping again becomes rare. Increasing μ_{stop} also reduces stop errors, though here the relation is monotonic and the explanation is more obvious: with stronger excitation of the stop node it is more likely to reach threshold on stop trials before one of the go nodes. Stop errors correlate positively with μ_{go} and δ_{stop}. In both cases the effect of the parameter is transparent. With faster excitation of the go process or with greater delay, the stop process has less chance of reaching threshold before the relevant go process. Consequently stop errors are more likely.

Relating the results to the concept of response inhibition, it appears that good inhibitors are those who either have a) near optimal levels of lateral inhibition or noise, b) slow go processes or short stop process delays, or c) fast stop processes. Miyake et al. (2000) do not report the behavioural data that would help to discriminate between these options.

The Stroop Task

The Model

Many models have been developed of the Stroop task. We focus on the well-known model of Cohen and Huston (1994), as its principal functional mechanism, interactive activation, is shared with Boucher et al.'s interactive race model. The model, shown in Figure 3, consists of four sets of nodes, with nodes within each set competing for activation through lateral inhibition. There are two task demand nodes, three word input nodes, three colour input nodes, and two response nodes. One task demand node corresponds to the colour naming task while the other corresponds to the word reading task. The colour naming task demand node is connected to all colour input nodes, while the word reading node is connected to all word input

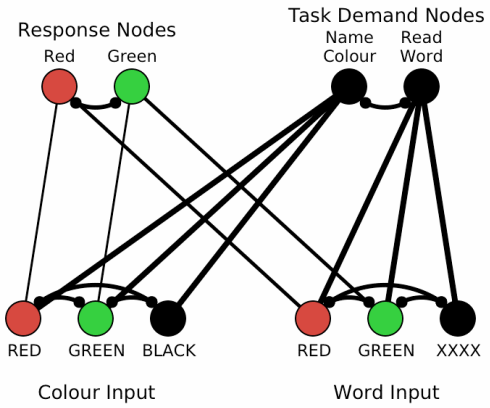


Figure 3: The Stroop model of Cohen and Huston (1994).

[1] Indeed, Boucher et al. (2007) consider how their model may be fit to data from different monkeys by varying these parameters.

nodes. Colour input nodes and word input nodes are each connected to one response node. Crucially, the connections from word inputs to response nodes are stronger than those from colour inputs to response nodes. This is justified on the grounds that word reading is the more practiced of the two tasks. As in the stop signal model, the operation of the network is cyclic with activation accumulating over time. However, the accumulation functions differ. For the Stroop model activation accumulates according to the logistic function of the time-averaged input to a node. (See Cohen & Huston, 1994, for details.)

Processing on any given trial occurs in two stages. First, input is provided to one of the task demand nodes (based on the task instructions). This causes that node to become active and the other task demand unit (through lateral inhibition) to become depressed. As a task demand unit becomes active, it excites the input nodes to which it is connected, raising the resting activation of either the colour input nodes or the word input nodes. The network settles into this temporary state, which, it is assumed, corresponds to a subject who is prepared for either a colour naming or word reading Stroop trial. Input is then provided to one colour input node and one word input node. If, for example, the trial was to name the colour of the word "RED" printed in green ink, then input would be provided to the GREEN colour node and the RED word node. In this case the colour nodes would already be moderately activated, and so the additional input to one colour node would tend to excite the appropriate response node (i.e. GREEN). At the same time, the less active word node for RED would also be receiving input and this would be tending to excite the RED response node. Hence both response nodes will receive excitation, and the balance of this excitation, plus the degree of lateral inhibition between the response nodes, will determine how quickly either response node reaches threshold.

As is clear from the architecture, there is no dedicated parameter of response inhibition. Thus, verbal descriptions of performance on the Stroop task are at odds with the computational details of the models. Nevertheless, what may be interpreted as response inhibition may well have a different label at the computational level.

Simulation Results

As in the case of the stop signal model, an initial set of simulations was performed to determine the relation between the model's performance and key parameters that could plausible be related to individual differences. Paralleling Miyake et al.'s study, the dependent variable was the difference in processing time between incongruent and neutral colour naming trials. Once again, five parameters were varied: lateral inhibition (β), the standard deviation of normally distributed noise (σ), the strength of the task demand units (μ), the gain of the activation function (γ) and the response threshold (τ). γ controls the rate at which a node's activation accumulates. It is included because Cohen and Servan-Schreiber (1992) suggest that it corresponds to an attentional modulation parameter. τ controls the sensitivity of the network to produce a response. It is fixed at 0.60 in the Cohen and Huston (1994) simulations, but we consider varying it here as it has a demonstrable affect on Stroop interference and might reasonable vary across individuals. We do not consider varying the weights from input nodes to response nodes, as these are intended to capture learned contingencies which, while possibly varying across individuals, should not vary systematically with any specific executive function.

The results of these five sets of simulations are summarised in Figure 4. The model is more complex than the stop signal model, and consequently the relations between the parameters and the relevant dependent measure – Stroop interference – are less intuitive. Nevertheless, four of the five relations are monotonic, with Stroop interference correlating negatively with β (lateral inhibition) and γ (gain), and positively with σ (noise) and τ (threshold). That is, good inhibitors correspond in the Stroop model to high lateral inhibition, low noise, optimal task demand weight, high gain or low threshold.

Yoked Simulation Studies

Recall the purpose of considering the effects of the various parameters on the performance of the two models: we are concerned with understanding the source of common variance in the tasks to which the models relate. It is hypothesised that this might be achieved by identifying a parameter that could plausibly vary across individuals and, in so doing, could underlie the observed behavioural correlation between Stroop colour naming interference and stop signal errors.

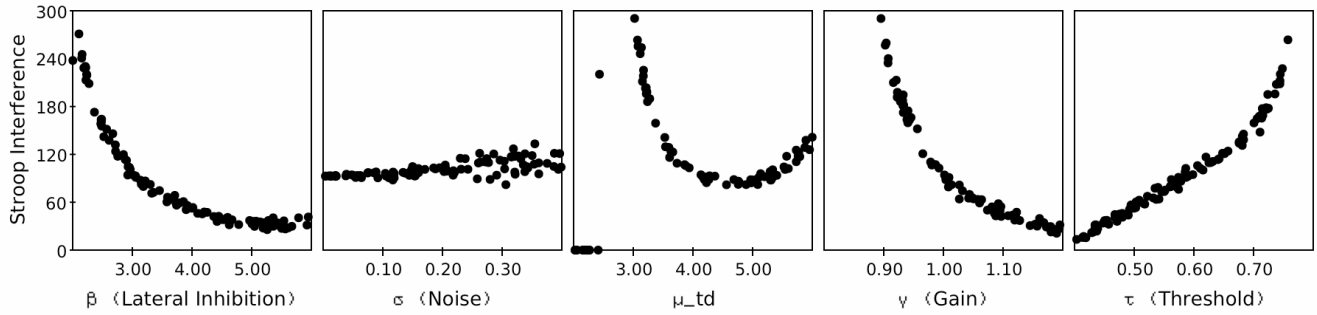


Figure 4: Effects of varying key parameters on the difference in processing time for correct incongruent and neutral colour naming trials produced by the interactive activation model of the Stroop task.

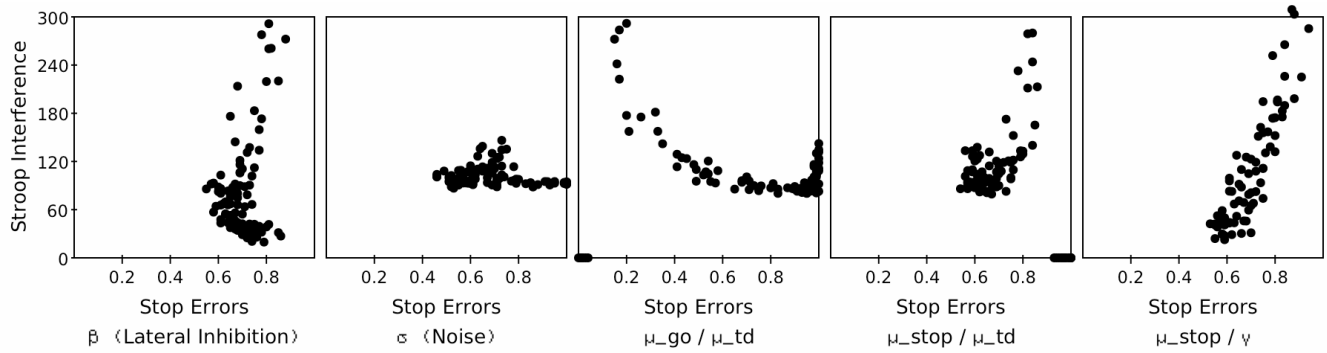


Figure 5: Effects of varying key parameters in a yoked fashion on the correlation between Stroop interference and the proportion of stop errors produced by the two models.

We are now in a position to consider candidate parameters. For example, both models share a mechanism of lateral inhibition, and pre-theoretically one could suggest that it is this mechanism, and individual differences in the shared parameter β, that underlies the behavioural correlation. The left-most panels of Figures 2 and 4 suggest that this is implausible. The issue is not the absolute size of the parameter's default value (0.025 for the stop signal model and 3.0 for the Stroop model). One can envisage re-engineering the models so that lateral inhibition in both is of a similar magnitude. The issue is that relatively high values of β lead to a reduction in Stroop interference accompanied by, if anything, a slight increase in stop errors, i.e., a negative correlation between the tasks. This is in direct contrast to the observed positive correlation.

In fact, because the relation between β and stop errors is non-monotonic, low values of β can yield a positive correlation between the tasks. This is shown in Figure 5 (left-most panel). The figure shows simulation results from 5 studies in which the value of a parameter in one model is yoked to the value of a corresponding parameter in the other model. In all 5 cases the relevant parameter values vary across the full ranges explored in Figures 2 and 4. Thus, the data in the left-most panel was generated by random sampling a dummy variable uniformly distributed between 0.0 to 1.0, and mapping the value of this onto a) the interval 0.00 to 0.05 to give a value of β for the stop signal model, and b) the interval 2.0 to 6.0 to give a yoked value of β for the Stroop model. This procedure was repeated 100 times for each of the five scatter-plots in Figure 5.[2]

From the figure we may immediately rule out several potential factors underlying the observed correlation between performance on the tasks and hence several candidates for the response inhibition function. Neither of the parameters shared by the models – lateral inhibition (β) or noise (σ) – produce correlations of the appropriate form. Hence, it would seem that individual differences in these parameters cannot underlie the observed correlations. Equally, as shown by the third plot in Figure 5, yoking the strength of the go process and the strength of task demand weights – an account not immediately related to any conceptual mechanism of response inhibition but one which, nevertheless, relates two parameters with similar functionality – fails to yield a positive correlation between the relevant dependent measures.

The desired positive correlation is shown, however, in the two right-most plots of Figure 5. Thus, the models predict that performance on the two tasks will correlate positively if a) the strength of the stop process and the strength of task demand weights are (positively) correlated, or b) the strength of the stop process and the gain in the Stroop model are (positively) correlated. There is no apriori reason to suppose the latter, but the former is plausible as both parameters concern the strength of deliberative or attentional bias. Thus, these simulation results fail to provide support for the idea that the positive behavioural correlation between Stroop interference and stop signal errors is due to a shared mechanism of response inhibition. Rather, they suggest that the correlation arises because subjects who are able to provide stronger activation to the stop process in the stop signal task are also able to provide stronger attentional bias to the colour naming task in Stroop. This suggestion is backed up by the right-most plot which shows a positive correlation resulting from yoking μ_{stop} and γ (gain). Recall that γ was also associated (positively) with attentional bias by Cohen and Servan-Schreiber (1992).

Discussion and Conclusion

In a companion paper (Davelaar & Cooper, 2010), we consider closed-form approximations to the same two models discussed here. It is demonstrated that the explanation of the behavioural correlation in terms of a shared process of response inhibition is suspect, and an attentional biasing account is proposed as a plausible alternative. The simulation results reported here corroborate both of these conclusions.

Our suggestion of attentional biasing, rather than response inhibition, as the locus of shared variability on the tasks resonates with the approach to response conflict

[2] One can envisage other approaches to yoking the parameters, e.g., by restricting attention to sub-ranges of a parameter in which its effect on the relevant dependent variable is monotonic. A further alternative focuses on the ranges of parameter values chosen. As yet there is no principled way of selecting the ranges other than through a cognitive architecture approach. Due to space limitations we do not consider these approaches here.

management of Botvinick et al. (2001). They demonstrate, within the context of three models including the Cohen and Huston Stroop model, how trial-by-trial regularities in behaviour might be accounted for in terms of a mechanism of conflict monitoring which measures the degree of conflict in the network's output nodes and modulates attentional bias, increasing it under conditions of high conflict and decreasing it under conditions of low conflict. Thus, rather than addressing response competition through response inhibition, Botvinick et al. (2001) do so through attentional biasing.

We are reluctant to fully endorse this account, however. Critically, the account is not fully consistent with the results of Miyake et al. (2000). They hold that while stop signal errors and Stroop interference are dependent upon response inhibition, they are also not dependent on two other putative executive functions – task shifting and memory monitoring and updating. Thus, if we are to account for the behavioural correlation between these tasks in terms of attentional bias, it is also necessary to show that attentional bias does *not* systematically affect behaviour on the other tasks of Miyake et al. which were held to tap these other two functions and not to tap response inhibition. Here there is reason to be cautious. Gilbert and Shallice (2002) consider performance on a task switching variant of the Stroop task in which subjects switch between colour naming and word reading. They model the critical behavioural affects by using essentially the same mechanism proposed here (i.e., by biasing task demand units) in exactly the same model (the Cohen and Huston model). Yet these are effects that, on the decomposition of Miyake and colleagues, should be explained in terms of a distinct task shifting function. Moreover in the study of Miyake et al. (2000) all correlations between putative task shifting tasks and putative response inhibition tasks were non-significant.

The concept of response inhibition held by Miyake et al. (2000) to underlie good performance in the stop signal and Stroop tasks was also held to underlie good performance in the anti-saccade task. Thus, a fuller analysis of response inhibition requires also consideration of process models of the anti-saccade task. This remains to be attempted. We would hypothesise, however, that performance in this task will also correlate with an attentional bias parameter.

Returning to the two models considered, it should also be noted that while they share principles of interactive activation, there are also major differences between them. For example, different equations govern the accumulation of activation in each model. Whether these differences are substantive or cosmetic remains to be demonstrated. However, these differences really only serve to reinforce our primary conclusion, namely, that until we have unified process models of the various putative separable executive functions, any theoretical account of their supposed unity and diversity is incomplete. By extrapolation, to understand the executive functions which underly the battery of tasks used by Miyake et al. (2000), we need to develop, within a single unified framework, models of all of those tasks. Such models must, of course, demonstrate the hypothesised shared mechanisms. Only then can we be confident that we have a plausible account of the various executive functions that contribute to the control of complex behaviour. This is, of course, one of Newell's arguments for the utility of Unified Theories of Cognition (Newell, 1990).

References

Botvinick, M.M. Braver, T.S., Barch, D.M., Carter, C.S. & Cohen, J.D. (2001). Conflict monitoring and cognitive control. *Psychological Review*, *108*, 624–652.

Boucher, L., Palmeri, T. J., Logan, G. D., & Schall, J. D. (2007). Inhibitory control in mind and brain: An interactive race model of countermanding saccades. *Psychological Review*, *114*, 376–397.

Cohen, J. D. & Huston, T. A. (1994). Progress in the use of interactive models for understanding attention and performance. In *Attention and performance 15: Conscious and nonconscious information processing*. C. Umiltà & M. Moscovitch (Eds) (pp. 453–476). Cambridge, MA, US: The MIT Press.

Cohen, J. D., & Servan-Schreiber, D. (1992). Context, cortex, and dopamine: A connectionist approach to behavior and biology in schizophrenia. *Psychological Review*, *99*, 45–77.

Davelaar, E. J., & Cooper, R. P. (2010). Modelling the correlation between two putative inhibition tasks: An analytic approach. In Catrambone, R., & Ohlsson, S. (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. Cognitive Science Society Incorporated, Portland, OR, USA.

Gilbert, S. & Shallice, T. (2002). Task switching: A PDP model. *Cognitive Psychology*, *44*, 297–337.

Goldberg, G., Mayer, N.H., & Toglia, J.U. (1981). Medial frontal cortex infarction and the alien hand sign. *Archives of Neurology*, *38*, 683–686.

Lhermitte, F. (1983). Utilisation behaviour and its relation to lesions of the frontal lobes. *Brain, 106*, 237–255.

Logan, G. D., & Cowan, W. B. (1984). On the ability to inhibit thought and action: A theory of an act of control. *Psychological Review*, *91*, 295–327.

Miyake, A., Friedman, N.P., Emerson, M.J., Witzki, A.H., Howerter, A. & Wager, T.D. (2000). The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: A latent variable analysis. *Cognitive Psychology*, *41*, 49–100.

Newell, A. (1990). *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA.

Norman, D.A. & Shallice, T. (1986). Attention to action: willed and automatic control of behaviour. In R. Davidson, G. Schwartz, and D. Shapiro (eds.) *Consciousness and Self Regulation, Volume 4*, pp. 1–18. Plenum: NY.

Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: The leaky, competing accumulator model. *Psychological Review, 108,* 550–592.

Proactive Interference in Location Learning: A New Closed-Form Approximation

Arindam Das, Wolfgang Stuerzlinger
Department of Computer Science and Engineering, York University, Toronto, Canada
{arindam, wolfgang} @cse.yorku.ca

Abstract

The ACT-R cognitive theory models forgetting in general with a constant "decay due to passage of time" parameter. However, this is not sufficient to predict learning for frequently executed tasks in dense arrangements of items. Prominent examples are two-dimensional location learning in finding keys on a keyboard or clicking on items on a web page or in a graphical user interface. Our work presents a new way to theoretically model the effect of Proactive Interference, i.e. the effect of the *history* of events on location learning, through an extension to ACT-R's mathematical model of declarative memory strength. It predicts that each time an item is searched for and found, the item gets "stronger", i.e. easier to remember. However, this strength diminishes not only through the passage of time, but also due to interference from other (non-target) items that have been encountered in the past. We tested the predictions of our new model against empirical measurements from two previous studies that involve simple visual search and selection. The predictions fit the experimental data very well.

Keywords: ACT-R declarative memory; Proactive Interference; Location Learning; User Interfaces

Introduction

Forgetting occurs not only due to passage of time but also through interference from information learned at other times (Wickens & Hollands, 2000, p. 252). Proactive interference (PI) is one explanation for this phenomenon, where some activity prior to encoding the target disrupts the retrieval of that target (Underwood, 1957; Keppel & Underwood, 1962).

Proactive Interference (PI) effects have been shown to be relevant for two-dimensional spatial memory tasks (Leung & Zhang, 2004). Spatial knowledge in two-dimensional spaces is built up primarily through interaction. That is, people remember locations after having had experience with that location (Darken and Sibert, 1996). When people are completely new to a spatial layout, such as a new grid-like arrangement of characters on a keyboard or a new arrangement of city names in a list, they will resort to visual search for the target stimulus. In the process of searching for the target, they may come across multiple non-target stimuli, i.e. irrelevant characters or city-names before they arrive at the target. These irrelevant stimuli get visually encoded during the visual search for the target. As a consequence, these non-target items, often called distractors, will interfere with the encoding of the memory for the target item.

The aim of our work is to model the effect of this PI together with the effect of the passage of time on the learning of spatially stable, two-dimensional layouts. More precisely, we limit ourselves to grid layouts in graphical user interfaces or keyboards. We choose the ACT-R cognitive theory (Anderson & Lebiere, 1998) as our mathematical modeling foundation.

The current ACT-R theory models PI through the probability of recall using a soft-max equation (Altmann & Schunn, 2002). However, previous work has established that latency to recall, i.e. reaction time, is a more sensitive indicator of proactive interference (Wixted & Rohrer, 1993, p. 1034) or interference in general (Anderson, 1983, pp. 271-272). Motivated by this fact, we modify ACT-R to generate better predictions of PI through a new model. We accomplish this as follows: 1) we replace the standard decay constant of the base-level activation equation of ACT-R theory with two terms – a constant term and a varying term. The constant term models the decline of memory strength with time, thereby preserving the standard notion of decay in ACT-R theory. The new varying term adds a function that depends on the proportion of distractor items that get visually encoded prior to encoding the target item. Thus, this newly extended model of base-level memory activation accounts for the decline of memory strength of a target item not only due to passage of time but also due to the number of distractors visually encoded while searching for the target. The result of this new activation function, later called PI activation equation, is then used by ACT-R to predict the (recognition or recall) reaction time, and therefore we generate more accurate predictions. 2) we compare the fit of *reaction time* responses, as opposed to recall probability responses, arising from the newly extended model of memory strength against empirical data from two previous studies involving visual search in two-dimensional layouts. This is a first step towards validating the new model. We choose studies involving visual search since repeated search for items leads to learning of the respective locations, and this learning process is impeded by the PI phenomenon owing to attention given to distractor items during that search.

We calculate the theoretical predictions for the empirical data as described by the equations presented in this paper through an Excel spreadsheet.

ACT-R Theory

The ACT-R cognitive theory (Anderson and Lebiere, 1998) describes a modular system that aims to replicate the human mind. It can be viewed from two perspectives: one, as a computer program that simulates the dynamic behavior of the mind; second, as a framework of mathematical equations that models the neural computations in order to realize human dynamic behavior.

Viewed from the perspective of a computer program, the ACT-R system is composed of memory, perceptual, and motor modules. The memory modules consist of a procedural memory and a declarative memory. The procedural memory is a subsystem that consists of a set of production rules and a computational engine for interpreting those rules. The production rules coordinate cognition, perception and motor actions. The declarative memory module contains chunks. Each chunk represents the memory trace of an item. A chunk can be retrieved or updated by the production rules. The activities of the memory modules together with the actions of the perceptual and motor modules enable ACT-R to simulate several dynamic aspects of the human mind.

Viewed from the perspective of a mathematical framework, ACT-R consists of independent sets of equations, each set driving the neural computation for the relevant ACT-R module. In this work, we choose to pursue this mathematical perspective. We replicate the PI effect in location learning by manipulating some of the equations embedded in the declarative memory module. We focus our upcoming discussion solely on those parts of the theory behind the declarative memory that are relevant for our objective.

ACT-R Equation of Base Level Learning

In declarative memory, chunks, i.e. memory traces of items, have different levels of activation to reflect their past use: chunks that have been used recently or chunks that are used very often receive a high activation. This activation decays over time if the chunk is not used. The activation of a chunk controls both its probability of being retrieved and its speed of retrieval. In the case where there are multiple candidates for retrieval, the chunk with the highest activation has the highest probability of being retrieved. A retrieval threshold sets the minimum activation a chunk can have and still be retrieved successfully.

The equation describing the base-level activation of a chunk *i* (representing item *i*) is given by

$$A_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \qquad \textbf{Base-Level Activation Equation}$$

where *n* is the number of practices of item *i* completed so far, t_j is the age of the *j*-th practice of the item, and *d* denotes the constant time-based decay parameter. More specifically, A_i is the strength of the memory trace of item *i* after *n* practices of that item. A *practice* of an item occurs whenever a trace of that item is presented to the declarative memory. Presentation may happen because of either recognition or recall of that item.

ACT-R Equation of Reaction Time of Declarative Memory

The time required for the declarative memory to respond to a request (recognition or recall) for an item *i* (represented by the chunk *i*) is given by the following equation:

$$T_i = Fe^{-gA_i} \qquad \textbf{Reaction Time Equation}$$

where A_i is the activation of chunk *i* and *g* is the latency exponent scale parameter. *F* is called the latency scale parameter, and maps activation to time. Traditionally, a constant term reflecting the fixed time cost of visual encoding and motor response has also been added to the right-hand-side of this equation. Since the effect of that constant term as well as the latency scale parameter, *F*, is only to scale the critical quantity e^{-gA_i} onto the range of the latencies (Anderson *et al*. 2004, p. 1044), we drop the constant term in favor of modeling simplicity. Instead, we account for the constant term by adjusting *F*, whenever necessary.

Given that the equation depends mainly on the activation of the chunks, any differences in activation will result in different times to respond to different tasks or trials.

Type Of User Interface, Task, User, And User Behavior

In this work, we consider only user interfaces, which contain items in a grid layout based on rows and columns. We assume that the user is initially not familiar with the layout of the items. In this case, it is not easy for a person to discriminate a target item from all distractors. We further limit ourselves to layouts that have only one item per location in this grid. Also, when we refer to an item on an interface, we are also referring to its location and vice versa. Examples of such interfaces include keyboards with an unfamiliar layout, Personal Digital Assistants (PDAs) that show a grid layout of similar looking textual or graphical items/icons, or an unfamiliar graphical application menu with items arranged in a list.

The task we consider is a simple visual search of items in such an interface, followed by a selection of the target item using a finger, a stylus, or a mouse pointer depending on the input device used.

Our aim is to mathematically model the gradual transition of novices – who do not have knowledge of item locations on the layout – to experts – who can recall multiple items and their locations successfully and ideally can do this for all items. We stay within the core mathematical framework of ACT-R's declarative memory.

With regards to learning of interface layouts by novice users, we point to the arguments of Nilsen (1991), Lee & Zhai (2004), and Cockburn, Gutwin *et al*. (2007). All of them describe in one form or the other that visual search and recall of item locations are of primary concern in spatial knowledge acquisition on a two-dimensional interface since these factors play a significant role in the early stages of skill development in such location learning.

A fundamental assumption behind our work is that at any given instant, the user will have zero or more items in a user interface that she can recall. Moreover, there will be zero or more items that she cannot recall and therefore she needs to visually search the interface to find and select them.

Model Extension For PI Effect

We next propose our extension to the base-level activation equation of ACT-R in order to account for the PI effect. We explain our model extension within the domain of tasks involving simple visual search and selection of items in user interfaces.

Decay Rate as a function of number of distractors

One way to predict the cost of searching for a target item in an interface with several similar looking items is through tracking the number of distractor items visually encoded before arriving at the target item. The number of visually encoded distractor items during a search contributes to the PI effect: The lower the number of distractors visually encoded during a search for a target item, the lower should be the decay of activation of the memory trace of the target item. Hence, the next recall of that item will be affected by the higher activation of its memory trace, leading to the lowering of its retrieval time. This will result in an improvement in the search-and-selection time during the use of the corresponding user interface. The effect of the number of visually encoded distractor items in a search task discussed here is analogous to the primary research results of Underwood (1957), Wickens (1972), and Wixted and Rohrer (1993) on Proactive Interference. Namely, they describe the effect that the number of previously learned similar items has on the recall of a target item: The higher (lower) the number of previously learned similar items is, the higher (lower) is the forgetting effect and therefore the higher (lower) is the recall latency for the target item.

In order to account for the PI effect in visual search-and-selection tasks in user interfaces, we propose a decay rate, d_j, for an item, after *j* practices of this item have been completed, as follows:

$$d_j = a + f(X_{j-1}) \qquad \textbf{Decay Rate Equation}$$

where *a* represents the decay-due-to-time constant replicating the portion of decay that occurs with passage of time, and *f* represents a decay-due-to-PI function which we will discuss shortly. X_{j-1} is the number of distractors visually encoded, at the time of j^{th} practice. Naturally, *j* has to be larger or equal to 1. X_0 denotes the number of distractors visually encoded during the first practice and is assumed to be the total number of items on the user interface. When X_{j-1} is 0, i.e. when user is able to complete the task by direct recall, without going through any explicit visual search, the decay rate equation degenerates to $d_j = a$. This implies that in the absence of the impact of distractors, the decay in activation of the item will occur only with the passage of time as in case of the traditional base-level activation equation discussed earlier.

Let us now turn to the decay-due-to-PI function, *f*. We introduce this function as one that replicates the memory decay due to proactive interference. As such, its job is to transform the number of distractors, X_{j-1}, to a valid decay-due-to-PI value. We assume valid decay-due-to-PI values to be between 0 and 0.5, both inclusive, i.e. $0.0 <= f(X_{j-1}) <= 0.5$. Since 0 implies no decay, it can be considered

as a valid lower bound on decay-due-to-PI values. The decay value of 0.5 is widely used as the decay constant in the traditional ACT-R literature and therefore can be safely considered as a valid upper bound on decay-due-to-PI values.

We assume that the maximum possible number of distractors in an interface is equal to the total number of items on it. The maximum possible number of distractors is therefore equivalent to X_0, the number of distractors visually encoded at the first practice. Hence, we set $f(X_0) = 0.5$, using the upper bound on decay-due-to-PI. On the other hand, $f(0) = 0.0$ implies the absence of the impact of distractors, and hence the absence of PI effect as a consequence. This occurs when the user is able to complete the task by direct recall.

Modified ACT-R equation of Base-level Activation

With the decay rate equation now in place, we modify the base-level activation equation to

$$A_i = \ln\left(\sum_{j=1}^{n}[qt_j]^{-d_j}\right) \qquad \textbf{PI Activation Equation}$$

where the decay d_j is not a single constant anymore, but a combination of the traditional decay-due-to-time constant and decay-due-to-PI function. The latter is a function of the number of distractors that builds up the PI effect on the recall of an item during the next practice. The factor q in the equation acts as the strength scale parameter. The usage of such a strength scale parameter, albeit in a different form and context, has been suggested previously by Anderson (1983, p. 277) as well as Stewart and West (2007, p.235).

Note that when $d_j = a$ and $q = 1$, the PI Activation equation collapses to the traditional base-level activation equation.

Our proposal for combining the effect of decay-due-to-time constant and decay-due-to-PI function is analogous to the results of experiment 3 of Keppel and Underwood (1962). There, the authors concluded that forgetting is a combined effect of the passage of time, i.e. the 'retention interval', and the number of previously visually encoded items, i.e. 'proactively interfering items'.

Activation boosts on distractors

The distractors visually encoded on the way to finding a target should be considerably less salient than the target itself. Hence, their base-level activations should receive considerably less boost compared to that of the target. Since our main interest is in replicating PI effect on the learning of target item and its location, we focus on the effect of the *number* of distractors rather than the negligible increments in strength they receive, as they are considerably less salient. For convenience of modeling, we set the reference level of activation boost to zero and consider the relative difference in boost between a target and every distractor involved during the search. We let the target get its full quota of boost during a given trial of search and selection, but set the activation boosts of distractors to the reference level, i.e. zero. This helps us to keep our analysis simple during model validation, as we will see in the next section.

Validation of Model Extension

We validate our new extension against two empirical studies on location learning in user interfaces. In order to adapt the observed data to the goal of analyzing only the PI effect, we first make a few assumptions. These assumptions help us to get an estimate of the number of distractors at any given instant. We then validate our extension by fitting it to the Reaction Time equation discussed earlier, using the data from those experiments. More precisely, we predict the average reaction time per item and per trial.

Note that the reaction time is dependent only on activation, as determined by the PI Activation Equation. All fits in this article are performed using the R^2 and root mean square deviation (*RMSD*) statistics.

Assumptions for adaptation of observed data

The heart of our extension lies in the term X_{j-1} of the decay rate equation. This term denotes the number of distractors seen at the time of j^{th} practice. In order to extract this information from the empirical data, we make the following assumptions: (i) Target items are always visible in the user interface. (ii) Target items are not easy to discriminate from the distractors. (iii) The position of an item on the interface layout does not change. (iv) We expect the user to search all items that cannot be directly recalled before finding the desired target item. This exhaustive search strategy is based on the findings of MacGregor *et al*. (1986). There, the authors carried out a visual search study on (database) menus and found that 59% of all visual searches were exhaustive in nature. (v) At any given instant, the searchable set of items is the set of all non-recallable items on the interface at that instant. (vi) On average, the visual search time is linearly proportional to the number of all items that the user cannot recall. This is warranted, since the visual search time is roughly a linear function of a given searchable set of items in the tasks where the target is not easy to discriminate from the distractors (Wolfe, 2000).

We compute X_{j-1} as follows: We first obtain the average search time per item corresponding to each session from the empirical data. Then, we use the formula

$$NIS = NISPS * ST \qquad \textbf{Distractor Computation Equation}$$

where *NIS* is a rough estimate of X_{j-1}, i.e. the number of items searched before finding the target, *NISPS* expresses the number of items searched per second, and *ST* is the search time for *NIS* number of items. We later show a sample use of this formula during our discussion of model validation. Note that in the strictest sense, *NIS* for a given trial includes the target as well. However, considering that throughout the model validation process we deal only with values that are relative and average in nature, using *NIS* as an estimate for X_{j-1} is an acceptable compromise.

Next, we show how we compute the PI-caused decay from X_{j-1} values using the decay-due-to-PI function *f*. In order to simplify our model validation process, we define *f* as a simple linear formula

$$f(X_{j-1}) = DVD * X_{j-1} \qquad \textbf{Decay-due-to-PI Equation}$$

where *DVD* is the decay value per unit distractor. The linear nature of this decay-due-to-PI equation makes it a closed-form approximation of PI on location learning. This, in turn, makes the decay rate d_j a closed-form expression as well. We later show a sample use of the decay-due-to-PI equation during our discussion of model validation.

Location Learning on a Graphical Virtual Keyboard

Cockburn, Kristensson *et al*. (2007, fig. 2, p. 1574) carried out an experiment that tests how well users learn the location of keys on a graphical virtual keyboard with one label per key. The labels were iconic symbols chosen from the Microsoft Webdings font. For the validation of our model, we focus only on the condition where the labels on the keys are always visible, i.e. the Visible Interface condition in that study.

All participants trained for 5 minutes through 10 iterations of searching and selecting symbols on the interface containing 18 iconic symbols, which were pre-cued in a separate target-cuing region. For our validation, we had to make a few assumptions, as the corresponding information was not given explicitly in that paper. These assumptions are as follows: An iteration consists of a sequence of trials. Each of the 10 iterations takes roughly equal time and each of them gets completed in 30 seconds on average – since 10 iterations took 5 minutes or 300 seconds as stated in that paper. We also assume inter-trial, and inter-iteration periods to be constant. Also,

except for the target-precue, we assume that environmental context cuing is minimal and can be ignored for our purposes.

Based on this, we now detail a sample computation of X_{j-1} using our Distractor Computation Equation. For iteration #1, we assume that the user exhaustively searches all 18 keys before hitting the target, i.e. the *NIS* corresponding to iteration #1 is 18. From the measured data we see that the search time, *ST*, corresponding to iteration #1 is 2.4 sec. Consequently the number of items searched per second, *NISPS*, is 7.5. Next, using *NISPS* = 7.5, we compute the *NIS* value corresponding to the *ST* for each iteration. These *NIS* values are then used for X_{j-1} (j = 1 to 10) in the Decay-due-to-PI Equation.

Note that for a given iteration or session, it is sufficient to use the average number of distractors, X_{j-1}, directly for computing an average activation per target through the PI Activation equation. This is possible since we consider the relative activation boost for distractors to be zero at any given trial, as mentioned previously.

We now detail a sample computation of *f* using our Decay-due-to-PI Equation. For iteration #1, we use the boundary condition $f(X_0) = 0.5$, which implies $DVD * X_0 = 0.5$. Since $X_0 = 18$, the decay value per unit distractor, *DVD*, is 0.028. Using this value for *DVD*, we compute the *f* value based on the X_{j-1} for each iteration.

Table 1 shows the *NIS* and the corresponding $f(X_{j-1})$ values for each iteration. Note that for simplicity, we assume the average *NISPS* to be same over all iterations. The same holds for the average *DVD* as well. The assumptions are warranted since the average *NIS* and *DVD* values themselves are only relative in nature.

Table 1. Relative estimate of the number of distractor items searched before finding the target item, in each iteration (for *NISPS* = 7.5) and the corresponding decay-due-to-PI value (for *DVD* = 0.028).

Iteration *j*	ST (observed search time per item, in secs)	NIS (approx. number of distractor items searched, X_{j-1})	$f(X_{j-1})$ decay-due-to-PI
1	2.400	18	0.500
2	2.031	15	0.417
3	1.892	14	0.389
4	1.708	13	0.361
5	1.673	13	0.361
6	1.592	12	0.333
7	1.569	12	0.333
8	1.431	11	0.305
9	1.465	11	0.305
10	1.408	11	0.305

Figure 1 shows our model fit to the observed data. We have set the values for the model fit parameters as follows: (i) The decay-due-to-time constant *a* in the decay rate equation is 0.058. In absence of any inter-trial and inter-iteration data in this empirical study, we assume that there have been insignificant pauses between any two consecutive trials or between any two consecutive iterations. Hence, we choose a relatively small value for the decay-due-to-time constant, implying that the decay due to passage of time had been minimal. (ii) The latency scale *F* is 0.96. This maps an activation value to its corresponding time value. Further, it also takes the fixed costs associated with visual encoding and motor response into account. (iii) The strength scale *q* is 150. (iv) The latency exponent scale *g* is 0.2. The last two parameters help in an overall adjustment of the activation value. With $R^2 = 0.992$ and $RMSD = 0.074$ for our prediction, our model extension closely agrees to the observed data.

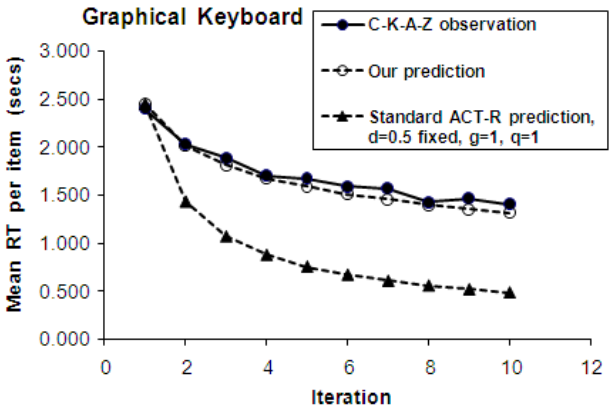


Figure 1. Mean Reaction Time, RT (in secs) per item (label) selected on a graphical keyboard, as observed in (Cockburn, Kristensson *et al.* 2007, fig. 2, p. 1574), named C-K-A-Z, the solid line with filled circles. Our prediction is the dashed line with unfilled circles (R^2=0.992, RMSD=0.074). Prediction by Standard ACT-R at d=0.5 (fixed default decay), g=1, q=1, is the dashed line with filled triangles (R^2= 0.952, RMSD= 0.824).

As evident from Figure 1, the prediction from our modified equations with a RMSD of 0.074 is significantly better than the prediction of reaction based on the standard ACT-R declarative memory equations with a RMSD of 0.824. In case of the standard ACT-R based calculations, the constant time-based decay parameter *d* in the base-level activation equation was left at its default value of 0.5 and the latency exponent scale parameter *g* in the reaction time equation was left at its default value of 1.

It should be noted that our choice of 0.058 for the decay-due-to-time constant *a* is so small that the term can be removed without incurring any significant change in the shape of the predicted curve. With this simplification, we can claim that we have introduced only a single new parameter into ACT-R theory of declarative memory, namely the strength scale *q* (see the PI Activation Equation).

Learning of Static and Unfamiliar Menu

Cockburn, Gutwin et al. (2007, fig. 2, p. 632) carried out an experiment that tests how well users learn the location of menu items in a single column, single level menu where the items are never relocated and all items are displayed at the same time to the user. The menu items were words that were unfamiliar to the user in this study. We are thus referring to the "Static+Unfamiliar" menu condition in that study.

The menu-item search and selection trials were executed by the participants in a series of 7 blocks. Participants began each trial by clicking on a 'Menu' button, which caused the menu to be shown and also the name of the target to appear beside it. For our model validation, we assume a menu of 8 items. We use this length since it is the next highest integer to the average of the menu lengths studied.

For our model validation and due to the lack of more accurate information, we assume the following: Each block consisted of a collection of trials. Each of the 7 blocks takes roughly equal time and gets completed in 10 seconds on average. We also assume inter-trial, inter-block periods to be constant. Again, except for the target-precue, environmental context cuing is assumed to be minimal and therefore ignored for our purposes.

We compute the X_{j-1} for the 7 blocks using the same technique as in the previous study. For block #1, let us assume that the user

exhaustively searches roughly all 8 menu-items before hitting the target, i.e. *NIS* corresponding to block #1 is 8. In figure 2, we see that the observed search time, *ST*, corresponding to block #1 is 0.819 sec. Therefore, the number of items searched per second, *NISPS* is roughly 10. Using $NISPS = 10$, we compute the *NIS* value corresponding to the *ST* for each block. These *NIS* values become the values for X_{j-1} (j = 1 to 7) in the Decay-due-to-PI Equation.

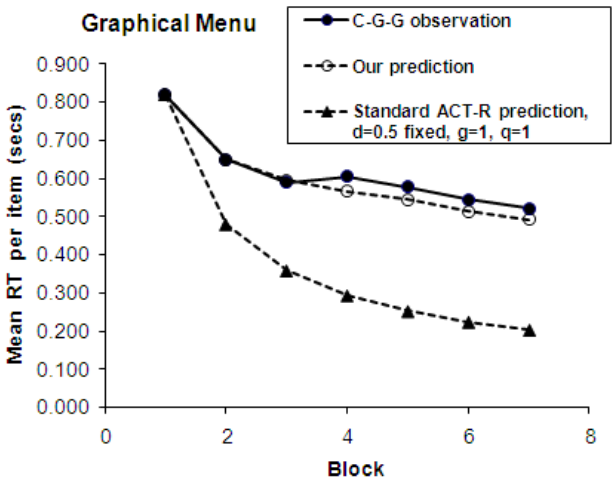


Figure 2. Mean Reaction Time, RT (in secs) per item selected on a graphical menu, as observed in (Cockburn, Gutwin et al. 2007, fig. 2, p. 632), named C-G-G, the solid line with filled circles. Our prediction is the dashed line with unfilled circles (R^2= 0.978, RMSD= 0.026). Prediction by Standard ACT-R at d=0.5 (fixed default decay), g=1, q=1, is the dashed line with filled triangles (R^2= 0.969, RMSD= 0.264).

Next we compute *f* using our Decay-due-to-PI Equation. For block #1, we use the boundary condition $f(X_0) = 0.5$, which implies $DVD * X_0 = 0.5$. Since $X_0 = 8$, therefore the decay value per unit distractor, *DVD*, is 0.0625. Using this value for *DVD*, we compute the *f* value based on the X_{j-1} for each block.

Figure 2 shows the fit of our model to the observed data. We have set values for the model fit parameters following similar arguments as in the previous example: (i) The decay-due-to-time constant, *a*, in the decay rate equation is 0.058. (ii) The latency scale, $F = 0.362$. (v) Strength scale, $q = 150$. (vi) Latency exponent scale, $g = 0.2$.

As evident from Figure 2, with $R^2 = 0.978$ and $RMSD = 0.026$, our adapted model shows good correspondence to the observed data. Also, the prediction generated from our modified equations is much better than the prediction based on the standard ACT-R declarative memory equations, with an RMSD of 0.264. Similar to the previous example and for the standard ACT-R based calculations, the constant time-based decay parameter *d* and the latency exponent scale parameter *g* were left at their default values of 0.5 and 1 respectively.

Discussion

General Comments

Our proposed mathematical extension to the ACT-R theory of declarative memory model closely predicts the PI effect on location learning in user interfaces. The model is based on the number of distractor items visually encoded on the way to finding the target item. Our proposal directly quantifies the PI effect on location learning at a high level of abstraction, and is based on well established results from PI studies. There are few potential concerns with the analysis described above that we enumerate below.

In our model, we implicitly assume that the number of distractors visually encoded at the time of j^{th} practice, i.e. the value for the term X_{j-1} in the decay rate equation, will be estimated by some visual search module whose implementation lies beyond the scope of this work.

We set the latency scale parameter *F* to different values for the two predicted curves; one being relevant to our model extension and the other being relevant to the original ACT-R equations of declarative memory. We decided to do this in order to match their co-ordinates for the first session (i.e. iteration #1 in the first example and block #1 in the second example) to the co-ordinates of the first session of the observed data. Such adjustment merged the session #1 co-ordinates of the three curves (two predicted and one empirical) into a single reference point thereby making visual as well as quantitative comparison of data easier. Since the effect of *F* in the reaction time equation is only to scale the critical quantity e^{-gA_i} onto the range of the latencies (Anderson *et al*. 2004, p. 1044), we can safely consider that changing *F* has a negligible effect on the shape of the curve. Hence, we can state that our decision to set *F* to different values for different predicted curves was an acceptable compromise.

We set the value of the strength scale *q* to 150 and the latency exponent scale *g* to 0.2 in order to match the shape of our predicted curves to the corresponding observed data as closely as possible. While traditionally *q* and *g* have been left at their default values of 1, still our choice of the same value for *q* and *g* across both the studies, albeit different from the default, avoids compromising the fidelity of our new model to a considerable extent.

In order to validate our model, we needed to extract the number of distractors at a given practice (i.e. X_{j-1} in decay rate equation) from the empirical studies, which did not report this information directly. Hence we were forced to make assumptions that enable us to extract a rough average estimate of the number of distractors per practice, at a given session, from those studies. Although these relative estimates seem sufficient to demonstrate our model's ability to replicate the PI effect, we feel that a future empirical study that directly measures the number of distractors visually encoded by a novice user on the way to finding a target item in a given layout would be worthwhile. However, this would involve eye tracking and a very carefully constructed experiment. Such an effort would enable us to identify more accurate values of X_{j-1}, thereby increasing the fidelity of our model extension further.

Comments on computational design: A suggestion

We now briefly suggest one possible way to implement the computation model to simulate the PI effect as presented here.

We assume that we are given a visual search module that is based on the attentional vision module of standard ACT-R software. We use this module as a black box and assume that it is able to return us a list of distractors for every time the layout in question is scanned for a pre-cued target item. We also assume that the positions of items in the layout do not change; the target item always exists in the layout and is found whenever searched for.

The distractors visually encoded on the way to finding a target should be considerably less salient than the target itself. Hence their memory strengths should get significantly smaller boosts than the target. For simplicity of our design, we assume that, every distractor gets zero boost in its memory strength, while in comparison the target gets the full quota of boost it deserves, at every execution of the visual search and selection task. One way to realize this would be through exercising appropriate control on buffer clearing in the productions. The other way to realize this would be through explicitly using the getter and setter functions for manipulating base-level activations of the chunks from within the productions.

In the Lisp implementation of ACT-R, there are many side-effects, i.e. situations where code in the model that explicitly does

one thing also causes other actions to be performed that are not explicitly represented in the model code (Stewart and West, 2007). In order to avoid such side-effects, we recommend to avoid manipulating the attributes of visual location chunks or the visual object chunks of the vision module; instead, we recommend to maintain a parallel set of user-defined chunks, each containing information related to an item on the layout. Whenever a pre-cued target item is found and the distractors involved in the search are identified by the aforementioned visual search module, the memory strength of the user-defined chunks representing the target and its distractors can then be updated appropriately.

Summary

The work reported in this paper developed a model extension that captures the proactive interference effect on two-dimensional location learning. The extension was added to ACT-R's model of declarative memory strength and recognition/recall reaction times. The model was then validated by fitting the data of two previous experiments that tested location learning on a graphical virtual keyboard and a graphical menu. The new model resulted in a significantly better fit to the observed times.

References

Altmann, E. M., & Schunn, C. D. (2002). Integrating Decay and Interference: A New Look at an Old Interaction. *Proceedings of the 24th Annual Conference ofthe Cognitive Science Society*. (pp. 65-70). Mahwah, NJ: Erlbaum.

Anderson, J. R. (1983). A Spreading Activation Theory of Memory , *Journal of Verbal Learning and Verbal Behavior*, 22, pp. 261-295.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Quin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4), 1036–1060.

Anderson, J. R. and Lebiere, C. (1998). *The atomic components of thought*. Lawrence Erlbaum.

Cockburn, A., Gutwin, C., & Greenberg, S. A. (2007). A Predictive Model of Menu Performance, *CHI* 2007, 627-636.

Cockburn, A., Kristensson, P. O., Alexander, J., and Zhai, S. (2007). Hard Lessons: Effort-Inducing Interfaces Benefit Spatial Learning, *CHI* 2007, 1571-1580.

Darken, R., and Sibert, J. (1996) Wayfinding in Large-scale Virtual Environments, Proc. ACM CHI 1996, 142-150.

Keppel, G., and Underwood, B. J. (1962). Proactive inhibition in short-term retention of single items. *Journal of Verbal Learning & Verbal Behavior*, 1, 153-161.

Lee, P. and Zhai, S. (2004). Top-down learning strategies: can they facilitate stylus keyboard learning? *Int. J. Human-Computer Studies*, 60, 585–598.

Leung, H-C. and Zhang, J. X. (2004). Interference resolution in spatial working memory. *NeuroImage*, 23, 1013–1019.

MacGregor, J., Lee, E., & Lam, N. (1986). Optimizing the structure of database menu indexes: A decision model of menu search. *Human Factors*, 28, 387–399.

Nilsen, E. L. (1991). Perceptual-motor control in human-computer interaction (Technical Report Number 37). University of Michigan, Ann Arbor, MI.

Stewart, T. C. and West, R. L. (2007). Deconstructing and reconstructing ACT-R: Exploring the architectural space. *Cognitive Systems Research*, 8, 227–236.

Underwood, B. J. (1957). Interference and forgetting. *Psychological Review*, 64, 49-60.

Wickens, C. D. and Hollands, J. (2000). *Engineering psychology and human performance*. 3rd Ed. pp. 252, Prentice Hall.

Wickens, D. D. (1972). Characteristics of word encoding. Melton & Martin (Ed.), *Coding processes in human memory*. pp. 195-215.

Wixted, J. T. and Rohrer, D. (1993). Proactive Interference and the Dynamics of Free Recall. *J. of Expt. Psychology: Learning, Memory, and Cognition.* 1993, Vol. 19, No. 5, 1024-1039.

Wolfe, J. M. (2000). Visual attention. In K. K. De Valios (Ed.), *Seeing* (2nd ed., pp. 335–386), Academic Press.

Appendix

We show values from few functions corresponding to the first study, Location Learning on a Graphical Virtual Keyboard. Constant parameters are a=0.058, F=0.96, q=150, g=0.2. All are average values per target. X_{j-1} values are from Table 1. Human data (search time) is rightmost.

Iteration# j	X_{j-1}	d_j	t_j (sec)	e^{-gA}	$T = F * e^{-gA}$ (sec)	Observed search time (sec)
1	18	0.558	30	2.556	2.454	2.400
2	15	0.475	60	2.097	2.013	2.031
3	14	0.447	90	1.889	1.813	1.892
4	13	0.419	120	1.745	1.675	1.708
5	13	0.419	150	1.661	1.595	1.673
6	12	0.391	180	1.577	1.514	1.592
7	12	0.391	210	1.521	1.460	1.569
8	11	0.363	240	1.458	1.400	1.431
9	11	0.363	270	1.413	1.356	1.465
10	11	0.363	300	1.377	1.322	1.408

Cognitive Modeling of Strategies in Dynamic Tasks

Alberto De Obeso Orendain (a.de-obeso-orendain@sussex.ac.uk)
Representation and Cognition Group, School of Informatics
University of Sussex, Falmer, Brighton, BN1 9QJ, UK

Sharon Wood (s.wood@sussex.ac.uk)
Representation and Cognition Group, School of Informatics
University of Sussex, Falmer, Brighton, BN1 9QJ, UK

Abstract

Computer simulations, or microworlds, have been used for studying various topics including problem solving. This work investigates strategies for complex, dynamic problem solving in a fire-fighting microworld. Using data from a study by Cañas, Antolí, Fajardo & Salmerón (2005), an ACT-R cognitive model is developed with the aim of providing insight into the development and selection of strategies participants use. One particular behavior observed in participants when trained repetitively on the same scenario, the creation of a fire-break barrier to prevent the fire spreading, is discussed. It was found that selection of a particular strategy depends on the fine-tuning of ACT-R production rule utilities as a consequence of environmental rewards, highlighting the role of reward size and timing. The model is able to capture various aspects of the data by promoting a free competition of small blocks of behavior based on rational analysis. A key finding is that good performance is linked to effective combination of strategic control with attention to changing task demands reflecting time and care taken in informing and effecting action.

Keywords: Cognitive Modeling; ACT-R; problem solving; strategy; microworlds.

Introduction

Microworlds are computer simulations that represent a middle point between naturalistic scenarios and laboratory tasks (Brehmer and Dörner, 1993). Although microworlds are relatively simple, they embody the essential characteristics of real-world dynamic decision-making environments (Gonzalez, Vanyukov and Martin, 2005). Microworlds allow for an economic and standardized presentation of scenarios, data registration and computing of results (Frensch and Funke, 1995; Brehmer and Dörner, 1993). These tasks have been used for studying various domains including problem solving (Frensch and Funke, 1995; Brehmer and Dörner 1993; Taatgen 2005).

Microworlds have three characteristics. Firstly, complexity, owing to the number of elements and number (and nature) of their interrelationship (Frensch and Funke, 1995). Second, lack of transparency; the problem solver does not have access to all relevant task information, making interaction with the world necessary for knowledge requirements. Last, the problem state changes both independently and as a consequence of the participant's actions. Microworlds consequently place a variety of cognitive demands on the problem solver. According to Anderson et al. (2004) dynamic tasks require considerable goal-directed processing within demanding perceptual displays and execution of motor commands under severe constraints. They require continuous processing of feedback in order to select appropriate actions within an ever-changing situation (Brehmer and Dörner, 1993). This paper focuses on the demands posed by these dynamic task characteristics, in particular the way performance feedback from a dynamic environment is processed, and how this allows the consolidation of strategies.

Frensch and Funke (1995) suggest that it is important to understand the process of Complex Problem Solving (CPS), rather than the product; this process is an interaction between the problem solver, the task and the environment. A cognitive model is able to reveal the internal processes for selecting actions together with their interaction with the environment, increasing our understanding of these processes. Cognitive modeling has been used in dynamic environments such as air traffic control (Taatgen, 2005). The work presented here uses the FireChief fire-fighting microworld (Omodei & Wearing, 1995).

The FireChief Microworld

FireChief participants combat fires spreading in a landscape using truck and copter units. Trials last 260 seconds. A FireChief scenario is specified by a variety of properties such as landscape distribution of forest, clearings and property, the number and position of initial fires, the direction and strength of the wind, and the initial position of fire-fighting units. Figure 1 shows the central cells of a FireChief trial display converted for model use. Copters (shown as CR) and trucks (TR) can move between landscape grid cells (R, L & H) and can Drop Water (DW) over cells to extinguish fires (F*n* where *n* indicates fire intensity). Copters are three times faster than trucks and cannot be destroyed by fire, but a truck's water tanks have twice the capacity and are able to Control Fire (CF) by creating a fire-break. Commands are issued through a combination of mouse and keyboard operations and their execution takes a fixed amount of time, 4 seconds to DW, 2 seconds to CF, and a variable amount of time to Move a unit depending on distance and type of unit. Wind strength and direction are in the upper right-hand corner of the display.

FireChief is a dynamic decision-making problem solving task environment where a series of interdependent decisions

are required to reach the goal, the environment changes over time, and user actions change the state of the world (Gonzalez et al., 2005). The problem solver is engaged in a strategic situation where he or she has control over a limited number of fire fighting units and has to use them to accomplish one mission: to fight and quell the fire. Task performance is inversely proportional to the number of cells destroyed by fire at the end of the trial.

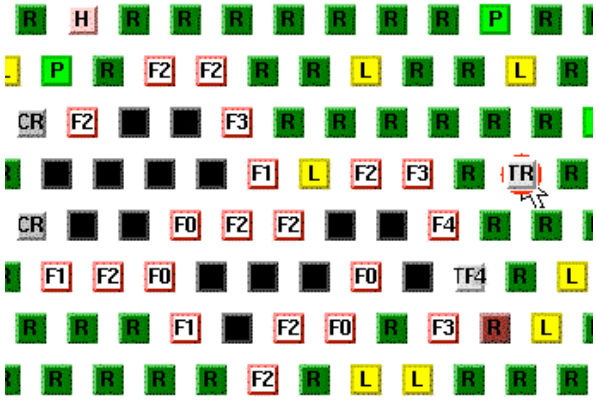


Figure 1: Central cells of the *model version* of a FireChief trial display using 'buttons' (Lisp)

ACT-R Architecture

The CPS model is implemented in ACT-R 6.0 (Anderson et al., 2004). ACT-R is divided into various modules according to the kind of information they process: a visual module for identifying objects in the visual field (in Figure 1 the focus of attention is on the cell in the fourth row of the penultimate column), a manual module for controlling the hands (the mouse pointer is located in the same cell), a declarative module for retrieving information from memory, and a goal and imaginary modules for keeping track of current goals and intentions. Communication between modules is achieved through buffers where the content of any buffer is limited to a single declarative unit of knowledge, a 'chunk'. Thus the system can only respond to a limited amount of information. Behavior in ACT-R occurs through interaction of its specialized modules via the buffers, coordinated by a central production system.

There are two types of knowledge in ACT-R: chunks encode declarative knowledge whereas procedural knowledge is represented by production rules, where each rule corresponds to a cognitive processing step. Each ACT-R production has two elements: the condition, a combination of states from the different buffers, and an action, which can perform transformations over the state of buffers and trigger actions in modules. ACT-R functionality is achieved through many mechanisms, but two are of the utmost importance in this model: utility and reward.

Utility designates the value of executing a rule; it represents the perceived value of a production and is updated by rewards from the environment. Utility of productions is compared during the process of conflict resolution where only the rule with the highest utility is acted upon. From a computational perspective, a participant can be considered as a collection of utility values. By interacting with FireChief, these utility values are tuned throughout a sequence of trials in a unique fashion within constraints imposed by the properties of the FireChief task, the procedural knowledge represented by rules, and rewards from the environment. The combination of ACT-R utility learning mechanisms with the dynamic nature of FireChief means the model can run a number of times under the same task conditions with the same knowledge and yet produce a different pattern of behavior each time. Rewards are the ACT-R mechanism for giving the model feedback from the environment. When a reward is triggered the utilities of all productions that have fired since the last reward are updated. The amount and distribution of rewards have an important impact on model's behavior (Janssen, Gray and Schoelles, 2008).

Human Study Data

The data used for specifying and fitting the CPS model comes from a study by Cañas et al. (2005). Those participants trained on the same, reliably predictable FireChief scenario for 16 trials were found to increasingly preferentially select the fire-fighting strategy that achieved the best outcome. This paper focuses on modeling strategy selection during constant training in order to understand this process and thereby gain insight into strategy formation. The constant scenario is characterized by a strong, constant easterly wind. Participants are limited to 2 copters and 2 trucks. To begin with there are two groups of fire in close proximity which quickly spread eastward (Figure 1 shows their initial distribution). A variety of different strategies can be used to stop the fire, as described in the next section.

Strategy Use

In total, 1728 protocols from 72 participants were analysed to identify four main strategies. In the *Non-Barrier* strategy CF commands are issued with noticeable spatial dispersion and are interleaved with DW commands. In the *Stop* strategy DW commands are used alone and are issued over the most intense fires within sufficient proximity to stop the fire. In the *Follow* strategy only DW commands are used but they do not target the strongest fires nor are they issued in close proximity to each other. The most structured strategy is called *Barrier* and it turns out to be very effective in the constant training scenario; it is used twice as often (50 vs. 27) by the top four performers compared to the four worst. For these reasons it is discussed here in more detail.

The Barrier strategy

The *Barrier* strategy presents a very characteristic way of dealing with the fire: the issuing of an ordered pattern of CF commands in a shape, similar to a barrier, intended to stop the fire spreading. There are many forms in which the barrier is created but a semicircle or straight line is the most frequent. In Figure 2 the barrier has the form of a semicircle where the black squares represent CF commands and the

grey squares represent DW commands. The strategy recruits top-down processes in constructing a fire-break but is sensitive to bottom-up perceptual processes so the final form of the barrier is a function of the shape of the fire that is being controlled.

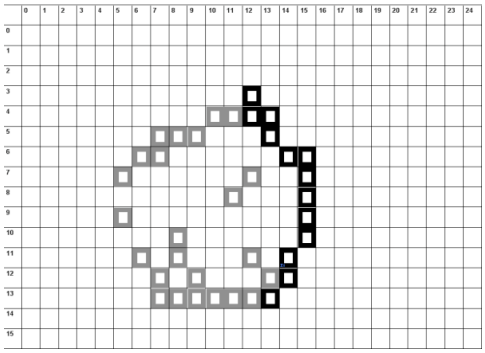


Figure 2: A typical *Barrier* strategy formation.

The Cognitive Model

To allow interaction between ACT-R and the FireChief task a Lisp version of FireChief was developed following the original specification provided in the FireChief manual (Omodei & Wearing, 1993). This is able to control all relevant aspects of the task: the landscape, development of fire, execution of commands, and performance calculations. Before running the model a FireChief scenario is loaded in an experimental window in the form of a matrix of multi-colored labeled buttons. Buttons enable interaction between the model and the experimental window by means of mouse and keyboard commands

The model implements all four main strategies, deciding which to use (based on initial utility comparisons) or switching to another (as utilities change) during the trial if the fire is not under control. An ineffective strategy, poorly rewarded, can be abandoned at any point, therefore. A chosen strategy is held in the imaginal buffer and affects model behavior by defining, for example, whether the model will use a mixture of DW and CF commands, whether or not a barrier will be created, or which ways of attacking the fire are preferred. In the very first trial the rules that select a strategy have an initial random utility determined by the standard ACT-R utility equation that has a random component. After the trial ends the utility of these productions is modified according to the final result. In this way, the actual means of executing a strategy emerges by rewarding certain rules over others (so a strategy is more precisely a *set* of strategies manifesting similar behaviour).

Creating a barrier

The functional block of rules described here belong to the set of strategies for creating barriers (see Figure 3). These rules represent a small subset of all the productions that are available to the model which is able to select and perform any of the four main strategies identified from the human data analysis. A FireChief trial lasts 260 seconds and a typical barrier is created in 60 seconds. Each cell in a barrier requires a Move followed by a CF command and the average number of grid cells needed for a barrier is 15. The average number of commands in a trial is 110.

First the model must specify a starting point for the barrier. This will depend upon the current state of fire and wind conditions. Second, the location of the next section of the barrier must be determined. A design decision was that the form of the barrier should be the result of a competition for locating the next cell of the barrier; top-down and bottom-up processes compete through the ACT-R conflict resolution mechanism. The selection of a target cell follows a process in which the candidate cell is proposed and then various tests (based on perceptual actions) are conducted. Third, a truck is moved to the selected cell before executing a CF command comprising a sequence of steps: locate the target, store location of target in working memory, find a truck, attend the unit, move the cursor to the unit, click the unit, attend target, move mouse to target, click mouse. Of these actions moving a cursor shows the highest time variability in the model (this information is not recorded in the human study protocols) stressing its importance in the total latency of the command and its corresponding importance to overall performance. When the truck has finished moving a CF command can be initiated. Fourth, the status of the barrier is monitored. Eventually, the barrier is considered complete when the fire-break is sufficient to contain the fire. The shape of the resulting barrier is a product of competition between various rules and the reward they receive when executing commands.

In the excerpt shown in Figure 3, the model is following the *Barrier* strategy and has just started a Move command with a truck. The current intention of the model is to create a fire-break barrier using CF commands. In step 1 the model must choose between waiting for the truck that has initiated its movement (and is disabled until it arrives) or using the other truck. In this step the utilities of productions 1-A and 1-B are compared and the one with the highest expected value is fired. In this case the model decides to wait. In step 2, the model searches for a visual-location that satisfies a set of constraints. In this example the model is verifying if the truck has arrived at its destination. The first constraint is spatial: the column and row of the destination cell. The second constraint is graphical: the cell must have a light-grey color (if the destination cell is white it means that the truck is still moving). The result of this search determines step 3. If the truck has not yet arrived, the model returns to step 1. When the model detects that the truck has arrived at its destination a shift of attention is made to that location. At the end of this attention shift the visual buffer is loaded with a chunk representing the content of the cell, namely the type of landscape and whether the cell is on fire (plus its intensity). Step 4 starts by checking whether the visual chunk encoded in the visual buffer is a product of an explicit shift of attention or the product of *buffer stuffing*. Buffer stuffing is an ACT-R mechanism in which a chunk is stored in the visual buffer without an explicit request from a

production rule. This can be a recurrent source of distraction for the visual system but also allows the detection of unforeseen events (for example new fires appearing in the scenario). In this example, if the model is distracted a visual chunk (that does not represent the location details for where the CF is going to be executed) is placed in the visual buffer. If the model proceeds with step 4 it will move the mouse pointer to the cell that distracted its attention instead of the correct cell. If the visual element encoded in the visual buffer is a product of the explicit attention shift executed in step 3, the CF command can be applied there because now the unit is in position. Before issuing a CF command the mouse pointer must be located over the truck, so step 4 initiates a mouse movement towards the attended cell. During this time the target cell may catch fire; in this case the model aborts the execution of the CF command. In step 5, after the mouse movement is complete, the CF command is initiated by pressing a key. In the normal flow of events the CF command would start after the click. Figure 3 shows a different outcome: just after rule 5-A fires the target cell catches fire, rendering the execution of a CF command impossible and consequently an alarm is emitted. Following this, the model is able to detect this alarm and, making use of the contents of the imaginal buffer, can select an appropriate course of action based on its strategy choice.

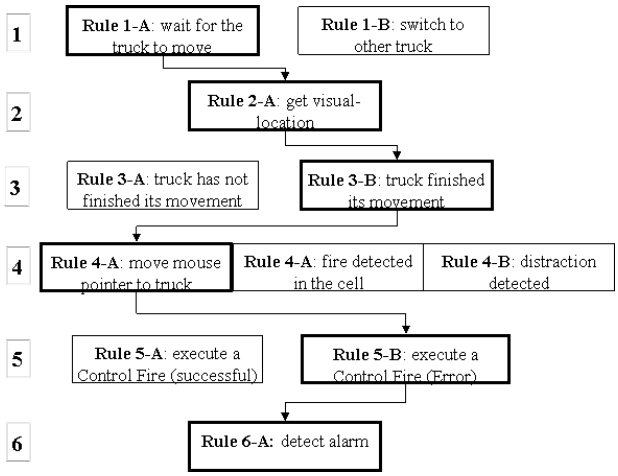


Figure 3: Sequence of Barrier strategy rules

A model run lasts 4160 seconds (16 sessions of 260 seconds). The model was run 40 times, following the same experimental design as in the Cañas et al. (2005) study. The data generated by the model provides a complete protocol of interaction with FireChief, as for each human participant, as well as a detailed trace of the operations being executed inside its various modules.

Data Fitting

During initial development, the simplest natural model was implemented based on a GOMS (Card, Moran, and Newell, 1983) analysis of the task and then fitted to the human study data. This initial model was highly efficient: all units were used all the time so time wasted was negligible. This initial model also followed a rigid strategy specification; however, the data reveals that participants do not use time as efficiently as in the initial model nor do they repeatedly execute the same strategy, making the importance of achieving flexibility in behavior evident. The approach adopted was to provide the model with complete knowledge about all the available strategies (cf. Gray & Boehm-Davis, 2000) but to allow them to compete freely based on their perceived utility.

Various reward schemes were tried, the most successful being the one that focuses on individual commands. In the 'single reward' scheme a reward (based on final performance) is given at the end of the trial. In the 'reward sub-task' scheme the completion of salient tasks is rewarded. For example, in the *Barrier* strategy stages are completion of a barrier, refilling a unit, or extinction of the fire. The problem with both these schemes is that, because several hundreds of rules may fire between rewards, the utility values of the most recent rules are changed only. This affects the model's behavior because the rules responsible for achieving good performance may not receive the proper reward and hence appropriate learning is deterred. In the scheme selected for use here positive rewards are awarded for successfully completing individual commands and negative rewards for executing unsuccessful commands and wasting time. Executing Move and CF commands generates a fixed amount of reward but the reward of a DW command is a function of the intensity of the fire that is extinguished.

In fitting the model there was no attempt to obtain the exact behavior of any individual; rather, data fitting centered on identifying decision points, encoding rules for executing actions and assigning rewards.

Results

Three metrics are used here to compare behaviour: task performance (reflecting appropriate strategy use); command duration (reflecting underlying cognitive and other processing steps); and interactions between commands (reflecting performance-related functional relationships between the Move and the CF and DW commands). There are other metrics not discussed here.

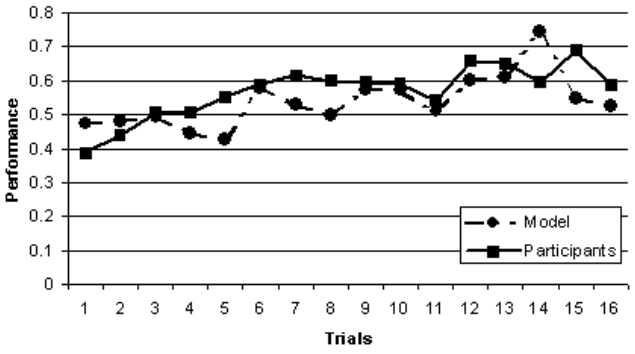


Figure 4: Comparison of performance between model and Cañas et al (2005) study participants

Figure 4 compares performance in the constant training condition for participants and the model. As can be seen, the model is able to replicate performance levels and also capture the incremental improvement in performance (R=.538). A significant performance increment was obtained by comparing the first and last four trials for both participants and the model. ($F(1,33)=4.417$, $p<.05$ and $F(1,33)=5.17$ $p<.05$ respectively).

	Performance		Frequency(%)	
Strategy	**Data**	**Model**	**Data**	**Model**
Barrier	81.59	81.05	0.65	0.66
NonBarrier	72.38	71.74	0.17	0.18
Stop	91.98	71.3	0.02	0.11
Follow	57.69	66.42	0.16	0.06

Table 1: Strategy use during constant training trials

Table 1 shows that *Barrier* is the most frequently used strategy during constant training[1]. Due to the high wind strength in the constant training scenario it is very difficult to stop the fire using DW commands only.

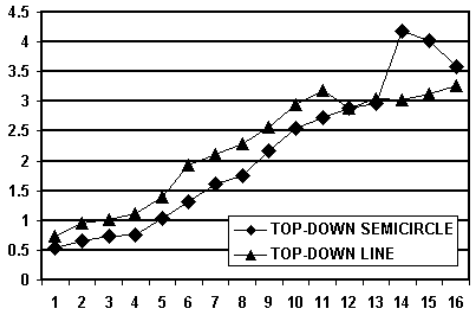


Figure 5: Increase in production utilities during consolidation of the *Barrier* strategy

A typical run of the model involves around 200 decisions, and the execution of each decision requires between 1 to 6 rules. On average the model executes 103 commands and participants execute 110 commands per trial. The model improves performance due to the tuning of its production utilities to the constant training trial scenario. Figure 5 shows how the utility of productions related to the creation of the barrier steadily increases as trials are completed. This continuous increment of utility values implies that FireChief commands are being completed with success with more frequency over trial runs.

[1] The good performance shown in the human data for the *Stop* strategy is based solely on two participants who used it extremely successfully from the outset whilst other less proficient participants rapidly abandoned it in favour of more reliable strategies.

Good vs. Bad Performers

A comparison of the best and worst performers in the constant training condition is presented with the aim of showing how utility values can be used for understanding more about participant behavior. Performance metrics for the top four participants in the constant training group are compared with the worst four participants and the same is done with model data. The best performers use the *Barrier* strategy twice as often as the worst performers (50 vs. 27 times) and performers/model-runs have an average performance per trial of 86.80/87.41 while the worst performers have an average performance of 70.91/71.80 when using this strategy.

All participants and model-runs, take a similar amount of time to issue a CF command that forms part of a fire-break barrier ($F(78,1)=.637$, $p=.427$) and ($F(81,1)=1.792$, $p=.185$), so the performance differences do not lie here. However, there is a functional dependence between moving a unit and issuing a CF command. Before executing a CF command the truck must be moved to the right place. The model embodies the assumption that the decision about where to move the truck is taken when the execution of the movement is initiated. There is a significant difference between the best and worst participants in the time it takes to execute a movement prior to issuing a CF command when forming a barrier ($F(1260,1)=67.980$, $p<.001$). The model captures latency times for the best performers only; worst performers spend much less time on this activity than the model. The best approximation to worst performance provided by the model is to execute only a single perceptual action to ascertain the fire location without checking whether the target fire-break cell is on fire. The model uses the fire-front for selecting where in a particular row the next fire-break cell should be, and poor performers often get this wrong (see next section). Even so, the model remains slower than participants by 800ms. on average. Even if all perceptual and cognitive processing could be removed from the model it cannot reduce the time taken by a sufficient amount to match human latencies. An explanation for this could be connected to the duration of motor commands: a Move command requires two key-presses and two mouse pointer moves. Perhaps poor performers execute these actions with more hastiness. Evidence to support or refute this explanation is subject to ongoing research

Utility profile

With the aim of gaining insight into what differentiates best and worst performers, two profiles were created based on utility values for each group from the model run. To obtain the profiles, the utility of relevant productions for each group is queried at the end of the training phase and averaged. In doing this, the comparison is focused only on the rules relating to the creation of a barrier: the way trucks are used, how they are moved, and how the barrier is created.

The comparison shows that the most striking difference between good and bad performers is that good performers

successfully combine top-down and bottom-up processes to create a barrier, while the worst performers apply only top-down processes successfully, failing to combine them well with bottom-up processes so that cells selected for the fire-break prove less effective. The key differences are that the best performers pay more attention to the fire-front, and also that they wait for the trucks to finish their (short) movements before executing a CF command, thereby completing the sequence of commands successfully. These differences can be identified by looking at the utility values of the productions that compete at the relevant decision points (as in Figure 5).

Discussion

This paper is focused on the adaptive selection of strategies for fire fighting with the aim of demonstrating how cognitive modeling can improve our understanding of problem solving behavior when interacting with dynamic microworlds, with implications for real-world complex problem solving. The model continuously interleaves cognitive with perceptual-motor operations, selects different strategies and implements them according to the reward structure of the task. A particular implementation of a strategy depends on the fine-tuning of ACT-R production rule utilities as a consequence of environmental rewards and thus is a product of both the configuration of the trial (in this case the constant training trial) and the history of interactions between problem solver and task (which is stored in the collection of utility values). As noted by Cañas et al. (2005) the constant training condition allows participants to consolidate strategies (see Figure 5).

The most important learning mechanism for the model is the one that updates utility. The main objective during the fitting of the model was to allow rules to be rewarded (or punished) by their effects in the environment, however the set of available strategies was not altered. In other words, fitting the model was restricted to affecting the competition between strategies.

This work highlights the role of size and location of rewards for strategy selection. As pointed out by Janssen, Gray & Schoelles (2008) the definition of reward has an important influence on model behavior. Due to the large number of rules being fired in each trial, it is necessary to arrive to an appropriate reward frequency to enable appropriate learning. Rewarding productions for their effectiveness in successfully completing individual commands seems a good criterion; however, in doing this it is important to identify where cognitive effort is made. In the case of FireChief relevant cognitive effort for e.g., placing a new section of barrier, is traced to the time a sequence of actions is initiated prior to the final successful movement being executed, and not just when that final CF command is issued (that is, there is a causal link between the CF command and those actions previously taken).

The process by which a barrier is created is only one amongst many others that occur during a model run. A similar analysis based on utility comparisons can be carried out for other strategies by identifying the rules that govern them. Understanding strategy selection as a consequence of previously learned utility also offers a means to understand more about performance differences. Worst performers reflect a different pattern of utility values in rules used for the creation of the fire barrier, owing to impoverished attention to the dynamic problem solving state and apparent lack of care in issuing commands. Overall the work presented demonstrates that complex dynamic tasks can be fruitfully explored through a cognitive modeling approach. By providing a loose strategy definition the model is able to implement complex patterns of behaviour which in turn are able to successfully stop the fire while replicating many other aspects of the human study data.

Acknowledgments

This research is funded by CONACYT.

References

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review,* 111, (4), 1036-1060.

Brehmer, B., Dörner, D.B. (1993) Experiments with computer-simulated microworlds: Escaping both the narrow straits of the laboratory and the deep blue sea of the field study. *Comp. in Human Behavior*, 9, 171-184.

Cañas, J.J., Antolí, A., Fajardo, I., Salmerón, L. (2005) Cognitive inflexibility and the development and use of strategies for solving complex dynamic problems: effects of different types of training. *Theoretical Issues in Ergonomic Science*, 6 (1) 95-108.

Card, S., Moran, T., Newell, A. (1983) *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum.

Frensch, P.A., Funke, J. (Eds.) (1995) *Complex problem solving: The European Perspective*, Hillsdale, NJ: Lawrence Erlbaum.

Gonzalez, C., Vanyukov, P., Martin, M. (2005) The use of microworlds to study dynamic decision-making. *Computers in Human Behavior,* 21, 273–286

Gray, W.D., Boehm-Davis, D.A. (2000) Milliseconds matter: an introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, 6(4), 322-335.

Janssen, C.P., Gray, W.D., Schoelles, M.J. (2008). How a modeler's conception of rewards influences a model's behavior: Investigating ACT-R 6's Utility Learning Mechanism. *Proceedings of the 15th Annual ACT-R Workshop,* Pittsburgh, PA, p42.

Omodei, M.M., Wearing, A.J. (1995). The FireChief microworld generating program: An illustration of computer simulated microworlds as an experimental paradigm for studying complex decision-making behavior. *Behav. Res. Meth. Instr.& Comp.*, 27, 303–316.

Taatgen, N.A. (2005). Modeling parallelization and speed improvement in skill acquisition: from dual tasks to complex dynamic skills. *Cognitive Science*, 29, 421-455.

Towards Efficiently Supporting Large Symbolic Declarative Memories

Nate Derbinsky (nlderbin@umich.edu)
John E. Laird (laird@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121

Bryan Smith (bryanesmith@gmail.com)
Ann Arbor, MI

Abstract

Efficient access to large declarative memories is one challenge in the development of large-scale cognitive models. Prior work has provided an initial demonstration of declarative retrievals using ACT-R and a relational database. In this paper, we provide extended analysis of the computational challenges involved. We detail data structures and algorithms for an efficient mechanism over a large set of retrievals, as well as for a class of activation bias. We have implemented this work in Soar, and present detailed evaluation on synthetic data as well as the WordNet 3 lexicon.

Keywords: large-scale cognitive modeling; declarative memory; cognitive architecture; Soar.

Introduction

Typical cognitive models have very modest declarative memory (DM) requirements. In these cases, naïve data structures and algorithms, despite inefficiencies, suffice for declarative retrievals. However, prior work (Douglass et al., 2009) has shown that cognitive models of complex tasks require more substantial DMs, such as a large subset of the WordNet lexicon (Miller, 1995), and that existing retrieval mechanisms, such as the ACT-R implementation, do not scale to large DMs. If we are ever going to study human behavior in knowledge-rich, temporally extended tasks, additional research is required on the underlying computational data structures and algorithms that support declarative memory storage and retrieval.

In an effort to efficiently support large declarative memories in ACT-R (Anderson et al., 2004), Douglass et al. developed a DM using the PostgreSQL relational database management system. While their work produced an ACT-R module supporting persistent declarative access to large declarative knowledge stores, there are significant opportunities for extension and improvement. First, while achieving significant empirical performance improvements over the ACT-R retrieval mechanism, the authors do not address the analytical computational profile of the DM retrieval problem, thereby missing, for instance, situations in which even DBMS query optimizers will not support efficient performance. Additionally, their presented evaluation is limited to their target application and DM, and does not include any calculation of chunk activation.

In this paper, we extend that work along many dimensions. First, we contribute an extended analysis of the computational challenges of efficient declarative retrievals. To address many of these problems, we describe system-independent methods for efficient retrieval functionality. Also, while not achieving the full functionality of ACT-R activation, we move towards that goal by formulating and efficiently supporting a simpler class of activation bias.

To evaluate this work, we have implemented a semantic memory system in the Soar cognitive architecture (Laird, 2008). We evaluate the system on a scalable, synthetic data set, as well as the entire WordNet 3 lexicon. For successful retrievals on data sets scaling to millions of declarative chunks, we achieve retrieval times that are two orders of magnitude faster than previously reported results.

A forewarning: much of the presented work delves into the details of data structures, algorithms, and complexity analysis, which are critical for communicating the results of our work to developers of cognitive architectures. However, these details may be of less interest to model developers. We recommend that modelers focus on the problem formulation sections and the empirical evaluation.

Symbolic DM Retrieval Problem

To begin, we develop an abstract problem formulation of symbolic declarative retrievals. To exemplify this formulation, we then map it onto the ACT-R DM.

Problem Formulation

We define a *declarative memory* (DM) as a set of *elements*. A DM element is decomposed into a set of symbolic *augmentations*. For example, consider the following example DM, in which the letters A-D identify elements and lower-case Greek letters represent augmentations:

A: $\{\alpha, \beta, \varepsilon, \phi\}$
B: $\{\alpha, \varepsilon\}$
C: $\{\gamma\}$
D: $\{\gamma, \phi\}$

We define a DM symbolic retrieval *cue* as having a required *positive* component and an optional *negative* component, each of which is expressed as a set of symbols (corresponding to the augmentations of a DM). For instance, consider the following retrieval cue, corresponding to the example DM above, consisting of both positive (+) and negative (–) components: $+\{\alpha, \varepsilon\}, -\{\gamma\}$. Semantically, the positive set specifies augmentations that an element *must* contain, and the negative set those that it must *not* contain.

Given a DM and a cue, we define the *result* of a declarative retrieval to be a single element from the DM, including all augmentations, that satisfies the constraints represented semantically by the cue. Thus, the result of the example cue and the example DM would either be element A or B (with respective augmentation set {α, β, ε, ϕ} or {α, ε}). A retrieval is considered a *success* if there exists a result (as with our example) and a *failure* otherwise.

ACT-R DM

We now compare our symbolic declarative retrieval problem formulation to ACT-R's declarative memory module retrieval interface. We begin with a review of the ACT-R DM and then map it onto our definitions above.

In ACT-R, declarative knowledge is encoded as a set of *chunks*, which are collections of labeled *slots* that have *values*. For example, consider this chunk, representing one of the noun senses of the word "roach" from the WN-LEXICAL interface to WordNet (Emond, 2006):

```
(S-105261088-1 ISA S
      SYNSET-ID          105261088
      W-NUM              1
      WORD               "roach"
      SS-TYPE            "n"
      SENSE-NUMBER       1
      TAG-COUNT          0)
```

To retrieve declarative knowledge, a production rule issues a request to the declarative module by populating the declarative buffer with positive and negative slot-value pairs. These pairs are interpreted as hard constraints that either *must* be met (positive tests) or *must not* be met (negative tests). The DM module also supports non-symbolic tests (<=, >, etc), but we do not consider them.

For example, consider a cue that requests a sense chunk ("ISA S") where the value of the WORD slot is equal to "roach" and the SS-TYPE is not equal to "v" (*verb*):

```
+retrieval>
      ISA          S
      WORD         "roach"
    - SS-TYPE      "v"
```

Given this request, the ACT-R DM module searches the store for matching chunks. If any are found, the module, given default module parameter settings, indicates a successful retrieval and selects randomly amongst the candidates chunks and reconstructs it in the appropriate buffer. The module also supports the use of non-symbolic activation to bias selection amongst candidate chunks, functionality that is used in many cognitive models. We comment on this functionality later in this paper. If no perfect match is found, the default behavior of the DM is to report a retrieval failure. The module also supports the use of customizable partial matching. While some modelers may use this functionality, it makes the retrieval problem strictly harder computationally, and we leave research on an efficient implementation of it to future work.

We now map the ACT-R DM to our abstract formulation. First, without loss of generality, we interpret the chunk type (above, "ISA S") as a slot-value pair (slot label "ISA" and value "S"). Next, since we are considering qualitative matching (equality is defined as symbolic equivalence), each distinct slot-value pair can be equivalently represented as a single, composite symbol (by concatenating the slot label and value with a unique separating character, such as "ISA:S"). Since slot-value pair order is arbitrary, a chunk instance can be equivalently represented as a set of [composite] symbols. In ACT-R, all chunks of a given type must contain values for the same set of slots and a chunk type can only have one slot of a given label; without loss of generality, we eliminate both of these constraints. Given the analysis above, a chunk maps to a declarative memory *element*, and slot-value pairs to *augmentations*.

We apply a similar analysis to DM retrieval requests, with distinct slot-value pairs compressed to a single composite symbol. If we require that equivalent slot-value pairs in chunks and retrieval requests resolve to the same composite symbols, then the set of positive tests form the *positive* cue component and the negative tests the *negative* component.

With this analysis, we claim that the symbolic ACT-R DM retrieval interface is an instance of our problem formulation. Thus, results from our work, though implemented in Soar, extend to ACT-R models, and any other system that can be similarly mapped.

Supporting Efficient Retrievals

In this section, we discuss indexing structures and processes to efficiently support a large class of symbolic DM retrievals, accompanied by a brief computational complexity analysis. We decompose our description into the required *positive* cue component, followed by the *negative*. Prior to getting lost in the weeds of data structures and algorithms, however, let us first consider what is meant by *efficient support* with respect to our problem formulation.

Contextual Meaning of Efficient Support

As a baseline, consider a naïve retrieval mechanism that iterates through the DM, comparing each element to the cue, and returning the first valid result, if one exists. To understand the costs, we define E as the set of elements in a DM, and a as the average number of augmentations per element. Given a cue C, we define P as the positive cue component and N as the negative cue component. Sets surrounded with vertical bars, such as $|E|$, refer to the *cardinality*, or number of items contained in the set.

Assuming no specialized indexing, the memory cost of the baseline mechanism grows with the product of the number of elements and the average augmentation cardinality ($a|E|$). In the worst case, the baseline mechanism must traverse all of this memory for each cue element, and thus the time cost multiplies by the size of the cue ($a|E||C|$). In context of large declarative memories, it is likely that $|E|$ will dominate a and $|C|$, and thus memory and retrieval costs will scale linearly with the number of elements in the DM.

Memory, though not unlimited, is generally considered cheap and plentiful, while time is expensive and limited, and thus our goal is to minimize retrieval time, possibly at the cost of memory. Thus we pose *efficient support* for declarative retrievals as sub-linear in the number of elements in the DM, $|E|$, while remaining linear in memory. We further require that these computational bounds hold in the general case of our problem formulation, supporting a broad variety of DMs and retrieval cues, as opposed to an optimized mechanism for a specific knowledge-base and/or query load. We now present our mechanism, revisiting these requirements for theoretical evaluation.

Positive Cue Component

To review, the positive cue component for symbolic declarative retrievals is a non-empty set of augmentations that a declarative element *must* contain. To assist in our analysis, we define R_p as the elements that contain an augmentation p and, accumulated over all p in P, R to be the bag of candidate elements (which may contain duplicates, if an element contains more than one augmentation, p, in P).

Before presenting our mechanism, we note that this component of the retrieval problem is a constrained form of a *subset* query on *set-values*, which has been widely studied in database and information retrieval (IR) communities (Terrovitis et al., 2006). In its general form, the worst-case time cost is known to be linear in the sum of the number of candidate elements for each positive cue augmentation, $|R|$, though clever indexing methods have shown massive average-case improvements in real-world data.

Indexing Building on this prior work, the primary indexing structure for our mechanism is an inverted table of DM elements, combined with cached frequency statistics. The structure contains a sorted list of each augmentation, p, in the DM, each paired with a sorted list of elements in which they are contained as well as the size of this list, R_p. We note that this structure roughly doubles the size of the store and can be updated very efficiently as the DM changes. Consider the following index over the example DM above:

α (2): [A, B]
β (1): [A]
γ (2): [C, D]
ε (2): [A, B]
ϕ (2): [A, D]

Algorithm To retrieve based only on the positive cue component, we first generate a sorted list, Q, of all augmentations p in P, keyed ascending on R_p, which requires $|P|$ queries on the inverted index. Q represents a specialized query plan, sorted in ascending order of candidate element list size. With the example positive component above, Q is either [α,β] or [β,α] (as $R_\alpha = R_\beta$), and we use the former for the remainder of this analysis.

Next, we pop the first augmentation from Q (α) and retrieve a pointer, w, to the head of the element list in the inverted index (initially referring to the first element, A). Note that since this list is updated incrementally with changes to the DM, we do *not* have to compute this list in response to the query. Iterating over the remaining augmentations in Q ([β]), we verify, using the original DM, that w satisfies all remaining positive constraints. If so, return w and *success*. Otherwise, increment w to point to the next element in the inverted index and retry verification. If no element successfully verifies, the retrieval is a *failure*.

Analysis In the worst case, this retrieval mechanism grows linearly with $|E|$ (as demonstrated later). However, the small amount of indexing and query optimization bounds element iteration to $\min(R_p)$, the set of elements containing the most selective positive query augmentation. Furthermore, we only need to fully examine this list in the *failure* case, which, as we see in the later empirical evaluation, can be achieved in near constant-time queries in many cases.

Negative Cue Component

The negative cue component for symbolic declarative retrievals is an optional set of augmentations that a declarative retrieval must *not* contain.

We have struggled with how to efficiently support this type of constraint given our problem formulation. What makes this component difficult is that given a large DM with a sparse distribution of augmentations, it can be prohibitively expensive to maintain an index of the elements *not* containing an augmentation, analogous to issues surrounding the closed-world assumption and negated conditions in production matching (Doorenbos, 1995).

Initial Integration Currently, we integrate this functionality with the positive cue component above by special-casing negative augmentations. First, $|R'_n|$, the number of candidate elements that do *not* contain a particular augmentation n, equals ($|E|$ - $|R_n|$), the total number of elements less the number of elements that *do* contain the augmentation. This quantity can be computed efficiently and used to order Q with negative augmentations. Second, because we cannot efficiently enumerate R'_n, w is initialized as the head of the list of the first *positive* augmentation in Q. Finally, when verifying a candidate element, we simply invert the result of the set-inclusion query on E.

Analysis Using this approach, our mechanism loses a major performance benefit. This forfeiture arises when there exists an augmentation in the negative component that is more selective than any positive component augmentation, which is probably not uncommon. While we are theoretically able to integrate this functionality, we have neither implemented nor evaluated this work empirically in Soar, and plan to address this deficiency in the future.

Supporting Efficient Activation Bias

A major contribution of the ACT-R DM module to cognitive modeling is the sub-symbolic influence of the current context and prior retrievals as a form of activation bias for declarative retrievals (Anderson et al., 2004). This functionality, however, has been shown to come at a

significant computational cost that does not scale to large declarative memories (Douglass et al., 2009).

While we have not achieved the functionality of all aspects of ACT-R's activation scheme, we have made progress by formulating and efficiently supporting a simpler class of activation bias. In this section, we first extend our problem formulation to include retrieval bias, then define the class of activation update processes we can efficiently support, and discuss how we achieve this functionality.

Problem Formulation Extension

To integrate activation bias in our problem formulation, we extend our definition of a declarative memory element to include a numerical *activation value*, as exemplified below by the numbers in square brackets:

A [1.41]: {α, β, ε, ϕ}
B [1.73]: {α, ε}
C [3.14]: {γ}
D [2.72]: {γ, ϕ}

We refine our previous definition of a retrieval result as an element from the DM, including all augmentations, that satisfies the constraints represented semantically by the cue *and* has the maximal activation value. Given the example cue (+{α, ε}, −{γ}) and this expanded DM, the result is now unambiguously B (and its associated augmentations), as it has a greater activation value than A.

Efficient Activation Bias Updates

The expanded retrieval mechanism described in the next section efficiently incorporates activation. However, just as the DM must support efficient updates to elements and augmentations, so too must it support efficient updates to activation values. In this context, for large DMs, we propose that an activation value update process must be *locally efficient*. An activation update process is locally efficient if it satisfies two properties: (1) the update can affect the activation value of *at most* a constant number of elements and (2) updating the activation value of an element takes time strictly sub-linear in the number of DM elements.

The locally efficient activation update process we implement in Soar is a straightforward mechanism to bias retrievals towards recency. After each successful retrieval, the activation value of the retrieved element is updated to be one greater than the previously largest activation value. This update process is local, as it only changes a single element per retrieval, and it is efficient, as the largest activation value can be cached to avoid any search over E.

In ACT-R, chunk activation includes retrieval history (base-level), current context (spreading), partial matching, and noise. Both the base-level approximation and permanent noise computations appear to be local, so it should be possible to extend our approach to cover those components. However, transient noise, partial matching, and spreading activation appear to be global to the elements of the DM, which suggests significant further theoretical and engineering research are necessary to develop locally efficient mechanisms. For reference, the mechanism in Douglass et al. does not efficiently compute any portion of ACT-R chunk activation, and those components were not included in their empirical evaluations.

Efficient Support

The most direct method of integrating activation values in our efficient algorithm is to sort the candidate list (w) by activation values on demand. This approach, henceforth referred to as Scheme I, suffers from retrieval times that are *always* dependent upon augmentation selectivity, as the candidate list must be fully computed to be sorted.

Another method of integrating activation values, Scheme II, is to maintain, for each augmentation, an element list sorted by activation value. Thus, w is sorted in order of activation, independent of augmentation selectivity. However, the time required for updating activation values is dependent upon the number of different augmentations an element can have (its augmentation cardinality), and for large cardinalities, this cost can be prohibitive.

Our approach to integrating activation values combines these schemes by exploiting an assumption that most elements will have "small" augmentation cardinality. Given this information, we explain how we can extend our implementation to yield efficient retrievals and then we validate our assumption empirically by studying three large, commonly used knowledge bases.

Our Approach. If an element has small augmentation cardinality, Scheme II is efficient, independent of DM size. If few elements must be sorted per retrieval, Scheme I is efficient, independent of element augmentation cardinality. To resolve this tension between augmentation cardinality and element selectivity, we apply these schemes on a per-element basis: we apply Scheme II when an element has small augmentation cardinality, and otherwise apply Scheme I. What we describe here are the data structure modifications and additional processing necessary to efficiently implement this split strategy.

First, we introduce a threshold parameter, t, which represents a *small* value of augmentation cardinality. By default, we integrate activation bias as described in Scheme II above. However, if the augmentation cardinality of a particular element is greater than t, we associate a one-time special "infinity" (∞) activation value with all its augmentations and maintain a separate list associating the element with its activation value, per Scheme I. For instance, if t=3, we would have a list wherein [A=1] and our inverted index would contain the following information:

α (2): [A=∞, B=2]
β (1): [A=∞]
γ (2): [D=4, C=3]
ε (2): [A=∞, B=2]
ϕ (2): [A=∞, D=4]

By default, an update to an element's activation value will involve updating a small number of references ($\leq t$) throughout the inverted index. For elements with augmentation cardinality greater than t, such as A, we need

only update this value once, thereby bounding the update to constant time and addressing the weakness of Scheme II.

During retrieval, as we are populating the list of augmentations, Q, which is sorted by activation level, we may now encounter one or more infinite activations at the head of the list. If so, we perform a lookup for its true activation level and execute insertion sort into a second, special list, Q'. We then merge Q and Q' to form our query plan. Notice that if the size of Q' is small (i.e. few elements have augmentation cardinality greater than t), this process is cheap and independent of augmentation selectivity, the weakness of Scheme I. Thus, if we can select an appropriate value of t, we will achieve efficient activation bias support.

Validation. To validate that our split strategy works well on real data sets, we studied three large, commonly used knowledge bases (KBs): SUMO (Niles et al., 2001), OpenCyc (Lenat, 1995), and WordNet (Miller, 1995). For each KB, we extracted the number of features of each named entity. Each distribution was unimodal and exhibited strong right skew, suggesting that while most elements had a similar feature size, there were rare cases with exceptionally large cardinalities. Then, we sampled from these distributions to form synthetic data sets that were reasonably large (5040 elements) and empirically valid in augmentation cardinality. We then collected empirical retrieval data, summarized in Table 1, showing that for each KB there was a range over the value of t that optimally balanced the performance effects of cue selectivity and augmentation cardinality. For two of the KBs, we could efficiently employ Scheme II above for more than 99% of elements, versus only about 93% for the SUMO data set.

Important components of this analysis for future examination are (1) automatically selecting a value of t for a given DM and (2) tuning this value online for changing DM contents. As to the former, we see in Table 1 that the optimal threshold typically covers greater than 90% of the elements using augmentation cardinality, but that value is not constant across data sets. Further analysis of the KBs may uncover why this is the case and suggest better factors for prediction. As for the latter, we expect that *caching t* in indexing structures will allow the algorithm to adapt in real time, while maintaining efficient retrievals.

Table 1: Optimal Thresholds.

Data Set	Optimal t Range	Element Coverage
SUMO	50 – 70	92.78 – 93.86%
OpenCyc	40 – 60	99.17 – 99.74%
WordNet	20 – 40	99.50 – 99.90%

Evaluation

To evaluate our work, we implemented our data structures and algorithms as the Semantic Memory long-term, symbolic memory system in the Soar cognitive architecture (Laird, 2008). We used version 3 of the SQLite in-process relational database engine to manage the semantic store and all experimental results were run on a 2.8GHz Core 2 Extreme processor with 4GB of RAM.

Our final evaluation spans two data sets: (1) the WordNet 3 lexicon and (2) a scalable synthetic benchmark of our design. WordNet offers a large, ecologically valid knowledge base with which we can compare to previous results in this space (Douglass et al., 2009). Our synthetic dataset offers us the ability to exhaustively benchmark our retrieval mechanism on arbitrarily large DMs.

WordNet

As with Douglass et al., we used the WN-LEXICAL WordNet 3 data conversion (Emond, 2006). The data set has over 820K chunks, which includes over 212K word/sense combinations. Once imported, Soar's semantic store, including all indexing structures, is about 400MB.

Our first experiment was to verify (a) that retrieval time was independent of augmentation selectivity and (b) that the activation bias was processed efficiently in under-specified cues. We performed DM retrievals on 100 randomly chosen, single-augmentation cues, averaged over 10 trials. Retrieval time was 0.1887 msec. each (0.0216 std. deviation).

Our next experiment focused on larger cues. We randomly chose 10 nouns and formed a cue from their full sense description (such as the "roach" example above). Retrieval time was an average of 0.2973 msec. over 10 trials each (0.0108 std. deviation).

Douglass et al. used a derived subset of the WN-LEXICAL dataset, so direct replication of their work is difficult. They reported retrievals of about 40 msec. with cues of 1-4 augmentations on a DM with about 232.5k chunks. Our results show 100x faster retrievals on a comparable set of cues scaling to a 3x larger DM.

Synthetic Data

In addition to running on a known data set, we tested our implementation more exhaustively to measure how it scales with much larger DMs. We developed a scalable, synthetic DM generator and, in Table 2, we list statistics of the data sets we used as they scale with k, the size control parameter:

Table 2: Synthetic Statistics.

k	Elements	Store Size (MB)
7	5,040	3.00
8	40,320	27.81
9	362,880	291.95
10	3,628,800	2048.00

While we have a DM generator, we do not have a model of what are typical cues used to access a DM and how those cues could interact with the performance profile of the DM retrieval mechanism. For instance, we do not know how *selective* the cues are likely to be, meaning how many elements, termed *candidates*, could possibly satisfy any part of the cue. Furthermore, we do not know the proportion of cues that will have no perfect matches. To allow us to test these different interactions, we constructed the DMs so that we can generate cues with independently controlled selectivity. In each KB, there are $k!$ elements and each

element has augmentation cardinality of (k+1). For i = 2 … k, the ith augmentation of an element has selectivity ($k!/i$). The 0th augmentation of each element is shared by all elements and the 1st augmentation is unique.

Selectivity Sweep. Our first question is whether the DM mechanism provides bounded retrievals for under-specified cues, independent of the number of candidate elements. For each distinct augmentation in the DM, we constructed a cue and measured retrieval time. We found nearly constant-time retrievals within each data set, independent of augmentation selectivity, measuring just under 0.4 msec. for k=10.

Cue Sweep. Our next question is whether combinations of augmentations result in complex cues that adversely affect retrieval time. We constructed all possible lengths of cues using all combinations of augmentation selectivity and measured retrieval time. As shown in Figure 1, the only factor affecting retrieval time within a data set was the number of augmentations in the cue ($R^2 \approx 1$), achieving a maximum of about 0.5 msec. for k=10.

Failure Sweep. For our mechanism, retrieval failure is the algorithmic worst-case, as it must examine and fail to verify all candidate elements. We constructed our last experiment to measure retrieval time for cues that fail only after examining significant proportions of the elements in the KB. While our mechanism minimizes the chance of this situation, these results are useful to set an expectation for the unlikely worst-case retrieval time in any given DM. As shown in Figure 2, the number of inspected candidate elements was the only factor affecting retrieval time, independent of the data set. Because the time is linear in the number of candidates, and not the total number of KB elements, our mechanism, for even worst worst-case cues, scales to arbitrarily large data sets when cue augmentations are sufficiently selective.

Conclusions

In this work, we formulate and address the computational challenges involved with supporting efficient symbolic retrievals for the core functionality required in representing and accessing large DMs. We extend the research of Douglass et al., demonstrating two orders of magnitude improvement in retrieval times for comparable functionality on significantly larger data sets. There are still challenges ahead to efficiently support partial match, spreading activation, and other non-local biases for retrieval for large data sets, for which it may be necessary to explore algorithm approximations or massively parallel computation.

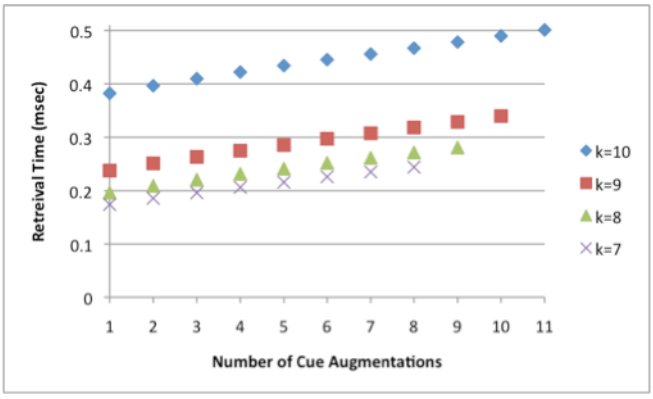


Figure 1: Synthetic cue sweep results.

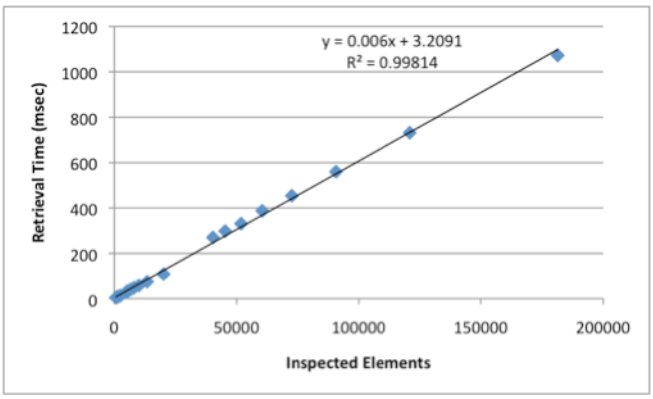


Figure 2: Synthetic failure sweep results.

Acknowledgments

The authors acknowledge the funding support of the Office of Naval Research under grant number N00014-08-1-0099.

References

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An Integrated Theory of the Mind. *Psychological Review* 111, (4). 1036-1060.

Doorenbos, R.B. (1995) *Production Matching for Large Learning Systems*. PhD Thesis, Carnegie Mellon.

Douglass, S., Ball, J., & Rodgers, S. (2009). Large Declarative Memories in ACT-R. *Proc. of the 9th International Conference on Cognitive Modeling.*

Emond, B. (2006). WN-LEXICAL: An ACT-R Module Built from the WordNet Lexical Database. *Proc. of the 7th International Conference on Cognitive Modeling*.

Laird, J.E. (2008). Extending the Soar Cognitive Architecture. *Proc. of the First Conference on Artificial General Intelligence (AGI)*.

Lenat, D. (1995). CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38, (11). 33-38.

Miller, G.A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM* 38, (11). 39-41.

Niles, I., Pease, A. (2001). Towards a Standard Upper Ontology. *Proc. of the Second Conference on Formal Ontology in Information Systems (FOIS)*.

Terrovitis, M., Passas, S., Vassiliadis, P., Sellis, T. (2006). A Combination of Trie-trees and Inverted Files for the Indexing of Set-valued Attributes. *Proc. of the 15th Conference on Information and Knowledge Management (CIKM)*.

Concurrent Knowledge Activation Calculation in Large Declarative Memories

Scott A. Douglass (scott.douglass@mesa.afmc.af.mil)
Christopher W. Myers (christopher.myers2@wpafb.af.mil)
Air Force Research Laboratory, 6030 S. Kent St.
Mesa, AZ 85206 USA

Abstract

To behave effectively and flexibly in complex situations, models specified in cognitive architectures must be able to store and access large amounts of declarative knowledge. However, as research efforts employing cognitive modeling grow in scope and complexity, currently available modeling tools, languages and cognitive architectures are being pushed to their practical limits. This paper describes research looking specifically at how a large declarative memories challenge the current implementation of ACT-R and describes an applied effort to develop an alternative implementation of ACT-R's retrieval process. The alternative exploits concurrency features of the Erlang programming language to extend the practicality of ACT-R's retrieval mechanisms to new levels of scale. The ideas and methods underlying the alternative implementation are general and illustrate how concurrency can accelerate calculation in other architectures struggling to support large associative declarative memories.

Keywords: declarative memory; concurrent activation calculation; semantic networks; ACT-R; Erlang.

Introduction

As research efforts employing cognitive modeling grow in scope and complexity, available modeling tools and languages are being pushed to their practical limits. For example, the implementation of ACT-R within the Lisp programming language may hinder the development of large-scale models due to limitations in declarative storage capacities (Douglass, 2009). If cognitive modeling is to grow in scope and complexity, we must meet the challenges underlying these limits.

An AFRL large-scale cognitive modeling (LSCM) initiative is currently exploring potential solutions to these challenges. The LSCM initiative is committed to integrating well understood mechanisms from cognitive architecture research into new modeling approaches that facilitate model scaling. For example, the empirical strength of ACT-R's declarative system (Anderson, 2007) has motivated us to ensure that the LSCM initiative's solutions preserve ACT-R's declarative memory mechanisms.

LSCM initiative efforts to develop domain-specific modeling languages (DSML-s) supporting increased model scale and persistence involve efforts to increase the scale and persistence of a declarative memory system that mimics ACT-R's. This paper describes recent efforts to retain and scale ACT-R's memory mechanisms in a modeling and simulation framework supporting RML1 (research modeling language), the first DSML developed in the LSCM initiative. RML1 is a generic DSML tailored to the needs of cognitive modeling. RML1 has a hybrid (graphical and textual) syntax, and executes in a runtime environment implemented in the Erlang programming language (Armstrong, 2007).

Modeling with Large Declarative Memories

In the following sections we provide a brief overview of ACT-R and describe how to extend ACT-R's declarative retrieval process by "carving it up at the joints." We conclude this section with a discussion of replicating top-down (i.e., endogenous) and bottom-up (i.e., exogenous) constraints on ACT-R's memory retrieval process.

Brief Overview of ACT-R

ACT-R is a cognitive architecture for developing computational cognitive process models (Anderson, 2007). In ACT-R, cognition revolves around the interaction between a central production system and several modules. There are modules for vision, motor capabilities, memory, storing the model's intentions for completing the task (i.e., the control state), information retrieved from memory, and a module for storing the mental representation of the task at hand (i.e., the problem state). Each module contains one or more buffers that can store one piece of information, or chunk, at a time. Modules are capable of massively parallel computation to obtain chunks. For example, the memory module can retrieve a single chunk from long-term memory and place it into the module's buffer.

Chunks are defined by the modeler to have a particular type, or chunk-type, and a set of key-value pairs. Retrieval in ACT-R is based on a combination of: (1) endogenous influences expressed in retrieval constraints; and (2) exogenous influences originating from chunks in the slots of buffers assigned activation weights by the modeler. When retrieving a chunk, the modeler must specify the type of chunk to retrieve, and all chunks of that chunk-type are candidates for retrieval. All candidates' activations are computed, and the one with the highest activation is retrieved. Chunk activation can be exogenously influenced (i.e., primed) by spreading activation from other modules—any module that contains a chunk as the value in a key-value pair spreads activation to related chunks. As the number of chunks in declarative memory increases, the number of candidates during retrieval also increases. As retrieval candidates increase, retrievals may become slow, and in some instances too slow to support large-scale models that must interact with other system components in real-time.

Increasing Scale by Externalizing Chunk Storage

Our initial efforts to extend the viability of ACT-R's retrieval system to large-scale modeling contexts focused on the storage of chunks outside of ACT-R and Lisp. Database management systems (DBMS) such as PostgreSQL can be effectively used to store a large and persistent set of ACT-R declarative memories (Douglass, et al, 2009). This research determined that services provided by the PostgreSQL DBMS can be integrated into ACT-R via a custom "persistent-DM" module. We found that the persistent-DM module greatly reduced ACT-R's storage burden and significantly increased the practical size of declarative memory sets that could be accessed by cognitive models.

The effectiveness of the persistent-DM module was based on the fact that ACT-R's application of retrieval constraints mimics the behavior of a DBMS executing a SQL query. When the persistent-DM module is employed, requests for instances of a particular chunk-type possessing specific sets of key-value properties are translated into SQL queries and then executed to recover matching chunk instances from an external database. "Outsourcing" the storage and recovery of matching chunks through SQL queries in this way is beneficial because of the capacity of PostgreSQL databases and the effectiveness of indexing in relational databases. Unfortunately, while persistent-DM assumed some of the retrieval burden by efficiently isolating the subset of chunks that had to have their activations re-calculated, the module simply relayed them to ACT-R's default serial activation calculation mechanism.

Carving the Retrieval Process at the Joints

We started the development of RML1's memory system by asking ourselves three questions:

> Q1. How do the equations that explain activation and associative strengths in ACT-R define the fundamental nature of the ACT-R retrieval process?
> Q2. How does the current ACT-R implementation computationally realize the retrieval process?
> Q3. Can the fundamentals of the retrieval process be computationally realized in other ways?

Q1 Human memory is more than an information storage and retrieval system. Likewise, declarative memory in ACT-R is more than just a mechanistic account of information storage and retrieval (Anderson, 2007). Human memory is a part of a system that learns and acts in the world. Human behavior is as flexible as it is because we know lots of things and can use what we know to craft contextually appropriate and effective actions in many different circumstances. It is not enough to know a lot; we also have to be able to quickly cull through all that we know in order to retrieve and apply the right knowledge given our circumstances. The crown jewels of ACT-R's memory system are a set of equations explaining how sub-symbolic calculation, learning, and the utilization of activations and associative strengths enable these critical properties of human memory (see Anderson, et al., 2004 and Anderson, 2007 for detailed descriptions). The equations are presented in Table 1 below so that their details—specifically their indexing of chunks *i* and *j*—can be used to confirm a claim that they describe how sub-symbolic properties related to the activations and associative strengths of *individual chunks* influence the probabilities and time costs of their retrievals. That is, the equations precisely explain how activation is calculated for individual chunks in what can be considered independent calculations.

Table 1: Equations describing chunk activation. The key components of the equations are a single focal chunk indexed as *i* and chunks in context indexed as *j*.

Common Name	Equation
Activation	$A_i = B_i + \sum_{j \in C} W_j S_{ji}$
Base-Level Learning	$B_i = \ln\left(\sum_{k=1}^{n} {t_k}^{-d}\right)$
Attention Weighting	$W_j = {W}/{n}$
Associative Strength	$S_{ji} = \ln\left(prob(i\|j)/prob(i)\right)$
Retrieval Time	$Time = Fe^{-A_i}$
Retrieval Probability	$Prob = 1/\left(1 + e^{-(A_i - t)/s}\right)$

Any declarative memory system adhering to ACT-R's theory of human associative memory must minimally calculate each chunk's activation according to these equations. The equations define a fundamental unit of computation scoped around each chunk in declarative memory and abstract away from how the process of retrieval executes all the chunk activation calculations underlying a single retrieval.

Q2 The current ACT-R implementation (ACT-R 6) sequentially realizes all the chunk activation calculations underlying a single retrieval. Hence, chunk activation calculations occur one after the other as a process, not described in the equations above, searches for and retrieves the chunk with the highest activation. To ensure that this point is clear, the retrieval process in ACT-R will now be summarized.

Retrieval in ACT-R is influenced by bottom-up contextual cues and the application of top-down constraints. Retrievals based on top-down constraints generally proceeds in the following way. An "ISA" property in a ***retrieval request*** is used to isolate type-compatible chunks in declarative memory into a ***candidate set***. Slot value constraints representing additional properties required of a chunk contained in retrieval requests are then used to further reduce the candidate set. The activations of chunks surviving all these top-down constraints are then computed in accordance with the equations above. The chunk meeting all top-down retrieval constraints with the highest activation is returned in the retrieval buffer.

The impact of the serial calculation of activation is illustrated in Figure 1 below. The top and bottom diagrams

in the figure represent two extreme situations. When activation calculations are computed sequentially, the total time cost is roughly equivalent to a per-activation computation time, *t*, multiplied by the number of chunks. When activation calculations are computed concurrently, the total time cost will be slightly more than *t*. Given that the ACT-R activation equations function in the scope of single chunks and in so doing "modularize" the calculation of chunk activations, we argue that the challenge to extend the scale of ACT-R's memory system is really a challenge to maximize the concurrency of chunk activation calculation during retrieval events.

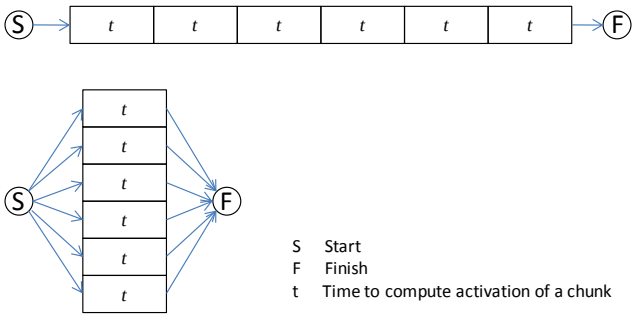


Figure 1: Costs of serial & concurrent activation calculation.

Q3 To find a way to incorporate concurrent activation calculation into the persistent-DM module, we set out to: (1) extend persistent-DM with the ability to compute activations; and (2) develop ways of partitioning databases across multiple PostgreSQL DBMS instances. The first of these challenges was low-hanging fruit; queries to an extended persistent-DM can now include a query capturing top-down retrieval constraints and a representation of context capturing bottom-up sources of activation. Retrievals executed by this version of persistent-DM isolate a sub-set of chunks meeting the top-down constraints, re-compute their activations, and then return the set sorted by activation. Stymied by the second of these challenges, we turned away from trying to find ways of improving query-based retrieval with concurrency and started researching more radical alternatives for realizing massively concurrent retrieval processes. We quickly realized that two problems oppose the development of a memory system utilizing concurrent activation calculation:

> P1. To parallelize activation calculation, one needs a language supporting concurrent computation. What language can do this for us?
> P2. To continue allowing retrievals to be based on top-down retrieval constraints, we have to integrate the processing of top-down information with the process of concurrently computing chunk activations. How can a retrieval process utilizing concurrent activation calculation use top-down information and constraints?

Concurrently Computing Activations in Erlang The semantic anchoring of RML1 is currently realized in a modeling and simulation framework developed using the Erlang programming language (Armstrong, 2007; Cesarini & Thompson, 2009). Erlang is an open-source general-purpose functional programming language developed by Ericsson. Erlang is chiefly used to develop persistent, fault-tolerant, dynamically re-configurable, soft real-time constrained control systems that use large databases. Furthermore, it supports multiple process threads and automatically exploits multi-core and networked computing resources. In Erlang, program components are represented as sets of separate parallel threads. Erlang manages threads through a middleware framework called the Open Telecom Platform (OTP) which simplifies the development and execution of programs consisting of large numbers of concurrent processes. Programs written in Erlang can contain *millions* of concurrent processes (Armstrong, 2007).

RML1's Erlang-based semantic anchoring represents declarative knowledge in OWL-compatible ontologies (Smith, Welty, & McGuiness, 2008) that describe the classes, class properties, object properties, data properties, and instances constituting a domain. Each node in a semantic network is realized as a separate OTP process thread in Erlang. These process threads maintain information about: (a) retrieval parameters; (b) reference histories; (c) last activation level; (d) lists of class, object, and data relations constituting the defining properties of the individual; and (e) lists of object relations the individual serves a range role in. Process threads also receive and respond to messages sent to them by OTP supervisor processes. Each individual process thread is capable of responding to requests to re-compute and report their activation. Activity spreads in RML1 semantic networks as messages are asynchronously exchanged between the process threads constituting their nodes. Since process threads in Erlang execute concurrently, spreading activation achieved through asynchronous message passing and activation re-computing are massively parallel. The retrieval of declarative knowledge from a RML1 semantic network involves all concurrent multi-core computation available.

In order to maximize the parallelization of the activation computation, retrieval in the RML1 declarative memory system is based solely on the spread of activation in semantic networks. At first blush, it is not obvious how something functionally equivalent to an ACT-R top-down "isa" constraint can be obtained through bottom-up spreading activation. The following discussion explains how this is accomplished.

Replicating Top-Down Constraints with Message Filters and Endogenous/Exogenous Message Sources Table A1 (in Appendix) shows how the behavior of top-down retrieval request patterns in ACT-R can be replicated in RML1. Deliberate retrieval constraints introduce top-down network activity into semantic networks as endogenous messages. Endogenous messages introduce network activity into semantic networks but do not convey weighted activation to nodes and therefore do not influence a receiving node's calculation of its activation. Contexts

introduce bottom-up network activity into semantic networks as exogenous messages. Exogenous messages function just like spreading activation in ACT-R; network activity introduced into semantic networks by exogenous sources convey weight and fan and therefore do influence a receiving node's calculation and reporting of its activation. Message filters prevent network activity from being sent to nodes lacking defining properties corresponding to the properties in them. For example, the "k1,v1" message filter in example 3 of Table A1, prevents the endogenous message "type,c1" from passing network activity into nodes lacking the "k1,v1" property.

Retrieval in RML1 proceeds in the following way:

1) An OTP supervisor process sends, in parallel, *"spread network activation"* endogenous and/or exogenous messages to nodes serving domain roles in the relations expressed in the messages that pass any present message filters. For example, in example 1 of Table A1, the OTP supervisor process will send a message to c1. Since "type,c1" is an endogenous message in this circumstance, the message will convey a weight of 0.

2) Nodes receiving "spread network activation" messages relay them, in parallel, to instances serving domain roles in relations with them. In example 1 of Table A1, any node serving a domain role in the "type,c1" relation will receive network activation. As mentioned earlier, individuals maintain lists of the relations they participate in with other individuals. Instances receiving these messages store the weighted activation increments they contain and notify the OTP supervisor that their activation has been influenced by network activity. Because "weights of activation spread" incorporated into endogenous supervisor messages are 0, stored activation increments from endogenous sources force the individual to re-compute their activation but do not increase spreading activation. If, as is the case in example 2 of Table A1, context produced an exogenous message "k2,v2", the "weight of activation spread" incorporated into exogenous supervisor messages would reflect attentional weight and fan.

3) The OTP supervisor process sends, in parallel, *"report your re-computed activation"* messages to nodes that reported contributions to their activations. Individual processes receiving these messages concurrently re-compute their activation. Individuals that received only messages containing 0 weights of activation spread report activation values based solely on changes to their base level activations.

4) Finally, the OTP supervisor posts the defining properties of the node reporting the highest activation to RML1's working memory.

Retrieval in a Large Declarative Memory

To determine the impact of concurrency in RML's retrieval process, a basic comparison study was conducted. In this comparison study, the wall-clock retrieval times of ACT-R and RML1 executing retrievals in large declarative memories were compared. To stress test the declarative systems of ACT-R and RML1, portions of the Moby Thesaurus II synonym database were transcribed into ACT-R's declarative memory and RML1's semantic network. The Moby Thesaurus II contains 30,260 root words that are related to each other by 2,520,264 synonyms. Compound root words were excluded from the comparison study. This exclusion process reduced the number of root words to 24,890. Five different declarative memory sets were created using this reduced set. Sets consisted of proportions of the reduced set of root words and the synonyms relating them. Table 2 below summarizes the properties of these sub-sets, and Figure 2 represents a portion of the smallest of these sub-sets.

Table 2: Properties of the synonym sets used in the comparison study.

Proportion	20%	25%	33%	50%	100%
Synonyms	53,560	77,313	145,073	318,435	1,281,763

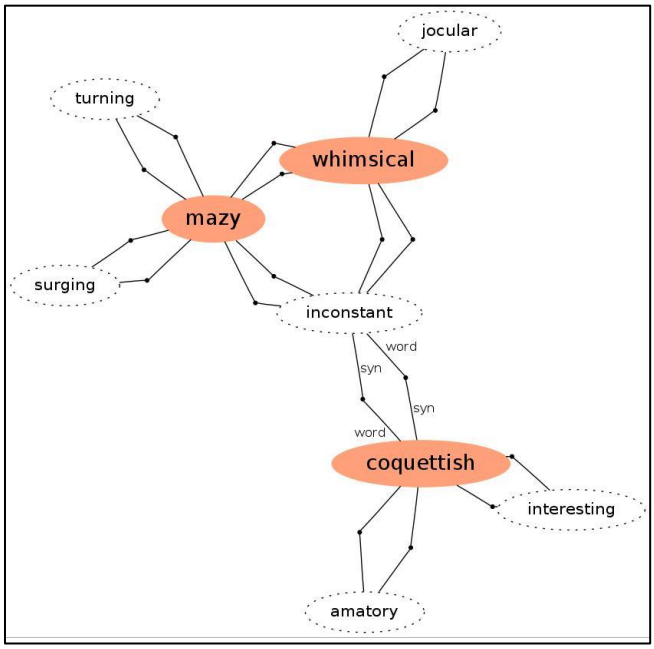


Figure 2: Portion of the Moby II semantic network showing a subset of the root words and synonyms related to the root words "coquettish", "mazy", and "whimsical". 29, 52, and 67 word/syn relations involving coquettish, mazy and whimsical are not shown.

To create a declarative memory in ACT-R, instances of a *root_word* chunk-type were used to represent root words and instances of a *synonym* chunk-type were used to represent word/synonym relationships between root words. Figure 3 shows chunk types and chunk instances that would allow an ACT-R model to represent and process some of the root words and relations displayed in Figure 2. To create an ontology-based semantic network in RML1, *root_word* and *synonym* classes were defined. Object properties necessary to relate words to *syn* in *synonym* instances were also defined. Figure 3 shows the definitions of the *root_word* and *synonym* classes and definitions of employed object and

data properties. Representing these in an ontology allows RML1's runtime environment to search the semantic network and make inferences about arbitrary descriptions or entities lacking class identifiers.

```
(chunk-type root_word name)
(chunk-type synonym word syn)
(add-dm
  ...
  (coquettish ISA root_word name "coquettish")
  (inconstant ISA root_word name "inconstant")
  (flighty    ISA root_word name "flighty")
  (mazy       ISA root_word name "mazy")
  (whimsical  ISA root_word name "whimsical")
  ...
  (syn1 ISA synonym
        word coquettish
        syn flighty)
  (syn2 ISA synonym
        word coquettish
        syn inconstant)
  (syn3 ISA synonym
        word flighty
        syn mazy)
  ...
)
(set-all-base-levels 7 0)
```

```
{class, {root_word, [{subclass_of, thing}]}}.
{class, {synonym, [{subclass_of, relation}]}}.

{object_property,
  {word, [{sub_property_of, base_object_property},
          {domain, synonym}, {range, root_word}]}}.
{object_property,
  {syn, [{sub_property_of, base_object_property},
         {domain, synonym}, {range, root_word}]}}.
{data_property,
  {name, [{sub_property_of, base_data_property},
          {domain, root_word}, {range, string}]}}.

{individual,
  {coquettish, [{type, root_word}], [],
               [{name, "coquettish"}], 7}}.
{individual,
  {inconstant, [{type, root_word}], [],
               [{name, "inconstant"}], 7}}.
{individual,
  {mazy, [{type, root_word}], [],
         [{name, "mazy"}], 7}}.
{individual,
  {whimsical, [{type, root_word}], [],
              [{name, "whimsical"}], 7}}.

{individual,
  {s1, [{type, synonym}],
       [{word, coquettish}, {syn, inconstant}], [], 7}}.
{individual,
  {s2, [{type, synonym}],
       [{word, inconstant}, {syn, coquettish}], [], 7}}.
{individual,
  {s3, [{type, synonym}],
       [{word, mazy}, {syn, whimsical}], [], 7}}.
{individual,
  {s4, [{type, synonym}],
       [{word, whimsical}, {syn, mazy}], [], 7}}.
```

Figure 3: ACT-R (top) and RML1 (bottom) root_words and synonyms matching some of the Figure 2 information. Note the object and data property specifications in RML1 .

Equipment

A Dell Precision T7500 was used in the comparison study. The Dell's physical configuration included 2 quad core Intel 3.33Ghz Xeon (W5590) CPUs and 48 GiB of RAM. The computer's software configuration included the openSUSE 11.2 Linux-based OS, SBCL 1.0.35 running ACT-R6 r845, and Erlang R13B04.

Procedures

Context-sensitive retrievals of chunks from the sub-sets of the Moby Thesaurus II were carried out in ACT-R and RML1 using the request patterns and context representation shown in Table A2. Real-time costs of executing retrievals in ACT-R were measured by: (1) placing three chunks corresponding to root word chunks into slots of a goal chunk representing retrieval context; (2) initiating a retrieval request corresponding to the "+retrieval> isa synomym" request pattern; and (3) measuring elapsed system time until the retrieval process returned a chunk. The real-time costs of executing retrievals in RML1 were measured by: (1) distributing messages from endogenous and exogenous message sources that passed through message filters into the semantic network; and (2) measuring elapsed time until the OTP supervisor process managing the retrieval determined the network node with the highest activation.

Results

The same retrieval parameters were used in both systems: maximum associative strength was set to 5.0, the base-level constant was set to 0, and the base-level learning rate was set to 0.5. All chunks were initialized with 7 references.

Retrievals executed through ACT-R and RML1 returned the same synonym chunks, computed equivalent chunk activations, and retrieval latencies. The use of the "isa synonym" constraint in the ACT-R retrieval pattern required that the activations of all synonym chunks be calculated before the retrieval process could finish. Treating "type, synonym" as if it were from an endogenous message in the RML1 retrieval process correspondingly lead to all synonym instances re-computing and reporting their activations. Table 3 summarizes the results of the comparison study.

Table 3: ACT-R and RML1 performance. Times (seconds) represent average wall-clock time to execute 10 retrievals.

	20%	25%	33%	50%	100%
Synonyms	53,560	77,313	145,073	318,435	1,281,763
ACT-R	3.22	6.00	18.63	86.39	NA
RML1	0.44	0.64	1.21	2.65	10.90

The most important thing to notice in Table 3 is that while ACT-R (SBCL) performance time is increasing at a rate faster than the increase in chunks, RML1 (Erlang) is essentially scaling linearly. Added concurrency from additional processor cores will further improve the relative performance of RML1.

Conclusion

The declarative system underneath RML1 discussed in this paper is interesting because it: (1) does not depend on a top-

down retrieval process that functions like a query against a relational database followed by activation calculation; (2) is capable of producing behavior that is functionally indistinguishable from ACT-R; (3) exploits concurrency in Erlang and therefore scales nearly linearly; (4) is part of the runtime environment supporting RML1, the first DSML researched and developed by the LSCM initiative. If cognitive modeling is to successfully grow in scope and complexity, it must find effective ways of meeting the challenges associated with maintaining and using large declarative memories. RML1's declarative system illustrates how concurrent knowledge activation calculation in large declarative memories can be technically realized and is therefore progress towards meeting LSCM challenges associated with modeling human memory on a large scale.

Acknowledgements

This research was funded through a seedling grant from the Chief Scientist of the 711th Human Performance Wing of the Air Force Research Laboratory awarded to Drs. Douglass & Myers, and through AFOSR grant #10RH05COR awarded to Dr. Douglass.

References

Anderson, J. R. (2007). How Can the Human Mind Occur in the Physical Universe? Oxford: OUP

Anderson, J. R., Bothell, D., Douglass, S. A., Byrne, M. D., Lebiere, C., Qin, Y., et al. (2004). An integrated theory of the mind. Psychological review, 111(4), 1036–1060.

Armstrong, J. (2007). Programming Erlang: Software for a Concurrent World. Raleigh: The Pragmatic Bookshelf.

Cesarini, F., & Thompson, S. (2009). Erlang Programming. O'Reilly Media, Inc.

Douglass, S. A., Ball, J., T., & Rogers, S. (2009). Large declarative memories in ACT-R. In A. Howes, D. Peebles, R. Cooper (Eds.), 9th International Conference on Cognitive Modeling – ICCM2009, Manchester, UK.

Smith, M. K., Welty, C., & McGuiness, D. L. (2008). OWL Web Ontology Language Guide. W3C.

Appendix

Table A1. Examples of how query-based retrieval behavior in ACT-R can be replicated using message passing in RML1 semantic networks. The character "*" is used in messages to represent a wildcard that is free to match against any relation. The "*" is necessary because contextual priming in ACT-R is insensitive to the key component of the key/value pairs in context chunks. Notice that examples 3 and 4 yield the same retrieval behavior while using the "type,c1" and "k1,v1" messages in different ways. Since it is likely to be the case that the fan of v1 is less than the fan of c1, treating the "k1,v1" message as endogenous will greatly reduce the spread of network activity and therefore expedite retrieval.

Example	ACT-R		RML1		
				Message Sources	
	Retrieval Request	Context	Message Filters	*Exogenous*	*Endogenous*
1	+retrieval> isa c1				type,c1
2	+retrieval> isa c1	isa c2 k2 v2		k2|*,v2	type,c1
3	+retrieval> isa c1 k1 v1		k1,v1		type,c1
4	+retrieval> isa c1 k1 v1		type,c1		k1,v1
5	+retrieval> isa c1 k1 v1	isa c2 k2 v2	type,c1	k2|*,v2	k1,v1

Table A2. ACT-R retrieval requests and contexts & RML1 message filters and message sources employed in the declarative memory system comparison study. To ensure the fairness of the comparison, all exogenous messages conveying activation due to contextual priming had to be insensitive to relation (they all had to use "*"). Parenthesized numbers indicate fan.

Example	ACT-R		RML1		
				Message Sources	
	Retrieval Request	Context	Message Filters	*Exogenous*	*Endogenous*
1	+retrieval> ISA synonym	=goal> c1 whimsical(73) c2 mazy (60) c3 coquettis(31)	type,synonym	*,whimsical *,mazy *,coquettish	type,synonym
2	+retrieval> ISA synonym	=goal> c1 vexing (20) c2 heavy (249) c3 operose (42)	type,synonym	*,vexing *,heavy *,operose	type,synonym
3	+retrieval> ISA synonym	=goal> c1 entangle (63) c2 stare (65) c3 woo (33)	type,synonym	*,entangle *,stare *,woo	type,synonym

Dimensions of Leader-in-Context Models

Ceyhun Eksin (ceksin@seas.upenn.edu), Barry G. Silverman (BaSil@seas.upenn.edu),
David Pietrocola (dpiet@seas.upenn.edu), Rui Kang (ruikang@seas.upenn.edu)
Ackoff Collaboratory for Advancement of the Systems Approach (ACASA),
Department of Electrical and Systems Engineering,
University of Pennsylvania, Philadelphia, PA 19107 USA

Abstract

In this study, we explore dimensions of comparison amongst complex agent-based models. Specifically, we look at holistic models of leaders-in-context. We focus our analysis on alternative models of the same phenomenon, that of the rise and fall of two corporations, respectively. The models were built by students with introductory training on the methodology and modeling framework. We extract dimensions and examine good vs. bad modeling behavior. We divide these dimensions into ones that are related to modeling leader context and ones that are related to leader profiling. We use these dimensions to address how to facilitate modeling alternative theories across a broad range of topics and how to compare resulting models.

Introduction

Studying the "traits of the great man" sitting atop a traditional organizational hierarchy is no longer sufficient to understand leadership. This approach like other schools of leadership study (e.g, cognitive, networks, cultural, etc.) tends to be singularly focused. Lichtenstein et al. (2006) and Avolio (2007) argue that leadership research today must be holistic and synthetic (see Silverman et al., 2007). Synthetic leadership theory underlines the necessity to integrate various theories on cognition, traits, and situational contingencies (e.g. context, culture, social networks, etc.) to provide a picture of the whole. This is what a leader encounters in the real world in the contexts he or she must manage. Hazy (2007) highlights the importance of hybrid computer modeling techniques to support experimentation on the holistic perspective. Hazy (2007) claims that hybrid models that include various techniques are likely to become abundant with increasing adoption of a holistic look at leadership. We feel that the most suitable approaches to a holistic perspective are socio-cognitive agent-based models where leader traits and affective reasoning in context are richly defined as endogenous parts of a complex system.

The reasons to model leaders are 1) to try and understand mechanisms that cause them to think under varying circumstances, and 2) once that is known and validated, to use these models to explore what-if possibilities, alternative courses of actions, and how to influence them.

In the social sciences, there are no set principles, no one-theory-fits-all situations. So ideally one wants to try different theories and factors. The modeling architecture must support this testing of theories allowing users to shift in different ideas and see if they better explain what is making leaders function as they do.

As a result, we want greater ability to plug theories and sub-models in and out of the framework. The holistic leader-in-context movement means that modelers must use a framework that covers many dimensions (cognitive, personality, cultural, socio-economic, etc.). How to model this breadth of topics while simultaneously permitting ease of trying different models is *one question* we explore here. In particular, this study examines how novice modelers (student trainees) can use a socio-cognitive architecture to plug in differing models of a leader-in-context.

The second author has developed a socio-cognitive modeling framework called PMFserv (Silverman et al., 2007) that provides a model of an agent's cognitive-affective state and reasoning abilities that is applied to profile the traits, cognitions, and social reasoning of agents alone and in groups. PMFServ utilizes cognitive appraisal theory where each agent goes through an observe, orient, decide, and act (OODA) loop (Boyd, 1995). For each agent, PMFserv operates its perception and runs its personality/value system to determine individual action decisions to carry out the resulting and emergent behaviors. The PMFserv framework also permits the modeling of groups, economic behavior, and socio-cultural factors. Hence, the framework is a reasonable candidate for analyzing leader behavior within varying contexts.

It is possible to build different versions of computational models when systems are complex. Yet, when these computational models are built, there are no existing common dimensions on which to evaluate them. A *second question* of interest is, "How can we compare models that claim to model the same phenomenon?" Recently, comparison amongst cognitive models has been studied by Lebiere et al. (2009) and John (2010). Lebiere et al. take on the task to compare cognitive models built by different individuals or teams that use different approaches. The hardest part of their approach is to come up with common grounds for comparison amongst different approaches. John explores the reduction in variation between novice modelers via guidance of CogTool (John, 2009). John first identifies common mistakes of modelers and then compares the variation between modelers with and without the tool support.

In this study, we take a different approach. We establish dimensions for comparison of a certain type of holistic leader models built by novice modelers (students) using a

common framework, i.e. PMFserv's existing socio-cognitive appraisal framework. Specifically, the framework allows modelers to define: 1) Context, i.e. how leaders perceive the world; 2) Decision making behavior, i.e. how leaders process information flowing in and determine actions accordingly; and 3) World behavior, i.e. how the world gets affected by these individual actions. In this study, we define world behavior beforehand and restrict modelers to focus on the first two parts to replicate a given scenario. Next, we specify dimensions of comparison in leader-in-context models by identifying the differences amongst models. Unlike John (2010), there are no errors in modeling but there is good or bad modeling. Finally, we use these dimensions to specify desired features for models of leader-in-context.

The next section summarizes the PMFserv framework focusing on cognitive appraisal theory. The methodology section describes the dimensions of comparison and outlines the good and bad practices of leader-in-context modeling. The subsequent section describes the specifics of the scenario and task given to modelers. The results section analyzes the differences amongst the models based on the dimensions explored. The last section concludes with discussion and related future work.

Cognitive Appraisal within PMFServ

The Performance Moderator Function Server (PMFserv) was designed by Silverman et al. (2006) as a modular system and socio-cognitive modeling framework for implementing and evaluating performance moderator functions (PMFs). PMFserv operates what is sometimes known as an observe, orient, decide, and act (OODA) loop. PMFserv agents utilize cognitive appraisal theory to help them cope with these contexts. This involves a perception system, a values system, an emotion model and a decision module.

Perception Module

Perception of agents and objects around each agent determine the context. The perception is based on "affordances" (Cornwell, 2003) which is a form of distributing perceptions so that an agent's knowledge of the world is marked up onto the perceived objects, instead of the perceiving agents. Each entity in the world, agents, objects, groups, organizations etc., applies perception rules to determine how it should be perceived by each perceiving agent. Hence, each agent can perceive the same entity differently based on these rules. For example, a bike might afford the actions 'ride', or 'walk alongside' to an agent if it knows how to ride a bike but it might only afford the 'walk alongside' action to another agent that does not know how to ride a bike. In this case, the mark-up rules that reveal actions depend on properties of the perceiving agent. An example of a company that is marked up for such perceptions is given in Figure 1. Each gray box represents one way the company can be perceived. Each element of the grid is called a perceptual type (p-type). These p-types are not mutually exclusive.

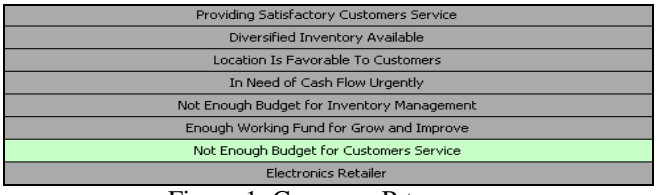


Figure 1: Company P-types

Modelers establish appropriate context via rules on a p-type. For example, a CEO might see that 'Not Enough Budget for Customer Service' is active and be afforded actions 'Decrease Customer Service' or 'Fire employees' whereas this context is not valid for a customer agent. Hence, p-type rules might require that the perceiving agent works at the company or that it is the CEO of the particular company. The set of active p-types determine the actions afforded to perceiving agents. We define the parameters that affect the p-type rules as input parameters.

Activations and Value System

An afforded action provides activations to those taking that action. These activations are fixed and irrespective of the agent that is afforded the action. Agents assess the activations of each action against their values system to compute the utility of taking that action. By comparing utility of all alternative actions, agents complete the primary appraisal, i.e. how alternative contexts affect their personal well-being, emotions etc. They then select the action that maximizes their utility.

For this to work, PMFserv requires every agent to have goals, standards, and preferences (GSP) trees filled out. GSP trees are multi-attribute value structures where each tree node is weighted with Bayesian importance weights. Within a simulation, each agent has the same tree structure, i.e. nodes are the same but the weights differ among agents. The assignment of node weights determines the traits of a certain agent. Figure 2 provides an example of a simple GSP tree structure for a company CEO.

In order to determine a specific agent's importance weights, modelers utilize differential diagnosis (Bharathy, 2006) and analytical hierarchy process (AHP). This provides a systematic and valid methodology for assessing the weight of each node to effectively

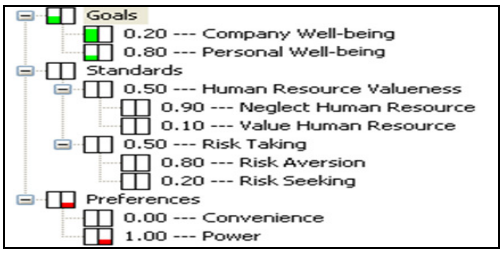


Figure 2: An example GSP tree

profile the agents and settings of interest. Using differential diagnosis, modelers collect and assess relevant evidence to attribute behavior. In this process, each hypothesis corresponds to a node in the GSP tree, i.e. behavior or traits. The output is organized in tabular form called an 'evidence table' with additional attributes such as reliability, frequency of occurrence, and relevance. Evidence tables allow one to consider all competing hypotheses at once and rank them accordingly by assigning confirmation scores to each hypothesis. Figure 3 provides a shortened example of an evidence table. The table shows that the first evidence relates to the nodes 'Risk Aversion' and 'Risk Seeking'. From the evidence table, the weights are estimated through the AHP process by pair-wise comparison of their confidence index.

	Standards					
Evidence (Ei) ✓ = Evidence supports, or can be made to support, the hypothesis x = Evidence rejects the hypothesis	Reliability (Ri)	Relevance Or (Diagnostic Value)	H1: Risking Aversion	H2: Risk Seeking	H3: Value Human Resource	H4: Negalect Human Resource
Circuit City did not change business mode since 1990s.	1	1	✓	x		
Circuit City laid off 3400 top salesmen in 2007.	1	1			x	✓
Circuit City had problems with inventory management.	0.9	1				
Circuit City CEO Schoonover received $1.4 million in salary and bonuses in fiscal 2006,	1	1				

Figure 3: Evidence Table

Emotion model

This is the module that calculates how each agent is likely to feel from taking an action based on arousals, i.e. combining activations and values system (GSP tree). Each afforded action has an activation mapping on the GSP trees. The activation mapping is a collection of success/failure levels on a set of GSP nodes. For a simple example, an activation mapping on the values system (in Figure 2) of the action 'Decrease Customer Service' is given in Figure 4. It shows that the result activates two nodes positively, 'personal well-being' and 'neglect human resource', and one node negatively, 'company well-being'. The set of emotions that each agent generates from taking an action is determined by the sum of their activations weighted by node weights. Thus an importance-weighted values system results in differing emotions being generated within the same context by different personalities. For mathematical underpinnings of the implemented model, see Silverman et al. (2006).

Actions	Results	Node	Op	Success Level	Op	Failure Level
Decrease customer service	100% - Decrease customer service	Company Well-being	=	0	+	0.1000
		Neglect Human Resource	+	0.1000	=	0
		Personal Well-bing	+	0.1000	=	0

Figure 4: Activation mapping for action 'Decrease Customer Service'

Decision Module

The decision model receives information from the value-driven emotion model and implements utility theory to select actions. A decision in PMFserv is a choice made by an agent when choosing between alternative afforded actions. A decision-making algorithm runs to select the decision with the highest subjective expected utility. Subjective expected utility (SEU) for each decision is determined by appraising all possible emotions that will be generated if the decision is taken by that agent. The decision taken is called an action. An action may generate effects on the environment – actor, target and other entities – based on its result. These result effects are called action bindings. We will refer to parameters that these action bindings affect as output parameters. Figure 5 gives an example of an action binding for the action 'Decrease customer service'. The output parameters are 'capital' and 'customerServiceQuality' of the target of the action.

Result Effects (Action Bindings)

Standard Action Binding | Custom Action Binding | Custom Perception Binding

	Target	Property	Operation	Variable Type	Value
1	target	capital	+	Number	1
2	target	customerServiceQuality	+	Number	-1
3					

Figure 5: Action Binding rule table

Methodology

In this section, we introduce the dimensions of comparison amongst the models. These dimensions also highlight good versus bad modeling behavior. We divide the dimensions of comparison into two major clusters: 1) Dimensions related to modeling leader-context interactions, i.e. how context, afforded actions, leader responses and its effects on the world are modeled, and 2) Dimensions related to modeling leader personality, i.e. how agent value systems are constructed.

Dimensions Related to Modeling Leader-Context Interactions

These are the dimensions that provide feedback on how conditions that lead to leader actions (p-types and afforded actions) and effects of leaders actions on the world are modeled. It is possible to further divide these dimensions into two: context richness and action-result balance.

Context Richness It refers to the depth of the model with respect to leader perception. Within the PMFserv framework, context is determined by p-types. If one wants to have finer levels of granularity in perception modeling, it is necessary to increase the number of p-types. This will enable one to pin down the reasons for events in finer detail. However, increasing only the number of p-types is not always sufficient. Number of input parameters that affect the perception rules often needs to be correlated with number of p-types. If number of affecting parameters is much smaller than number of p-types then there is a strong indication of overloading parameters with multiple meanings which in return means p-types are not clearly defined. This will often require accurate estimation of these parameters. In short, the context which affords

actions to agents should be clearly defined so that agents consider the correct set of actions at the right set of circumstances.

Action-Result balance It refers to the relations between actions and parameters that are affected by the results of those actions. One must consider all aspects of taking that action when one is defining an action's effects on the world. Often, results of actions come with trade-offs. The modeler has to reflect these trade-offs via output parameters.

Dimensions Related to Modeling Leader Traits

While the previous cluster of dimensions may reflect on how leaders perceive and how their actions affect the world, it is really the personality that determines how leaders vary from one another within the same context. The dimensions in this section refer to assessment of leader personality models.

Quality and Quantity of Evidence Organizing information from otherwise diverse or amalgamated sources is critical to the success of the modeling activities. Although differential diagnosis and AHP process minimizes subjectivity and biases within the process, the validity of results depend on the quality and number of pieces of evidence. Quality of evidence refers to the relevance and reliability of evidence. A modeler should try to obtain reliable evidence that is relevant to the story. Additionally, one would want to increase the number of pieces of quality evidence attributed to each node.

Coverage in Tree to Activation Mapping Activation mappings on GSP trees are used for emotion calculations which in return get used in decision-making. If a node does not get covered by an activation mapping from any of the actions then that node will be idle throughout the simulation. In other words, it will not have any effect on the decision-making calculations. Modelers need to make sure that each node gets mapped to an activation by at least one action.

Sensitivity Analysis If change in a parameter value causes significant changes to the main outcome of the model then it means that the model is sensitive to that parameter. This would require that parameter to be estimated with higher precision. The behavior of a validated cognitive model should ideally be fairly robust with respect to tweaking changes on a single personality trait. Within the PMFserv framework, sensitivity to a node indicates that for certain key actions, activation mappings affect mainly that node. The modeler has to be aware of this sensitivity and carefully use techniques discussed in the previous section and try to find additional evidence for more accurate determination of node weights.

Task and Scenario

After approximately 25 hours of framework and methodology training, students were given strict guidelines to come up with a working model that replicates a given scenario as one part of their coursework requirement. The class consisted of junior and senior Systems Science and Engineering (SSE) students with the exception of one Economics major. Most students are also completing a double major or a minor degree in our business school. Students were given two weeks to complete their assignment and they had support from experienced model builders. The students worked in groups of four or five. They were given a benchmark model that required certain tasks to be completed to fully function. Each individual had to model an agent by picking a theory of behavior and reflect this theory onto a values system for their agent. The set of agents to model were given to them. Team members had to decide on which agent each student would model. Each group had to come up with important parameters, contexts, afforded actions, activations and results of taking those actions for the set of agents. The benchmark model contained a set of rules that govern the dynamics of the world and groups were fully aware of how the world would function. Lastly, they were required to replicate scenario outputs within their model.

Specifically, students were given the story of Circuit City (CC) going bankrupt and Best Buy (BB) excelling. They were given a news article that overviews the story. Additionally, they were encouraged to do their own research on the story and their specific agents. The minimum required set of agents included Circuit City CEO, Best Buy CEO, and two or three (depending on group size) types of consumers. Further, two companies were modeled and placed under the control of the respective CEO. Each student focused on profiling a single agent. The decisions of consumers were predefined within the world dynamics as 'Shop from Best Buy' or 'Shop from Circuit City'. The teams were required to maintain these two actions and were not allowed to add new actions for the consumers. CEO agents did not have any predefined actions, thus the teams had to work on all parts of the OODA loop for those agents.

Results

This section provides examples of dimensions discussed in the methodology section from student models. We provide a summary of the models in Table 1. Out of the eight teams, six teams were able to create a model that replicated the desired output behavior, i.e. CC's fall and BB's rise. Two teams (Model_5 and Model_8) were not able to complete their model within the given time frame. In Table 1, we provide a collection of p-types from each model (except Model_3) that afford actions only to CEOs (BB CEO or CC CEO). P-type rules, action binding code, and a portion of the p-types have been omitted due to space restrictions.

The first set of examples relate to context richness. Teams had a hard time balancing affordances, actions and activations to create meaningful context. In Model_4, CEO gets afforded actions such as 'Acquire New

Business' and 'Expand to Prime Locations' via the p-type 'Business Expansion Possible'. These actions have no clear context because they get afforded to the CEO all the time. In fact, in Model_4, CEO gets afforded all the actions (listed in Table 1) at all times, i.e. the only requirement is for the agent to be CEO of that company. In Model_6, CEO agents are afforded the actions 'Increase customer service' and 'Increase number of new products' as long as companies have positive capital. Similarly in Model_2, p-types 'BB Customer Service Savings Available', 'BB Improvement', 'BB Price' are all active if parameter 'customerServiceQuality' is greater than zero. In other words, the CEO does not distinguish between these p-types. Additionally, Model_3 uses the parameter set 'Inventory' and 'capital' to define five different p-types indicating possible overloading. However, this group used different values of 'Inventory' and 'Capital' as thresholds to trigger these five p-types. Unlike the previous examples this kind of rule format is acceptable to define varying context but not desired as it relies on fine tuning of these parameters. Finally, we refer to Model_7 as an example model that defines context appropriately. Model_7 uses differing combinations of input parameters to define various contexts.

A majority of the modelers were able to capture the trade-offs of actions inside the action bindings. One obvious violation was in Model_6. 'Decrease number of new products' only has an effect on the parameter 'amount of products'. One would imagine that this action would have direct and immediate positive effect on the 'capital' of the company. As an example, in Model_4 the action 'Expand to Prime Locations' increases 'Accessibility Rating' but at the same time it hurts company's 'capital'.

In order to construct the GSP structure for their agents of interest, students were asked to collect evidence that could help to profile their agents. The number of evidence that students organized ranged from 8 to 25. Students were encouraged but not required to use reliability or relevance scores for their evidence tables. Most of the students utilized a low-medium-high scale and rated their evidence as medium or highly reliable. On the average, a team had 11 nodes for Goals, Standards, and Preferences. Hence, there was an average of 33 nodes in total on average. This meant that roughly 33 hypotheses existed within an evidence table. Students cross-compare these hypotheses with each piece of evidence. Furthermore, students were able to provide evidence for each node. Given the limited time the modelers had, we consider this an acceptable effort.

Each individual had to incorporate a theory and justify how their theory reflects on the values system (GSP structure and node weights) of their agents. Students utilized theories such as individual theory, marketing theory, Maslow's theory on the hierarchy of needs, economic buyer theory, utility theory, agency theory, consumer behavior theory, etc. GSP node names (hypotheses) were formed by these theories. Each team came up with a common GSP structure but each individual had to incorporate a different theory for their agent. The key here was to look at whether that theory was confirmed for their individual agent via pieces of evidence. The majority were able to justify that their individual theory applied to their agent.

As a final requirement for their coursework, students were required to come up with an if-then hypothesis based on a change in personality trait of the agents that each person was responsible for modeling. An example if-then hypothesis is: "If 'Save Money' node weight of CC CEO is reduced then CC would remain in business for a longer time." In short, students related a macro-level metric to a change in micro-level values. Out of the 12 students who modeled either CC or BB CEO for their teams, only four (only one of them was BB CEO) reported that their model was sensitive to the changes that were made on the GSP trait they analyzed. All reported that the change in behavior was in parallel with their initial expectations, i.e. their if-then statement. The rest reported that their model is relatively insensitive to their parameter changes and the hypothesis is disconfirmed.

Concluding Remarks

This study placed a benchmark model of two firms, CC and BB, in the hands of student trainees and challenged them to research and build alternative models of leaders in context. The leaders they built had to account for the cognitive and personality variables that may have caused the decline of CC and the success of BB. Further, these leader models had to operate in a holistic environment and cope with many types of networks and social dynamics that are spawned at run time: ego-networks, economic networks, transaction networks, and so on.

Six teams successfully completed the assignment. They researched alternative theories and built differing models of leaders-in-context. Thus they illustrate answers to question number one – can users build and plug-in alternative models covering the breadth of socio-cognitive dimensions dictated by the modern leader-in-context theory. Their results also address the answer to the second question and give us ample fodder to begin to understand how to compare different models of the same phenomenon.

We explored dimensions for comparison of leader-in-context models. The first set of dimensions concentrated desired features on modeling parts of the OODA loop and the second set concentrated on leader personality modeling and its effects on the model. We extracted these dimensions from working student models by focusing on differences between models. We realize that this variability between models is likely to reduce when models are built by experienced modelers. A future research direction is to analyze whether these dimensions remain salient and sufficient for assessment of expert models.

Table 1: Summary of student models (Input parameters, p-types, afforded actions, and output parameters)

Models	Input Parameters	P-Types that afford actions to CEOs	Afforded Actions	Output Parameters
Model_2	customerServiceQuality	BB Customer Service Savings Available	1. Decrease customer service	capital, customerServiceQuality
	customerServiceQuality	BB Improvement	1. Increase customer service	capital, customerServiceQuality
	customerServiceQuality	BB Price	1. Reduce Price	Price
Model_4		Business Expansion Possible	1. Acquire New Business 2. Expand to Prime Locations	capital, product Range capital, accessibilityRating
		Employee Quality	1. Allow Flexible Scheduling 2. Train Employees	capital, customerServiceQuality capital, customerServiceQuality
		Marketing Improvements Possible	1. Implement Centrizing	capital, customerServiceQuality, brandImage
		Payroll Increases Possible	1. Increase Top Management Salaries	capital, productRange, brandImage
		Payroll Savings Possible	1. Decrease Salesman Salaries	capital, customerServiceQuality, brandImage, accessibilityRating
Model_6	Capital	Improvements available	1. Increase customer service	capital, customerServiceQuality
	amountOfProducts	Not spending money on new products	1. Decrease number of new products	amountOfProducts
	Capital	Products available	1. Increase number of new products	amountOfProducts
Model_7	location	Liquidate Stores	1. Close 100 Stores	capital, location
	location, capital	Locations Available	1. Open 100 New Stores	capital, location
	newTechnology, capital	New Technology Available	1. Invest in New Technology	capital, newTechnology
	promotions, capital	Promotion Available	1. Hold Promotion	capital, promotions
	brandNames	Savings Available by Canceling Partnership	1. Cancel Partnership	capital, brandNames
	promotions	Savings Available by Cancelling Promotion	1. Cancel Promotion	capital, promotions
	websiteQuality	Web Savings Available	1. Decrease Online Presence	capital, websiteQuality
	websiteQuality, capital	Website Improvement Available	1. Improve Online Presence	capital, websiteQuality

Model comparison is fairly straightforward in traditional mathematical models that are tractable. However, cognitive agent-based models are hard to compare because each model includes a diverse library of models that have different assumptions and perspectives. This is the main reason why knowledge produced by different complex social models does not accumulate. In fact, every modeler prefers to start from scratch to build their own model which they can build confidence in. Furthermore, even under strict guidelines, modelers still come up with a whole variety of models.

Throughout the paper, we use dimensions instead of metrics of comparison to distinguish the fact that these dimensions of comparison are not quantified. In the future, we hope to be able to quantify these dimensions into metrics for assessment of socio-cognitive leader models.

References

Avolio, B. J. (2007). Promoting More Integrative Strategies for LeadershipTheory-Building. *American Psychologist, 62,* 25-33.

Bharathy, G. (2006). *Agent based Human Behavior Modeling: A Knowledge Engineering based Systems Methodology for Integrating Social Science Frameworks for Modeling agents with Cognition, Personality & Culture*. dissertation, U. of Pennsylvania.

Boyd, J. (1995). *The Essence of Winning and Losing.*

Cornwell, J., O'Brien, K., Silverman, B.G., Toth, J.(2003). Affordance Theory for Improving the Rapid Generation, Composability, and Reusability of Synthetic Agents and Objects. *12th Conf on BRIMS,* SISO.

Hazy, J. K. (2007). Computer models of leadership: Foundation for a new discipline or meaningless diversion? *The Leadership Quarterly, 18,* 391-410.

Lebiere, C., Gonzalez, C., Dutt, V., & Warwick W. (2009) Predicting cognitive performance in openended dynamic tasks a modeling comparison challenge. In A. Howes, D. Peebles, R. Cooper (Eds.), *9th ICCM, Manchester, UK.*

John, B. E. (2009). *CogTool user guide version 1.1.*, Carnegie Mellon University, Pittsburgh, PA.

John, B. E. (2010). Reducing the Variability between Novice Modelers: Results of a Tool for Human Performance Modeling Produced through Human-Centered Design, *BRIMS, Charleston, SC*

Lichtenstein, B. B., & Uhl-Bien, M., & Marion, R. (2006). Complexity Leadership Theory: An interactive perspective on leading in complex adaptive systems. *Emergence: Complexity and Organization,* 8, 2–12.

Silverman, B., G., Johns, M., Cornwell, J., & O'Brien, K. (2006). Human Behavior Models for Agents in Simulators and Games: Part I – Enabling Science with PMFserv, *Presence, 15,* 139-162.

Silverman, B. G., Bharathy, G., K., Nye, B.,Eidelson, R. (2007). Modeling Factions for 'Effects Based Operations': Part I – Leader and Follower Behaviors. *Journal of Computational & Mathematical Organization Theory, 13*, 379-406.

Improving the Reading Rate of Double-R-Language

Mary Freiman[1] and Jerry Ball[2]
L3 Communications[1]
Air Force Research Laboratory[2]
Mary.Freiman@mesa.afmc.af.mil, Jerry.Ball@mesa.afmc.af.mil

Abstract

This paper describes changes to a model of reading comprehension to improve its reading rate and bring it into closer alignment with human reading rates. The broader context of the research is development of language capable synthetic teammates that can be integrated into team training simulations. To use synthetic teammates in team training without detriment, we believe the synthetic teammates must be both functional and cognitively plausible. By functional, we mean that the synthetic teammate operates in real time, performs the task, and handles the range of linguistic inputs that are encountered. By cognitively plausible, we mean that the synthetic teammate adheres to well established cognitive constraints on human language processing—including the incremental and interactive processing of language at human reading rates. Achieving human reading rates in a cognitively plausible and functional model of reading comprehension is a research challenge that has not been met to date.

Keywords: human language processing, reading rate, synthetic teammate, functional, cognitively plausible

Introduction

We are developing a model of reading comprehension called Double-R-Language (Ball, 2007; Ball, Heiberg & Silber, 2007). Double-R stands for Referential and Relational—two key dimensions of meaning that get grammatically encoded in English. The initial application of the reading model is development of a synthetic pilot for use in a three-person UAV simulation. The synthetic pilot flies the simulated UAV from a ground control station and will eventually communicate with a human navigator and photographer in the completion of reconnaissance missions. A prototype system has been developed (Ball, et al., 2009) using the ACT-R Cognitive Architecture (Anderson, 2007). The synthetic pilot prototype communicates with lightweight agent versions of the navigator and photographer developed outside ACT-R.

The prototype communicates with the navigator and photographer using text chat and must be capable of reading and comprehending the messages it receives from them. The reading comprehension model is capable of incrementally processing linguistic inputs and generating linguistic representations of referential and relational meaning. These linguistic representations are interactively mapped into a non-linguistic representation of the objects and situations referred to in the linguistic input. The non-linguistic representation—called the situation model (cf. Zwann & Radvansky, 1998)—drives the task behavior of the synthetic pilot and determines when to communicate with the other teammates to acquire needed information.

A significant challenge for the reading comprehension abilities of the model is input variability. A corpus of text chat communications that was collected in an experiment involving human subjects and the UAV simulation is full of variability in the form of linguistic input (see Table 1). For competent readers, misspelled words activate the intended lexical items because they contain many of the same letters and trigrams (Perea & Lupker, 2003). Hence, key requirements of the reading model include the ability to handle misspellings in input; the ability to separate perceptually conjoined units (e.g. separating punctuation from words as in "He went.", but not "etc."; separating words lacking spaces as in "yougo" for "you go"); and the ability to recognize multi-word expressions (e.g. "speed up") and multi-unit words (e.g. "a priori", "h-area").

Table 1. Messages seen during a UAV simulation

MESSAGE:	VARIANT:
i need to be beloe 3000 for f area	i; beloe; f area
effective radiu	radiu
any requirements for altitde/speed?	altitde
can yougo faster yet or is it stll 200	yougo; stll

To satisfy these requirements, the model includes a word recognition subcomponent that uses ACT-R's spreading activation mechanism to influence lexical item retrieval. The subcomponent maps orthographic input directly into DM representations without recourse to phonetic processing, although a phonetic mapping is not precluded.

The model uses the spreading activation mechanism of ACT-R to retrieve words from the lexicon that are not an exact match to the input. Letters and trigrams in the input spread activation to the words containing those letters and trigrams in the mental lexicon. These processes and encodings are based on the Interactive Activation model of word recognition (McClelland & Rumelhart, 1981), with the addition of trigrams based on "letter triples" (Seidenberg & McClelland, 1989). The subcomponent is embedded in the reading comprehension model as a whole; the effects of context and previous activation levels are taken into consideration when encoding each individual word (Freiman & Ball, 2008). The reading model also includes a verification stage to check the retrieved lexical item against the perceptual input. The verification stage aligns with the Activation-Verification model of Paap et al. (1982). It splits concatenated words in the input (e.g. "yougo") to match the

retrieved word (e.g. "you"), leaving a residual (e.g. "go") for subsequent processing. If the retrieved lexical item is not a sufficiently close match to the input, the model treats the input as an unknown word.

Even without considering the mapping of the linguistic representations into the situation model, the previous version of the reading model was much slower than humans in both cognitive processing time and real time performance. Adult readers read at a rate of 200-300 words per minute (Taylor, 1965; Carver 1973a; Carver 1973b). The average reading rate of the model—prior to the introduction of the changes described in this paper—was 96 words per minute (cognitive processing time), making it impossible to match the model's performance against human performance. Since we are interested in building a model of reading comprehension that is cognitively plausible as well as functional, this presents a real challenge. The prior reading model read slowly for several reasons: 1) it required multiple declarative memory (DM) retrieval requests per word; 2) it lacked the ability to read units of language larger than the word; and, 3) it built complex linguistic representations necessitating the execution of multiple productions. In addition, the model relied on parallel spreading activation to retrieve lexical items, which is computationally expensive for large DMs on serial hardware.

It is important to distinguish between reading rate as measured by the real time functional performance of the model and the rate as measured by the cognitive processing time. ACT-R provides support for measuring cognitive processing time—how long it would take a human to perform some cognitive process. Execution of a single production in ACT-R takes 50ms of cognitive processing time; plus, the time it takes to retrieve a chunk from DM depends on the activation of the chunk and can be measured. Typical ACT-R models with small DMs are capable of executing much faster than real time while measuring cognitive processing time. However, large DMs tax the computational resources of serial hardware and can lead to models that run slower than real time or not at all (cf. Douglass, Ball & Rodgers, 2009). Although it is important to distinguish cognitive processing considerations from real time considerations, these considerations are intertwined. For example, reducing cognitive processing time by eliminating retrievals also reduces the computation of parallel spreading activation, speeding up the real time performance of the model. For each of the shortcomings listed above, one or more remedies is described below and its impact on cognitive and real time processing is considered.

Reducing retrievals

When the model retrieves chunks from DM, the ACT-R Declarative Memory module calculates the activation across all chunks that match the retrieval template, selecting the most highly activated chunk for retrieval. The retrieval template provides hard constraints on memory retrieval—which are difficult to justify from the perspective of cognitive plausibility. Only chunks exactly matching the retrieval template are eligible for retrieval. The spreading activation mechanism provides more cognitively plausible soft constraints on retrieval. Chunks may be activated which are not an exact match to current input or context. For cognitive plausibility, we prefer ACT-R's spreading activation based soft constraint retrieval mechanism, minimizing the use of hard constraints in the retrieval template. For example, we do not want to use a hard constraint exact match to the input which would preclude retrieval of a word which is not an exact match (e.g. "altitde" should retrieve "altitude"). However, use of hard constraints reduces the amount of computation significantly by eliminating non-matching DM elements from the spreading activation computation.

Instead of relying on hard constraint retrievals to reduce the amount of computation, we have pursued the more cognitively plausible alternative of reducing the number of retrievals. An example of this is discussed next.

Combining Word Form and Part of Speech Chunks

In the previous version of the model, there was a word-form chunk for each word that encoded the graphical form of the word, including the letters and trigrams in the word (e.g. speed-wf), and part of speech chunks that encoded the various parts of speech of the word (e.g. speed-noun and speed-verb). The performance of the reading model has been improved significantly by collapsing the word form and part of speech chunks into a single word-form-pos chunk (e.g. speed-wf-noun, speed-wf-verb). Now, a single retrieval is required to determine the part of speech of a linguistic input. Since the production which initiates a retrieval takes 50ms to execute, by combining the word form and part of speech chunks for each lexical item, 50ms plus the retrieval time were saved per word.

From a representational perspective, combining the word form and part of speech chunks is not ideal. The word-form-pos chunks combine two distinct types of information (i.e. graphical vs. grammatical) which are better kept separate. A better solution would retain separate chunks, but support retrieval of part of speech chunks given the linguistic input. This could be achieved via multi-level activation spread if the linguistic input activated a word form chunk which in turn activated related part of speech chunks. Unfortunately, ACT-R does not support multi-level activation spread, although its predecessor ACT* (Anderson, 1983) did. It should be noted that single level parallel spreading activation is already computationally expensive for large DMs. Supporting multi-level spreading activation would add an additional multiple to the computation for each level.

Expanding the Perceptual Span

By default, ACT-R's vision module splits input text into perceptual spans at spaces and punctuation. The module even splits at word internal punctuation, so "*ACT-R"* becomes *"ACT" "-" "R"*, requiring three movements of

attention to read. This behavior was changed to a more plausible splitting of the input text, thereby reducing the number of retrievals per input. Words with internal punctuation are no longer split up and retrieved separately.

The width of the perceptual span is now determined dynamically, based on the length of the first word ($word_n$) in the perceptual span. The boundary of $word_n$ is determined by the first space. If $word_n$ is greater than twelve letters in length, it takes up the entire length of the perceptual span. If $word_n$ is fewer than twelve letters in length, up to six letters of the next word ($word_{n+1}$) can also be seen in the perceptual span. No more than twelve letters are contained in the perceptual span.

The size of the revised perceptual span is deliberately conservative, so that even though three very short words (e.g. "out of the") could be perceived at a single attention fixation, the model never retrieves information for more than two words. There is a great deal of evidence that the perceptual span of adult readers is about 14-15 letters to the right of fixation (DenBuurman et al., 1981; McConkie & Rayner, 1975; Rayner, 1986). We implemented a span of up to twelve letters, with the greatest amount of activation spreading from the first few letters of the span and decreasing toward the end of the span. As a result, incorrect letters at the beginning of words are more detrimental to correct retrieval than misplaced letters later in the word. Activation spreads from the letters, trigrams, and length of the first word ($word_n$). If there is more than one word in the perceptual span, $word_{n+1}$ spreads activation from its trigrams. The section of the perceptual span containing $word_n$ is roughly equivalent to the fovea; the perceptual span at $word_{n+1}$ is roughly equivalent to the parafovea.

The revised perceptual span is generally larger than ACT-R's default span. Just as for adult readers, information to the right of fixation is obtained when the next word is predictable from the preceding text (Balota, Pollatsek, & Rayner 1985). Again, we were deliberately conservative in determining how much information could be perceived from $word_{n+1}$. Our intent was not to model in high fidelity the perceptual span in reading, or movements of attention in reading; movement of attention is not our primary focus. We merely wanted to make the vision module more serviceable to our language comprehension model, and more faithful to human perceptual spans in the process.

An example of the reduction in reading time can be seen in the phrase "take us to h-area". Previously, ACT-R's vision module would chop the input into seven parts:

"take" "us" "to" "h" "-" "area"

The model would retrieve each part from DM, integrate it into a linguistic construction, and then move on to the next word. The last three sections of the input would need to go through additional processing for the model to recombine them into a single word. Reading the entire sentence took 2.8 seconds. If ACT-R does not chop up the input at spaces and punctuation, the same phrase takes only 1.74 seconds to read. In the next section, the advantage of the expanded perceptual span for processing multi-word expressions is described.

Multi-Word Expressions

To facilitate reading and word recognition we have modified the ACT-R architecture and the reading model to better interpret multi-word expression (i.e. lexical units containing spaces). By not splitting the perceptual input at all spaces, multi-word expressions and multi-unit words can be retrieved as a single chunk (e.g., "of course" and "a priori"). To accommodate multi-word expressions we modified our lexical chunks in DM to reduce the number of retrievals necessary per word. Multi-word expressions are treated in much the same way as singleton words. Many multi-word expressions are not syntactically alterable units and need not be parsed (Sag et al. 2002), so the model treats them as "words-with-spaces".

An important side effect of the new perceptual span mechanism is that it also increases the reading rate of the model in the process. Since the perceptual span can cross spaces as well as punctuation, multi-word units like "to go", "want to", and "believe in" can be recognized as a single unit and processed in a single attention fixation. This capability is really the key to getting Double R-Language to approach adult human reading speed.

Before the multi-word expression capability was implemented, the phrase "we need to go" took 1.99 seconds for the model to process. After the perceptual span was expanded, the model reads the same phrase in 1.79 seconds. In this phrase "to go" is treated as a single unit, since it is an infinitive verb. There is one fewer retrieval, and the infinitive can be integrated into the phrase as a whole without having to recombine "to" and "go". Whenever there are multi-word units, the model now saves time in retrievals and processing. There is no difference in the time it takes to process other sorts of words. In addition, multi-word expressions are less ambiguous than individual words. "To" in isolation is very ambiguous, whereas "to go" is much less ambiguous.

Linguistic Representations

The reading model incrementally processes the linguistic input and builds a representation of referential and relational meaning that is mapped into the situation model. The building of linguistic representations is driven by the execution of productions which retrieve or construct linguistic elements and integrate them into the evolving representation. It takes more productions and retrievals to build complicated linguistic structures. In an effort to reduce the number of productions and retrievals that are required, we investigated how linguistic representations could be simplified or reduced. Our current approach attempts to build the minimal structure needed to represent the linguistic input, but must support more complex structures when they are needed.

Retrieving object referring expressions

Determiners are words that project definiteness and (sometimes) number information to nominals (Ball, 2010). In the reading model, nominals are called object referring expressions (ORE) to emphasize their referential (referring expression) and relational (object) functions. Determiners include the articles "a", "an", and "the", as well as the negative "no", demonstrative pronouns "this", "those", etc. Linguists have long known that the determiner "the" is the most commonly used word in the English language (cf., Zipf, 1932); other determiners are nearly as common. As the most commonly used words, determiners are likely to be highly proceduralized or simplified in their use (Zipf, 1949). Therefore we concentrated on consolidating the processes associated with determiners.

Previously, the model identified a word as a determiner, then executed a production which projected an ORE. The determiner was integrated as the specifier of the ORE. Given that determiners are used so regularly and frequently, it seems likely that there is an ORE in DM associated with each determiner that can be retrieved without first identifying the part of speech of the word. By retrieving the associated ORE rather than first identifying the word as a determiner, the processing of determiners becomes more proceduralized, faster, and more cognitively plausible. Where separate, general productions were required to retrieve the part of speech, followed by projection of an ORE if it's a determiner, now a single specialized production projects an ORE directly from determiners. Although we manually created this specialized production, we would prefer that the model learn how to compile such productions automatically.

Reducing structure in nominal heads

Retrieval or projection of an ORE by a determiner establishes the expectation for a head to occur. In the previous version of the model, when a word following the determiner was identified as a noun, a subsequent production projected an object head and integrated the object head as the head of the ORE (Figure 1). Projection of the object head from the noun supported the integration of pre- and post-head modifiers (e.g. the post-head modifier "on the runway" in "the airplane on the runway"). When a post-head modifier occurred, it could be integrated into the object head in the post-head modifier slot. However, in the absence of a post-head modifier, projection of an object head is unnecessary and the noun could be integrated as the head of the ORE. The current version of the model adopts the simpler approach, integrating the noun as the head of the ORE (Figure 2). The tree diagrams below were generated by the previous and current versions of the model and show the contrast between the two approaches for the linguistic input "the restriction" (the pre- and post-head modifier slots in the object head are not displayed):

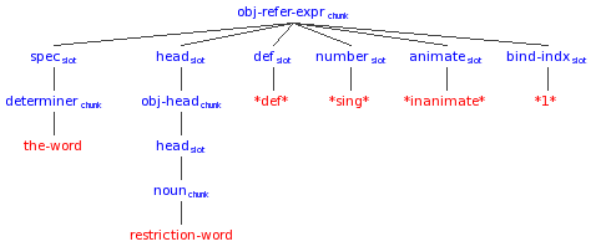


Figure 1. Original nominal structure (including a determiner, projected ORE and object head)

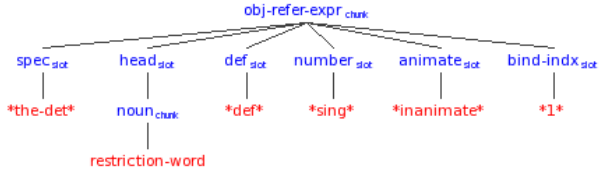


Figure 2. Reduced nominal structure (the retrieved determiner ORE and no object head)

But what happens when a post-head modifier occurs, or when the pre-head modifier slot turns out to be needed? In the processing of the input "the altitude restriction", when "altitude" is processed it is integrated as the head of the nominal projected from "the". When "restriction" is subsequently processed there is no expectation for its occurrence. The previous version of the model projected an object head, so "restriction" was accommodated by shifting "altitude" into the pre-head modifier slot so that "restriction" could be integrated as the head. In the current version, we have adopted a similar strategy. In parallel with the integration of "altitude" as the head of the ORE, an object head is constructed in which "altitude" is the head. This object head is available if needed to support subsequent processing. When "restriction" is processed, the object head overrides "altitude" as the head of the ORE and "altitude" is shifted into the pre-head modifier slot so that "restriction" can be integrated as the head (Figure 3). Note that the object head is projected in parallel to facilitate processing. A single production integrates the object head as the head of the ORE, shifts "altitude" to the pre-head modifier slot and integrates "restriction" as the head. It takes no more time to process "restriction" than in the previous version of the model, but it does require parallel projection of the object head.

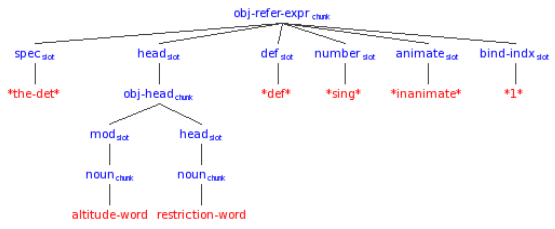


Figure 3. Accommodating "restriction"

Real Time Processing & Spreading Activation

Cognitive time is the time it takes the productions and retrievals in ACT-R to happen, with each production taking a fixed amount of time. When a production fires, 50ms of cognitive time elapses, so having many productions firing for the processing of each word takes up a great deal of cognitive time. Retrievals also take cognitive time—chunks with high activation are retrieved more quickly than chunks with low activation.

Retrievals take *real time* to calculate the activation of all eligible chunks. Real time is the wall clock time that passes while the computer executes the model. When a retrieval request is not very specific, for example, specifying only the chunk-type, then the activation for all chunks of that type must be calculated before the most highly activated chunk can be selected. There are thousands of chunks of type WORD, so when the chunk-type WORD is the only retrieval specification, thousands of activation calculations must be performed before a chunk is retrieved. While this is a parallel process in the brain, it is a serial process for a microprocessor. Since the language model specifies only the chunk type, and relies on spreading activation to retrieve words, thousands of calculations bring the real time reading rate down to 53words per minute (wpm).

Disjunctive Retrieval

One way to retrieve chunks faster in real time is to impose stronger hard constraints on the retrieval. Instead of a weak chunk-type specification that matches thousands of chunks, a strong constraint that matches only a limited set of chunks can be specified. For example, the model could try to retrieve an exact match to input text form, which might only match a single chunk in DM. However, imposing such constraints makes the model less flexible and less cognitively plausible. If the model relies on a hard constraint to match the input form against words in DM, variants cannot be read. Even a hard constraint on just the first letter means that words where the first letter is transposed with the second, or in any other way misplaced, cannot be read by the model.

The model needs the flexibility of a soft-constraint retrieval with the real time speed of a strong hard-constraint retrieval. In order to achieve this affect, we implemented a disjunctive retrieval mechanism. Using an ACT-R function called get-chunk, the model checks DM for the largest constituent of the perceptual span. If it does not find that constituent, it chops the perceptual span at the last punctuation mark or space. If that constituent is not found, it chops at the second to last punctuation mark or space, and so on. If an entire word does not match at any point, a simple soft constraint is attempted.

For example, if the input sentence is "og to h-area", we want the model to be able to retrieve GO for "og" (see Table 2). The get-chunk function is used to try to find chunks that correspond to smaller and smaller parts of the visual input. If at any point the function finds what it is looking for, the model uses that specification to make the retrieval. Get-chunk is a simple search function into a hash table—it is not computationally expensive, and it functions outside of the cognitive processes of ACT-R, so it does not take any cognitive time.

Table 2. Perceptual span contains "og to h-area"

SEARCH FOR:	RESULT:	RETRIEVAL REQUEST:	RESULT:
og to h-area	NIL	--	--
og to h-	NIL	--	--
og to h	NIL	--	--
og to	NIL	--	--
og	NIL	--	--
--	--	chunk-type WORD	GO-word

Using the disjunctive retrieval, the average reading rate for the model is 249wpm in real time. The cognitive time is unaffected, and the model runs with disjunctive retrieval are identical to the model runs using a pure soft-constraint. The results of retrieval requests are identical. Since the two retrieval methods are equivalent in ACT-R, the disjunctive retrieval is acceptable as a way to make our model fast enough to be functional in real time while we try to catch up in cognitive time.

Conclusions

Although we have not yet succeeded in achieving human reading rates, we have improved the reading rate of the Double-R-Language significantly. The initial version of the model read at a rate of about 96wpm, far from our goal of 200-300wpm, the average reading rate of adults. The model now reads at an average rate of 143wpm in cognitive time, and 249wpm in real time. This rate is the average, achieved while reading a text of just under 2,100 words, without counting punctuation as separate words.

The perceptual span is closer in size to that of human readers than previously. The expanded perceptual span allows for the expansion of the model's lexicon to include multi-word units, as well as speeding up the reading rate. An additional advantage of multi-word units is that they are less ambiguous than words in compositional phrases.

The model was improved by simplifying various linguistic constructions. Parallel constructions allow for simplified nominal heads, and object referring expressions in declarative memory allow the model to avoid constructing object referring expressions whenever determiners are encountered. We posit that the simplified representations are not only more expedient, but more cognitively plausible as well. Avoiding unnecessary constructions in the model is more likely to track the efficiency of human language use.

Ultimately, we believe that achieving human level reading rates will require a capability to recognize multi-word units that exceed a single perceptual span. Recognition of a linguistic unit as forming a part of a larger linguistic unit

across perceptual spans should minimize the amount of higher level processing required to integrate the recognized unit into the evolving representation and speed up the reading rate, allowing the model to approach adult human reading rates.

Although reading rate is important, the language comprehension model is being developed to model the full range of linguistic processes of a competent adult reader, rather than just modeling the reading rate. It is our hope that any improvements we make in the reading rate of our model will be accompanied by improvements in the models accuracy and cognitive plausibility.

References

Anderson, J. R. (1983). *The Architecture of Cognition*. NY: Harvard.

Anderson, J. R. (2007). *How Can the Human Mind occur in the Physical Universe?* NY: Oxford.

Ball, J. (2007a). Construction-driven language processing. *Proceedings of the 2nd European Cognitive Science Conference*, 722-727. NY: LEA

Ball, J. (2007b). A Bi-Polar Theory of Nominal and Clause Structure and Function. *Annual Review of Cognitive Linguistics*, 27-54. Amsterdam: John Benjamins

Ball, J. (2010). Projecting grammatical feature in nominals: cognitive processing theory & computational implementation. *Proceedings of the 19th Behavior Representation in Modeling & Simulation Conference.*

Ball, J., Heiberg, A. & Silber, R. (2007). Toward a large-scale model of language comprehension in ACT-R 6. *Proceedings of the 8th International Conference on Cognitive Modeling*, pp. 163-168.

Ball, J., Myers, C. W., Heiberg, A., Cooke, N. J., Matessa, M., & Freiman, M. (2009). The Synthetic Teammate Project. *Proceedings of the 18th Annual Conference on Behavior Representation in Modeling and Simulation*. Sundance, UT.

Balota, D. A., Pollatsek, A., & Rayner, K. (1985). The interaction of contextual constraints and parafoveal visual information in reading. *Cognitive Psychology*, *17*, 364-390.

Carver, R.P. (1973a). Understanding, information processing and learning from prose materials. *Journal of Educational Psychology*, 64, 76-84.

Carver, R. P. (1973b). Effect of increasing the rate of speech presentation upon comprehension. *Journal of Educational Psychology*, *65*, 118-126.

Cook, V. J, & Newson, M. (1996). *Chomsky's Universal Grammar*. Malden, MA: Blackwell Publishers.

Douglass, S., Ball, J. & Rodgers, S. (2009). Large declarative memories in ACT-R. *Proceedings of the 9th International Conference on Cognitive Modeling*.

DenBuurman, R., Boersma, T., & Gerrissen, J. F. (1981). Eye movements and the perceptual span in reading. *Reading Research Quarterly*, *16*, 227-235.

Freiman, M. & Ball, J. (2008). Computational cognitive modeling of reading comprehension at the word level. *Proceedings of the 38th Western Conference on Linguistics*, 34-45. Davis, CA: University of California, Davis.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception. Part I. An account of basic findings. *Psychological Review*, *88*, 375–407.

McConkie, G. W., & Rayner, K. (1975). The span of the effective stimulus during a fixation in reading. *Perception & Psychophysics*, *17*. 578-586.

Paap, K., Newsome, S., McDonald, J., & Schvaneveldt, R. (1982). An activation-verification model of letter and word recognition: The Word-Superiority Effect. *Psychological Review*, *89*, 573-594.

Perea, M., & Lupker, S. J. (2003). Does jugde activate COURT? Transposed-letter similarity effects in masked associative priming. *Memory and Cognition*, *31*, 829–841.

Rayner, K. (1986). Eye movements and the perceptual span in beginning and skilled readers. *Journal of Experimental Child Psychology*, *41*, 211-236.

Sag, I., Baldwin, T., Bond, F., Copestake, A., & Flickinger, D. (2002). Multiword expressions: A pain in the neck for NLP. *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*.

Seidenberg, M. S., & McClelland, J. L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review*, *96*, No. 4, 523-568.

Taylor, S. E. (1965). Eye movements while reading: Facts and fallacies. *American Educational Research Journal*, 2, 187-202.

Zipf, George Kingsley (1932). *Selected Studies of the Principle of Relative Frequency in Language.* Cambridge, MA: Harvard University Press.

Zipf, G. K. (1949)**.** *Human behavior and the principle of least effort: An introduction to human ecology*. Cambridge, MA: Addison-Wesley, 1949..

Zwann, R. and Radvansky, G. (1998). Situation models in language comprehension and memory. *Psychological Bulletin*, *123*(2), 162-185.

An Algorithm for Self-Motivated Hierarchical Sequence Learning

Olivier L. Georgeon (olg1@psu.edu)
Jonathan H. Morgan (jhm5001@psu.edu)
Frank E. Ritter (frank.ritter@psu.edu)

The College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802

Abstract

This work demonstrates a mechanism that autonomously organizes an agent's sequential behavior. The behavior organization is driven by pre-defined values associated with primitive behavioral patterns. The agent learns increasingly elaborated behaviors through its interactions with its environment. These learned behaviors are gradually organized in a hierarchy that reflects how the agent exploits the hierarchical regularities afforded by the environment. To an observer, the agent thus appears to exhibit basic self-motivated, sensible, and learning behavior to fulfill its inborn predilections. As such, this work illustrates Piaget's theories of early-stage developmental learning.

Keywords: Developmental learning; cognitive architectures; situated cognition; computer simulation.

Introduction

We report the implementation of an agent that autonomously engages in a process of hierarchical organization of behavioral schemes as it interacts with its environment. This mechanism moves towards taking on developmental constraints as Newell (1990, p. 459+) called for, and generates high-level and long-term individual differences in representation and behavior that arise from lower level behavior.

This implementation also refers to an "emergentist" and a constructivist hypothesis of cognition. According to these hypotheses, an observer can attribute cognitive phenomena (such as *knowing*, *feeling,* or *having motivations*) to the agent while observing its activity, provided that the agent's behavior can appropriately organize itself. These hypotheses have often been related to Heidegger's philosophy of mind, e.g., cited by Sun (2004). Additionally, these hypotheses correspond to work featuring constructivist epistemologies (Le Moigne, 1995; Piaget, 1937), situated cognition (Suchman, 1987), and embodied cognition (Wilson, 2002).

We describe the agent as self-motivated because it does not seek to solve a problem pre-defined by the modeler, nor does it learns from a reward that is given when reaching a pre-defined goal. Rather, the agent learns to efficiently enact its inborn predilections by exploiting regularities it finds through its activity. As such, the implementation constitutes a model of agents exhibiting intrinsic motivation, pragmatic and evolutionist learning, as well as sensible behavior.

To situate the technical approach in the field of artificial intelligence, we can refer to Newell and Simon's (1975) physical symbol hypothesis. We subscribe to the hypothesis's weak sense. We are using computation to generate intelligent behavior. We, however, do not subscribe to the hypothesis's strong sense, in that we are not implementing symbolic computation based on symbols to which we would pre-attribute a denotation. Instead, we will discuss how knowledge appears to emerge (to an external observer) from the agent's activity, and how the agent seems to make sense of the knowledge because it is grounded in the agent's activity (Harnad, 1990).

Although we did not follow a symbolic computational modeling approach, we have, nevertheless, implemented this model in a cognitive architecture, namely Soar 9. We chose Soar because it has proven efficient for implementing mechanisms for behavior organization. In particular, we found Soar 9's mechanisms for graph querying and operator selection based on valued preferences very helpful.

Knowledge representation

The agent's behavioral patterns are represented using two kinds of objects: *schemas* and *acts*. We use the term schema in its Piagetian (1937) sense, meaning a behavioral pattern or sensorimotor pattern. An *act* is a notion specific to our work that refers to a schema's enaction. By schema's enaction, we mean the association of a schema with the feedback the agent receives when enacting the schema. Concretely, an act associates a schema with a binary feedback status: *succeed (S)* or *fail (F)*. Hence, each schema is associated with at most two acts: its *failing* act and its *succeeding* act. Schemas and acts are organized in a hierarchy as shown in Figure 1.

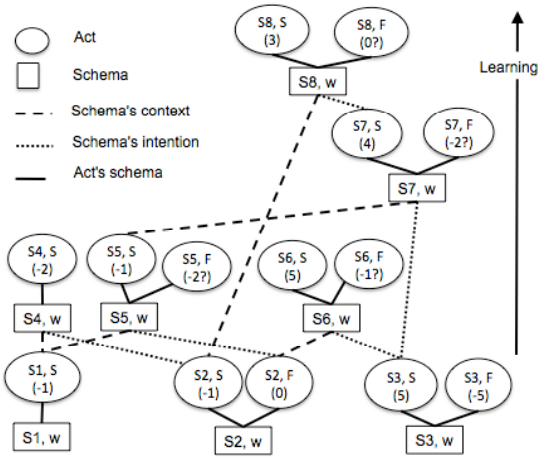


Figure 1: Example schema and act hierarchy.

At its lowest level, Figure 1 shows primitive schemas S1, S2, and S3. Primitive schemas define the agent's primitive possibilities of behavior within a given environment. For example, as further detailed in the experiment section, S1 may correspond to *turn right*, S2 *touch ahead*, and S3 *attempt to move forward*. Primitive acts are represented above primitive schemas. For example, act [S3, S, 5] corresponds to *succeeding in moving forward*, while [S3, F, -5] corresponds to *bumping into a wall*. Each act has a value associated with it, in this case: 5 and -5 (in parentheses in the figure). These values inform the selection of the next schema to enact, as explained later. For now, we can understand these values as the agent's *satisfaction* for performing the act.

Primitive satisfaction values are chosen and hard-coded by the modeler according to the behavior she intends to generate. In our example, act [S3, S, 5] means that the agent *enjoys* moving forward, while act [S3, F, -5] means that the agent *dislikes* bumping into walls. Similarly, act [S2, S, -1] means that the agent *touches a wall* in front of him, which he slightly dislikes; while [S2, F, 0] means that the agent *touches an empty square*, which leaves him indifferent. Therefore, primitive satisfaction values are also a way for the modeler to define the agent's intrinsic motivations.

Higher-level schemas are learned through experience, by combining lower level schemas. Schema learning consists of adding the new-learned schema to the agent's memory as a node and two arcs pointing to the schema's sub-acts. For example, schema S5 is learned when the agent has turned to the right and then touched an empty square. Schemas have a *context act* (dashed line in the figures throughout this paper), an *intention act* (doted line), and a weight (w). So, S5 means that, when the agent has successfully turned right, the agent can expect to touch an empty square. Similarly, S4 is learned when the agent has successfully turned right and touched a wall. S4 thus generates the opposite expectation from S5. A schema's weight corresponds to the number of times the schema has been enacted. Over the course of the agent's interactions, the relative schema weights thus balance the agent's expectations in specific contexts.

When a higher-level schema is learned, its succeeding act is also learned with a satisfaction value set equal to the sum of the satisfaction values of its sub-acts, e.g., [S4, S, -2] (-1-1) and [S5, S, -1] (-1+0). When a higher-level schema gains enough weight, it can be selected for enaction. Enacting a higher-level schema consists of sequentially enacting its sub-acts. For example, enacting S5 consists of enacting S1 with a succeeding status, then enacting S2 with a failing status. Hence, the satisfaction for enacting a high-level act is equal to the satisfaction for individually enacting its sub-acts.

When a high-level schema fails during enaction, it is interrupted. This happens if a status returned by the environment does not match the expected status of a sub-act. In this case, the failing act of the schema is learned or reinforced, as well as the *actually enacted act*. The satisfaction value of the failing act is set equal to the satisfaction value of the actually enacted act. For example, if schema S6 fails because S2 succeeds, then [S6, F, -1] is learned. Because high-level schemas can potentially fail at any step of their sequence, their failing act's satisfaction values are averaged over their different failures.

When a high-level schema is enacted, it generates the learning of higher schemas. For example, when S5 is successfully enacted and followed by succeeding S3, then S7 is learned. In this example, S7 consists of turning right, touching an empty square, and then successfully moving forward. [S7, S]'s satisfaction is set equal to 4 (-1 + 5). Similarly, S8 (learned after S7) consists of touching a wall, turning right, touching an empty square, and moving forward.

Algorithm

The algorithm follows two overlapping cyclical loops. The *control loop* consists of: 1: selecting a schema for enaction, 2: trying to enact the selected schema, 3: learning what can be learned from this trial, 4: computing the resulting situation, and finally looping to step 1. We call this loop the control loop because it is at this level that the agent *decides* what schema to try to enact.

Step 2: (trying to enact a schema) constitutes a nested loop that goes through the selected schema's hierarchical structure and tries to enact each of its primitive acts sequentially. We call this loop the *automatic loop* because this loop enacts sub-schemas below the agent's decision process. Figure 2 illustrates this procedure.

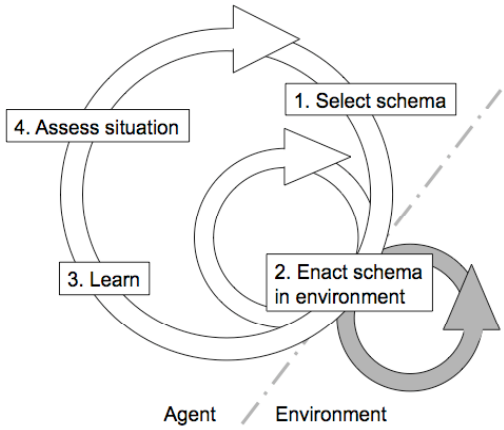


Figure 2: Algorithm procedure.

In Figure 2, the large white circle represents the control loop while the small white circle represents the automatic loop. The gray circle represents the environment's loop. Each revolution of the automatic loop corresponds to a revolution of the environment's loop that returns the status of the enacted primitive schema. From the viewpoint of the control loop, the schema's enaction constitutes only one step, whatever the schema level is in the hierarchy. Therefore, at the control loop level, any schema is handled similarly as a primitive schema, which makes possible the recursive learning of higher-level schemas.

The four steps of the control loop are:

Step 1: All schemas whose *context act* matches the previously assessed situation propose their *intention act*. The weight of this proposition is computed as the proposing schema's weight multiplied by the intention act's satisfaction. The schema with the highest proposition is selected (if several schemas are equal, one is randomly picked among them). In essence, this mechanism selects the schema that will result in the expected act having the highest satisfaction, balanced by the probability to obtain this expected act. This probability is based on what the agent has learned thus far concerning the current context. Due to this mechanism, the agent appears (to an observer) as though he was seeking to enact the act associated with the highest *believed* expected satisfaction and avoiding the acts with the lowest ones. Figure 3 illustrates this mechanism.

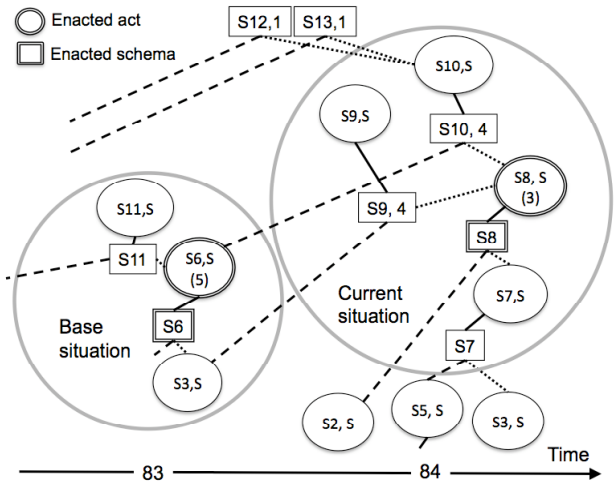


Figure 3: Enaction mechanism.

Figure 3 details the 84th iteration of the control loop in the experiment reported in Figure 5. On the 83rd iteration, schema S6 was successfully enacted (touch empty square, move forward), which resulted in a base situation of [S6, S], [S3, S], and [S11, S] (and other acts on top of [S6, S] not reported in the figure). In this context, S9 and S10 were activated and proposed to enact S8 with a proposition weight of 4x3+4x3 (sum of the proposing schema's weight multiplied by [S8, S]'s satisfaction) (the agent never experienced S8 failing). This proposition happened to be the highest of all the propositions, which resulted in S8 being selected for enaction.

Step 2: The algorithm next enacts all the selected schema's sub-acts. If all the sub-acts meet their expectations, the control loop proceeds to step 3. If the enaction of one of the sub-acts is incorrect, then the automatic loop is interrupted; the schema's enaction status is set to fail; and control is returned to the control loop. In Figure 3's example, the enaction of schema S8 consists of the enaction of acts [S2, S], [S5, S] (made of [S1, S] and [S2, F], as shown in Figure 1), and [S3, S] in a sequence. In this case, S8 was successfully enacted, resulting in the enacted act [S8, S].

Step 3: New schemas are learned or reinforced by combining the base situation and the current situation. In Figure 3's example, S9's weight is incremented from 6 to 7, and S10's weight is incremented form 4 to 5. In addition, new schemas are learned based on the penultimate situation and on [S10, S] (e.g., S12 and S13 are created with a weight of 1, as well as other schemas not represented in the figure).

Step 4: The *base situation* becomes the *penultimate situation* and the *current situation* becomes the base situation for the next cycle. A *situation* is made of the acts that surround the enacted act (i.e., the enacted act, the acts directly below it, and the acts directly above it). In Figure 3's example, the *situation* is made of [S8, S], [S7, S], [S9, S], and [S10, S]. The *situation* can be understood as the agent's *situation awareness*, that is, a representation of the agent's situation in terms of affordances (Gibson, 1979) capable of activating behavior. Limiting the situation to the acts directly surrounding the enacted act prevents the agent from being overwhelmed by a combinatorial explosion as the agent creates new schemas. In essence, the agent focuses on the current level of *abstraction* for representing his situation, for making his choices, and for finding and learning higher-level regularities. When a high-level schema fails during enaction, the agent constructs the *actually enacted* schema and falls back to a lower abstraction level.

Experiment

To test the algorithm, we developed an environment that afforded the agent hierarchical sequential regularities to learn and organize. Although the interaction's structure—resulting from the coupling of the environment with the agent's primitive schemas—is fundamentally sequential, the environment appears to external observers as a two-dimensional grid represented in Figure 4, implemented from Cohen's (2005) Vacuum environment.

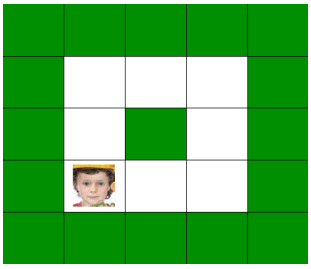


Figure 4: Experimental environment.

In Figure 4, white squares represent empty squares where the agent can go, and filled squares represent walls. The agent's primitive schemas and acts are defined as described above (S1=turn 90° right (-1/NA), S2=touch the square ahead (-1/0), S3=attempt to move one square forward (5/-5)). Additionally, we have primitive schema S0 consisting of turning to the left (-1/NA) (turning schemas S0 and S1 always succeed in this environment). These settings offer a first notable regularity, namely that the agent can increase his average satisfaction by touching ahead before trying to move forward, and not moving forward if he touches a wall.

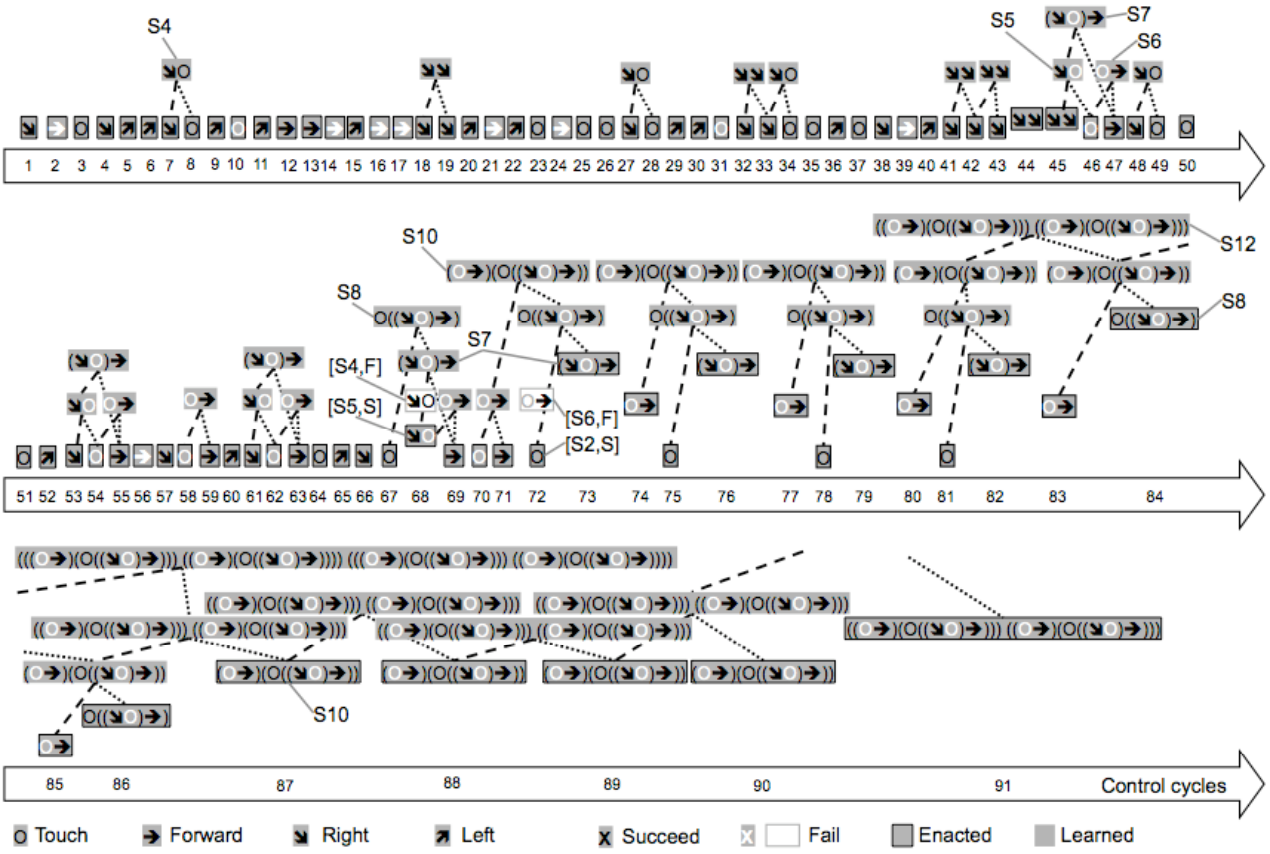


Figure 5: An example run among the 18 reported in row 6 of Table 1.

Next, the agent can increase his satisfaction by repeating the sequence consisting of moving forward twice and turning once. Higher-level regularities consist of repeating this later sequence. The effects of this learning mechanism are shown in detail in Figure 5 that reports an example run. Videos of other runs can be seen online[1].

In Figure 5, an attempt to move forward is represented as an arrow to the right, a turn-left as an upward arrow, a turn-right as a downward arrow, a touch as a O. Succeeding primitive schemas use a black font, while failing primitive schemas use a white font, i.e., white rightward arrows mean that the agent bumped into a wall, and white Os mean that the agent touched an empty square in front of him. Enacted schemas are represented at the lowest level in each line with a black outline. Learned schemas are represented on top of the enacted schemas. Failing higher-level schemas are represented as white boxes with gray outlines (steps 68 and 72). The numbers from 1 to 91 indicate the control-loop iterations (steps).

At the beginning, the agent acts randomly because he has not yet learned appropriate schemas that could propose their associated intention sub-schema. However, every cycle, the agent constructs or reinforces several schemas. For clarity, Figure 5 only reports the construction and the reinforcement of the schemas that matter for the purpose of explanation, and references these schemas when they are mentioned in the text. Schema S4 is constructed on step 8. S4 is then reinforced on step 28, 34, and 49. The agent attempts to enact S4 for the first time on step 68 but fails and enacts S5 instead.

Notably, a schema *turn right-turn right* (not named in this paper) is constructed on step 19. This schema is reinforced on steps 33, 42, and 43. It is then enacted twice on steps 44 and 45. It is, however, not used any further because other options prove more satisfying (its satisfaction value is -2).

On step 46, the agent constructs the schema S5 (using act [S1, S] that is the schema *turn right-turn right*'s intention act). Then, on step 47, the agent finds the schema S6 (touch empty, move forward), and also constructs the schema S7 on top of S5. After step 47, the schema S6 always prompts the agent to try to move forward after touching an empty square; therefore, from then on, S6 is quickly reinforced in steps 55, 59, 63, and 71. The agent tries to enact S6 for the first time on step 72, but unsuccessfully, which results in falling back to [S2, S]. This experience instructed the agent that schema S6's failing act has a satisfaction of -1, which is still better than trying to move forward without touching first and bumping into a wall (satisfaction -5). Therefore, from then on, the agent learned to touch before moving forward. S6 is then successfully enacted on steps 74, 77, 80, 83, and 85.

As said previously, on step 68, the agent intended to enact S4 but actually enacted S5. Because S7 is directly above enacted schema S5, S7 is included in the agent's situation

[1] http://e-ernest.blogspot.com/2009/07/ernest-64.html

awareness, which results in the learning of the fourth-order schema S8 on step 69. Then, on step 73, the enaction of schema S7 generated the learning of schema S10. As detailed in Figure 3, S8 is enacted for the first time on step 84, which generated the learning of S12. S10 starts to be enacted on step 87.

After step 87, the agent keeps on performing the sequence *touch empty, move forward, touch wall, turn right, touch empty, move forward.* This regularity introduces repeated circuits that lead to higher-level repetitions of this sequence. With this sequence, the agent obtains a satisfaction of 8 within 6 primary steps, i.e., 1.33 per primary step.

In this example, the agent did not learn the optimum sequence in the environment. In fact, the agent has no way to know whether the stabilized sequence is optimum or not. The agent only repeats a sequence when other actions appear less likely to bring satisfaction, based on what he has learned before. In most instances, the agent first learns to touch before moving, after which he begins to build other regularities based on this initial pattern.

The experiment was run 100 times, stopping each run when the agent has reached a stable sequence, and clearing the agent's memory between each run. The results are summarized in Table 1.

Table 1: Summary of hundred runs.

Row	Runs	Satisfaction/step	Cycles
1	22	3.00	50
2	22	2.25	79
3	4	1.80	75
4	4	2.00	69
5	16	1.60	62
6	18	1.33	84
7	1	1.40	76
8	1	1.17	109
9	1	1.00	108
10	2	0.75	116
11	3	1.00	61
12	1	0.80	95
13	3	1.00	71
14	2	0.40	96
	100	1.92	72

In Table 1, the runs are aggregated by average satisfaction per step obtained when the agent has reached a stable sequence. The column *Cycles* reports the average number of control loop cycles before reaching this sequence. Rows 1 through 6 report 86 runs where the agent learned to go around his environment and got a satisfaction per step greater than or equal to 1.33. Rows 7 to 14 report 14 runs where the agent has stabilized on a sequence that results in staying on one edge of the environment, and reached a satisfaction per step that ranged between 0.40 and 1.40.

The summary row shows that the average reached satisfaction per step was of 1.92. It was reached in an average of 72 cycles. In comparison, other experiments yielded an average satisfaction values per step of -0.93 without any learning and -0.38 with only the first-level schema learning. This data demonstrates that, in all the runs, the hierarchical learning mechanism has substantially increased the agent's satisfaction, compared to no or non-hierarchical learning.

Related work

To our knowledge, this work constitutes the first implementation of an intrinsically motivated agent who recursively learns to exploit hierarchical sequential regularities to fulfill drives. The closest related work is probably Drescher's (1991) attempt to implement Piagetian constructivist learning through what he called the *constructivist schema mechanism*. Like our implementation, Drescher's work constructed hierarchical schemas that associated context, actions, and expectations. In Drescher's implementation, however, schemas were neither associated with satisfaction values nor did the agents exhibit self-driven behavior. The agent's exploration was rather random and resulted in a combinatorial explosion as the agent encountered increasingly complex environments.

Chaput (2004) proposed the Constructivist Learning Architecture (CLA) to address Drescher's scalability issues. The CLA implemented a scheme *harvesting* mechanism at each hierarchical level. This harvesting, however, depended on goals defined by the modeler. Chaput's solution, therefore, relies upon a problem-solving approach that in fact differs from our self-driven mechanism of interest.

In developmental robotics (Weng et al., 2001), the literature often refers to Brooks's (1991) pioneering work. For example, Blank, Kumar, Meeden, and Marshall (2005) describe the principles for a self-motivated/self-organizing robot. They use the robot's anticipation reliability as a motivational regulator for the robot. As opposed to our work, these implementations do not make explicit the robot's driving satisfaction values. They also rely on a limited number of predefined hard-coded hierarchical layers, which restricts the agent's learning possibilities.

As for the testbed environment and self-driven learning paradigm, our approach appears to be rather unique. We must note that our learning paradigm substantially differs from maze solving experiments (e.g., Sun & Sessions, 2000) or from hierarchical sequence learning as depicted in the classical taxi cab experiment (Dietterich, 2000). In these experiments, the learning occurs over multiple runs (often thousands), and comes from a reward value that is given when the goal is reached and then backward propagated during subsequent runs. On the contrary, in our paradigm, there is no final goal that would provide a reward; the learning occurs through each run; and all the agent's memory is reinitialized between each run (including all forms of reinforcement).

Discussion and conclusion

Besides the quantitative results of the agent's measured satisfaction and that it learns at a nice pace (neither one shot nor thousands shots learning), this work offers qualitative results in the form of the agent's exhibited behavior. When

observing the agent, an observer can infer that the agent seems to enjoy certain behaviors (such as moving forward) and dislike others (such as bumping into walls). Moreover, the agent appears to learn to endure unpleasant behaviors (such as turning or touching) to have more opportunities to move forward. The agent thus appears to be self-motivated and appears to learn knowledge about his environment that he uses to satisfy his predilections. More elaborated behaviors can be watched in videos online[2].

In addition, the agent appears to learn to use certain schemas as perceptions (e.g., schema S2 to *sense* the square forward), and to determine subsequent actions based upon these schema's outcomes. Therefore, the agent seems to simultaneously learn to perceive his environment and to make sense of his perception. This result is original in that the agent's perception was not pre-defined by the modeler in the form of a perceptual buffer, as it is in many cognitive models. In our case, perception emerges from the agent's behavior, which grounds the meanings of the agent's perceptions in his activity.

Moreover, the agent constructs an internal data structure made of elaborated behavioral patterns, and uses this data structure to deal with his environment. The behavioral patterns used in this data structure are only those confirmed through experience, which helps the agent deal with the environment's complexity. These data structures can be seen as the agent's *situation awareness* that is constructed through his interactions, and that activates subsequent behavioral patterns based on expected enjoyment. At each point in time, the current agent's knowledge frames how the agent *sees* the world, which makes possible the recursive learning of higher-level regularities and which accounts for the agent's individualization through his development.

Preliminary experiments in more complex environments show that this algorithm faces two notable limitations. One limitation is that the algorithm may represent the same primitive sequence by different schemas that have different hierarchical structures. These different schemas are useful to find appropriate hierarchical regularities but they impede the agent's performance in more complex environments. Future studies should find a way to merge these schemas. The second limitation is that the algorithm is not good at finding spatial regularities. For example, if we replace the central wall square with an empty square, the agent becomes less likely to find the most satisfying regularity, that of making a continuous circuit around his environment.

We, nevertheless, believe that these limitations are not insurmountable, and we plan to gradually increase the complexity of the agent and of the environment in future studies. We will add new drives to the agent, for example homeostatic drives (similar to hunger) or boredom-avoidance based on top-level regularity detection. We will also add other primitive schemas, especially schemas associated with distal perception. These schemas should, we believe, help the agent deal with open spatial environments.

[2] http://e-ernest.blogspot.com/2009/10/enrest-72.html

Acknowledgments

Support was provided by ONR (N00014-06-1-0164 and N00014-08-1-0481) and DTRA (HDTRA 1-09-1-0054).

References

Blank, D. S., Kumar, D., Meeden, L., & Marshall, J. (2005). Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture. *Cybernetics and Systems, 32*(2).

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence Journal, 47*, 139–159.

Chaput, H. H. (2004). *The Constructivist Learning Architecture: A model of cognitive development for robust autonomous robots.* Unpublished doctoral dissertation, The University of Texas, Austin.

Dietterich, T. G. (2000). *An Overview of MAXQ Hierarchical Reinforcement Learning*. Paper presented at the SARA02 4th International Symposium on Abstraction, Reformulation, and Approximation.

Drescher, G. L. (1991). *Made-up minds, a constructivist approach to artificial intelligence*. Cambridge, MA: MIT Press.

Gibson, J. J. (1979). *The ecological approach to visual perception*. Boston: Houghton-Mifflin.

Harnad, S. (1990). The symbol grounding problem. *Physica, D*(42), 335-346.

Le Moigne, J.-L. (1995). *Les épistémologies constructivistes*. Paris: Presse Universitaire de France.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. (1975). Computer science as empirical inquiry: symbols and search. *Communications of the ACM, 19*(3), 113-126.

Piaget, J. (1937). *The construction of reality in the child*. New York: Basic Books.

Suchman, L. A. (1987). *Plans and situated actions*. Cambridge: Cambridge University Press.

Sun, R. (2004). Desiderata for cognitive architectures. *Philosophical Psychology, 17*(3), 341-373.

Sun, R., & Sessions, C. (2000). Automatic segmentation of sequences through hierarchical reinforcement learning. In R. Sun & C. L. Giles (Eds.), *Sequence Learning* (pp. 241–263). Berlin Heidelberg: Springer-Verlag.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., et al. (2001). Artificial intelligence - Autonomous mental development by robots and animals. *Science, 291*(5504), 599-600.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review, 9*(4), 625-636.

Modeling the Effects of Work Shift on Learning in a Mental Orientation and Rotation Task

Tim Halverson (thalverson@gmail.com)
Oak Ridge Institute for Science and Education
Air Force Research Laboratory
Mesa, AZ 85212 USA

Glenn Gunzelmann (glenn.gunzelmann@mesa.afmc.af.mil)
Air Force Research Laboratory
Mesa, AZ 85212 USA

L. Richard Moore Jr. (larry.moore@mesa.afmc.af.mil)
Lockheed Martin
Air Force Research Laboratory
Mesa, AZ 85212 USA

Hans P.A. Van Dongen (hvd@wsu.edu)
Sleep and Performance Research Center, Washington State University
Spokane, WA 99210 USA

Abstract

Circadian rhythms cause alertness declines at night, producing performance decrements across cognitive domains and tasks. Building on the learning mechanisms for declarative knowledge instantiated in the ACT-R cognitive architecture, this research seeks to explain the effects of circadian rhythms on performance of an orientation task performed repeatedly across two weeks by participants working either day or night shifts. The differences in performance between the two groups are best explained by varying the decay rate in declarative knowledge as a function of the time of day the task was performed. The model accounts well for task learning reflected in decreases in response times across days, as well as differences in learning between the day and night shift conditions.

Keywords: sleep; circadian rhythm; fatigue; learning; shift work; declarative memory; spatial; ACT-R

Introduction

Variations in alertness due to circadian rhythms and sleep loss have been shown to affect various components of cognitive functioning (e.g. Jackson & Van Dongen, in press). For example, vigilant attention (Lim & Dinges, 2008), perceptual learning (Mednick, Nakayama & Stickgold, 2003), and motor learning (Walker, Brakefield, Morgan, Hobson & Stickgold, 2003) are all affected by fluctuations in alertness associated with time awake and circadian rhythms.

Despite well-documented behavioral changes, it is not well understood how nighttime operations affect learning in different contexts. Most research on night and shift work has focused on how shift differences affect sleep and frequency of accidents (e.g. Åkerstedt, 1988). The affect of changes in alertness (e.g., as associated with work shift differences) on learning is one area of research where a better understanding of the mechanisms involved is needed. More detailed explanations hold the promise of enabling predictions about how learning experiences at different times of the day may differ, and how this may impact eventual task performance.

Previous cognitive modeling efforts have explored some effects of moderators on cognitive processes. In fact, several studies have examined such effects in the context of declarative knowledge. For instance, the effects of caffeine on memory retrieval have been modeled by increasing the activation of declarative knowledge (Kase, Ritter & Schoelles, 2009). Conversely, the effects of sleep loss on memory retrieval have been explained using decreases in declarative activation (Gunzelmann, Gluck, Kershner, Van Dongen & Dinges, 2007). The negative effect of time-on-task on response accuracy has been explained by increasing noise, making misretrievals more common (Fu, Gonzalez, Healy, Kole & Bourne Jr, 2006).

These research efforts focused on processes associated with retrieving declarative knowledge by impacting the availability or confusability of chunks when they are requested. In contrast, the effects of alertness on the learning and retention of declarative knowledge have not been addressed.

In the research presented here, we investigate how long-term learning may be affected by fluctuations in alertness resulting from circadian rhythms during laboratory-simulated shift work. This is accomplished within the context of a spatial direction task based on Gunzelmann, Anderson, and Douglass (2004), which was performed repeatedly by participants over two weeks. A computational cognitive model is presented that accounts for changes in observed response times across successive days of the study, including differences in learning rates as a function of

simulated work shift. Differences in performance between shift conditions are explained by manipulating the decay rate parameter in ACT-R's declarative knowledge activation function. Increased decay (reduced learning) in the night shift condition leads to performance decrements that match the human data. The details of the task, the human performance data, and the model are described in the following sections.

Orientation Task

This experiment was conducted as part of a larger study to understand how circadian rhythms and sleep disruption affect performance in a variety of domains. The participants were screened to be healthy and without sleep disorders, with no evidence of brain damage or learning disabilities, and free of drugs of abuse. Participants gave written informed consent, and were paid for their participation.

Figure 1 shows the orientation task used in this study. There are 8 possible target locations (left) and 8 possible misalignments (right; 45 degree intervals). Twenty-five randomly ordered trials were presented per session — 5 target locations (bottom, near, middle, far, and top) crossed with 5 misalignments (0, 45, 90, 135, and 180 degrees). Because performance is roughly equivalent for right-left mirrored stimuli for both target location and misalignment (see Gunzelmann, Anderson & Douglass, 2004), the location was selected at random from the left or right positions.

Participants received instructions that encouraged them to mentally rotate the relative positions of the viewpoint (indicated by the "You" arrow) and the target on the overhead view (left side filled circle) to align them with the viewpoint indicated on the map view (right side arrow). Specifically, they were taught to imagine an angle that connects the viewpoint to the target on the overhead view, with the vertex at the center of the field (a 90 degree angle in Figure 1). They were then told to mentally shift to the map view, and to rotate the angle so that the arrow in the overhead view was aligned with the arrow in the map view (a rotation of 90 degrees clockwise in the trial shown in Figure 1). At this point, the answer could be determined by finding the target end of the angle.

Participants responded using the numeric keypad portion of a computer keyboard, which was spatially mapped to the possible response locations on the map view. So, if the target was in the bottom position on the map (as it is in the sample trial shown in Figure 1), participants responded by pressing the "2" on the numeric keypad.

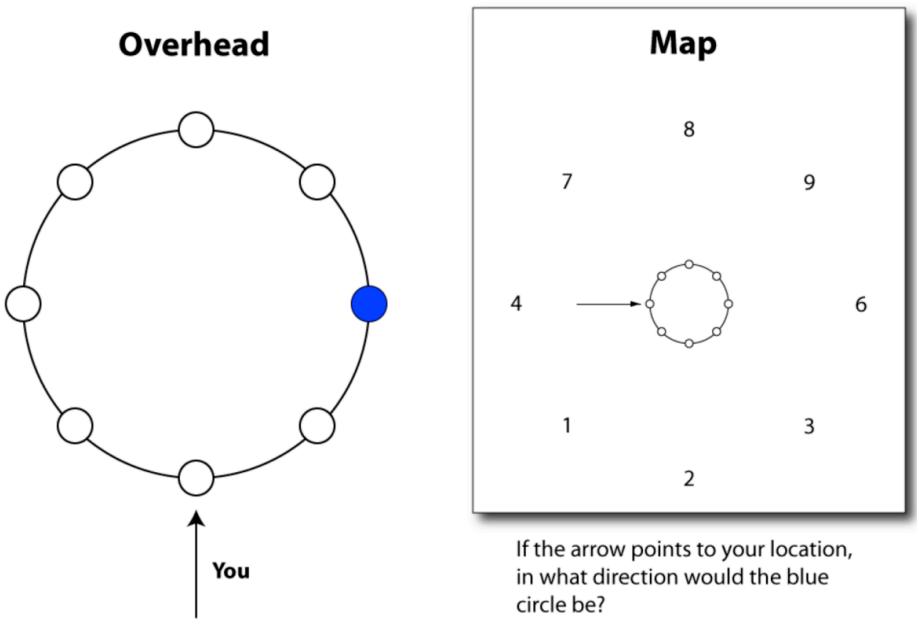


Figure 1: An example trial. The target on the overhead ego-oriented view (left side), indicated by the filled circle, is at middle distance to the right of center. The perspective on the map view (right side), indicated by the arrow, is misaligned by 90° clockwise. The correct response in this example trial is "2."

Method Thirteen participants, ranging in age from 22 to 39 years old (mean = 28), were in the laboratory for fourteen consecutive days. The first day was a baseline day with 10 hours in bed for sleep (22:00–08:00). Subsequently, some of the participants (n = 6) changed to a simulated night shift. Night shift participants were given five hours in bed (15:00–20:00) on the second baseline day, before starting five consecutive work days with 10 hours in bed during the daytime (10:00–20:00) on each day. On the seventh and eighth day, night shift participants had a simulated "day off" during which they had 5 hours in bed (10:00–15:00), 7 hours awake, 10 hours in bed during the night (22:00–08:00), 7 hours awake, and then 5 hours in bed (15:00–20:00), before resuming their night shift schedule for the next 5 days. This schedule represented a common schedule for individuals working a night shift, who frequently switch back to a nighttime sleep schedule during weekends. After the last night shift day, night shift participants received 5 hours in bed (10:00–15:00), 7 hours awake, and then, on the final day of the study, were given 10 hours in bed at night (22:00–08:00) for recovery.

Participants on the day shift (n = 7) were subjected to the same pattern of two baseline days, five consecutive work days, a "day off", another five consecutive work days, and a recovery day. They maintained the same sleep schedule throughout the study, however, with 10 hours in bed (22:00–08:00) each night. Note that participants on the day shift and night shift schedules were given the same amount of time in bed over the course of the experiment; it was merely distributed differently.

Over the course of the study, participants completed fifty-one test sessions of the spatial direction task, with 2 to 4 sessions per day. Before the first session, participants were presented with instructions for the task.

Eight to sixteen days prior to the first session (mean = 14 days), participants were given baseline training on the spatial direction task. This included a set of instructions for the task and four training sessions (these data are not modeled here).

Observed Data

Average response times for each day of the study are presented in Figure 2 for both the day and night shift conditions. Performance during the baseline days of the study (days 1 and 2) was similar for the two groups, and there was no significant difference in mean RT at that point. However, when the conditions diverged, so did performance on the spatial direction task. The performance of the night shift group did not recover during the simulated "day off", and differences in mean response time remained at the end of the experiment.

To evaluate the differences between shift conditions, we compared response times on the days when they were awake for different shifts (ten days; excluding the baseline, day off, and recovery day) using a linear mixed-effect model with subject as a repeated-measure grouping factor. This was planned *a priori* to most effectively evaluate the impact of shift on performance. However, for the model comparisons later in the paper, all of the observed data was used. See Halverson, Gunzelmann, Moore, and Van Dongen (in press) for more complete analyses of the human data.

Figure 2 shows the mean participant response times (solid lines) as a function of day in study and simulated work shift. There was a steady decrease of response time between days 1 and 14, as corroborated by a main effect of day, $F(9, 7769) = 112.2$, $p < .001$. While there was no evidence of an overall effect of shift, $F(1, 11) = 0.8$, $p = .37$, there was an interaction between shift and day, $F(9, 7769) = 2.1$, $p = .03$. Response times did not improve as quickly when a participant was on the night shift. Observed error rates were low (M = 4%, SD = 3%) and are not addressed in this work.

Mental Rotation Model

A computational cognitive model of the orientation task was developed using the ACT-R 6.0 cognitive architecture (Anderson et al., 2004). The model behavior is primarily driven by mental rotations and learning. The mental rotation operation is implemented using ACT-R's imaginal module and the imaginal-action buffer. Learning in the model occurs both in the declarative module and through the compilation mechanisms in procedural knowledge. The task procedure implemented in the model was based on the instructions given to the participants in the empirical study.

Model Overview

The model executes the task as follows: In the overhead view, the model encodes the angle defined by the target (blue circle), the center of the overhead view, and the viewpoint (circle nearest the "You" arrow) by visually attending those locations and encoding their coordinates in the imaginal buffer. The model then switches to the map view, encoding the vector defined by the viewpoint (circle

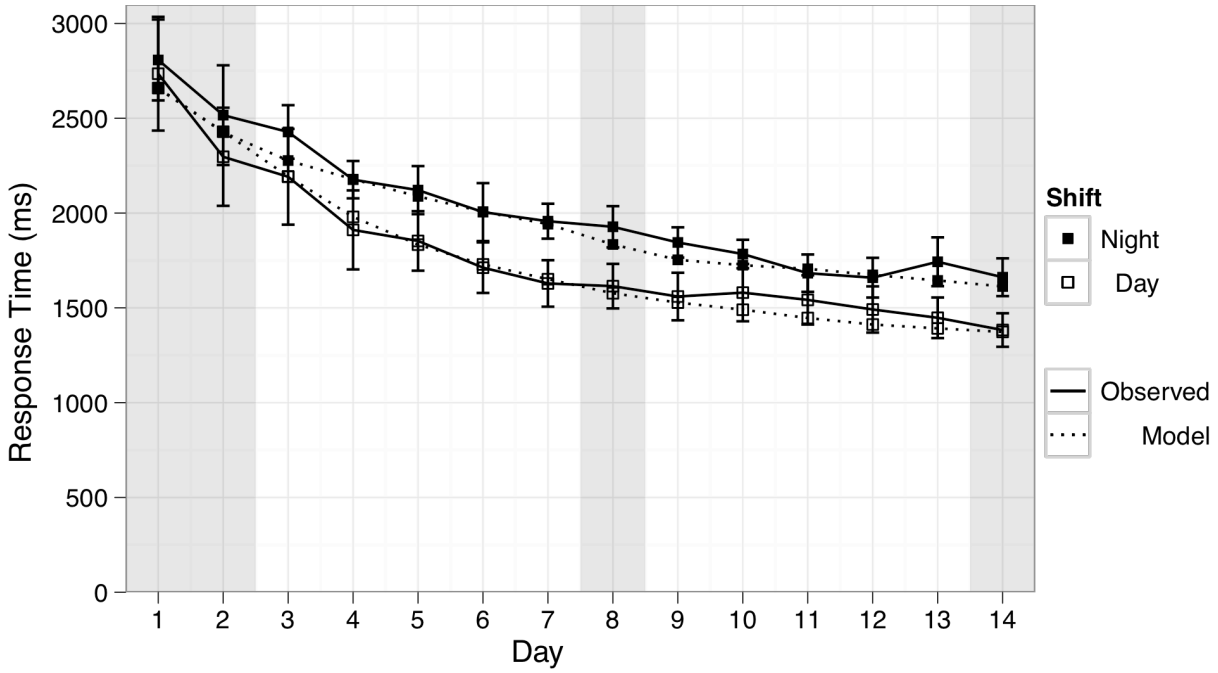


Figure 2: Observed and predicted mean response times as a function of day and simulated work shift (night or day). The shaded regions indicate simulated "days off" in which night shift participants (and the model) performed the task during the day at the same time as day shift participants. Shaded days are not included in the human data analysis. Error bars indicate ±1 standard error.

nearest the arrow) and center of the map view by attending those locations and encoding their coordinates.

The angle that was encoded in the overhead view is then translated to center it on the map view (an imaginal action; 200 ms) and rotated to align the viewpoints of the overhead and map views. The model visually attends the response location closest to the transformed location of the target, encodes the response digit, and presses the corresponding keyboard key.

Mental rotations were implemented using the ACT-R imaginal module. The time to perform the rotation was based on previous mental rotation research (e.g. Bethell-Fox & Shepard, 1988) and was a linear function of the angle of rotation. The slope of the linear function was a free parameter, as the slope can vary by task depending on the relative complexity of the object to be rotated.

Learning

The model's performance improves over time by learning in three ways. First, the angle from the overhead view is encoded in declarative memory when the first subtask is completed. In subsequent trials, the model attempts to retrieve an existing chunk based on the target's location. If a chunk exists and gets retrieved before the model completes the process of visually encoding the angle, then the information from the chunk that was retrieved from declarative knowledge is used. Over time, retrievals become more likely and faster than explicitly encoding the angle using perceptual and imaginal actions. This leads to a speed-up in the model's execution of the task.

In addition to an increasing reliance on declarative representations for target location information, the second step of the solution process is also stored in declarative knowledge once the response is made. These chunks contain information about the target location from the overhead view as well as the perspective on the map view (i.e., the misalignment). Consequently, with experience the model can attempt to retrieve the response based on the target location and map view perspective location. Like encoding the target location on the overhead view, if a chunk is retrieved before the model completes the mental transformations on the map view, the response is based upon the chunk retrieved from declarative knowledge.

The final learning process in the model involves ACT-R's production compilation (i.e. proceduralization). Production compilation is a process by which new productions are created dynamically to represent in one step the consequences of two productions that execute consecutively. With experience, it becomes increasingly likely that the new production will be used, as the model learns that the utility of the new production is greater than the utility of the original, separate productions. However, due to the many constraints imposed on production compilation by the architecture and the structure of this model, the only compilation that occurs in the current model involves encoding the mental rotation into productions specific to each pair of overhead target and map view perspective locations. Therefore, the only savings introduced by production compilation were the infrequent, but substantial, time savings from the mental rotation of trial layouts that were only seen once per session.

Explaining Night Shift Performance Decrements

Several alternatives were explored to explain the decrement in performance observed for participants on the night shift. The solution that resulted in the best explanation of the data was a variation of the decay rate of declarative chunks activation as a function of simulated work shift. Alternative solutions that did not explain the observed trends as well are described in the Results and Discussion section.

By default, the decay rate parameter is not allowed to vary in the implementation of ACT-R. That is, the decay rate can be set, but it assumes the same value for the duration of a model run. There have been various efforts to implement more dynamic mechanisms for decay in ACT-R. Most of these have been related to accounting for the spacing effect (Anderson, Fincham & Douglass, 1999; Jastrzembski & Gluck, 2009; Pavlik & Anderson, 2005).

In our case, we utilize the decay rate to represent differences in the effectiveness of learning as a function of when during the day the task was performed. To implement the mechanisms, the equation to calculate the base-level activation of declarative chunks was modified (Equation 1). The only change to the standard ACT-R base-level learning equation is that the value of the decay rate parameter can vary according to the time when a chunk was added to declarative memory or when the chunk was rehearsed (d_j), as opposed to a constant decay rate across all rehearsals (d) in the original equation. This modification does not change the effect of decay for current ACT-R models.

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d_j}\right) + \beta_i \qquad (1)$$

The current model was implemented with the simplifying assumption that the level of alertness, and thus the value of d_j, is constant across all hours of a work shift (day or night). It is well known that alertness due to circadian rhythms varies throughout the day and night (Van Dongen & Dinges, 2005). However, while the model executed the task the same number of times as the participants did through a simulated workday, we aggregated the data across each day to reduce noise. We have not yet evaluated the capacity of the mechanism to account for finer grained circadian rhythm fluctuations or varying inter-session intervals.

The model was fit to the day shift data using the retrieval threshold (best fit = 1.2), retrieval latency factor (8.0), and rotation slope (0.009 sec/degree) parameters. The rotation slope is similar to the slope found in previous research for simple rotations (Bethell-Fox & Shepard, 1988). The base level learning, which controls the rate of activation decay (d_j), was left at the ACT-R default (0.5) during sessions when participants were on the day shift. For predicting the night shift data, the declarative chunk decay rate was

allowed to vary. The best fitting decay parameter for the night shift sessions was 0.6.

Results and Discussion

Figure 2 shows observed (solid lines) and best fitting model (dashed lines) mean reaction times as a function of day in the study and simulated work shift (night or day). For both shifts, the observed behavior is well predicted (RMSD = 65 ms, r^2 = .98 for day shift; RMSD = 79 ms, r^2 = .98 for night shift). The night shift predictions are particularly noteworthy, as only one parameter was varied relative to the day shift model.

The model is able to predict the observed response times well across fourteen days, including differences across work shifts (i.e. the interaction of day and shift). The model is able to predict the effects of work shift changes well with variations in declarative memory decay rates based on the time at which the tasks are performed. While the declarative decay mechanism explains the observed decrements well, several alternative mechanisms for explaining the trends were considered.

One alternative mechanism involves manipulating overall declarative chunk activation at the time of retrieval, as was done in Gunzelmann et al. (2007). This model did fit the observed data on most days, but did not correctly predict the effect on the overall learning rate when the participants in the night shift condition temporarily switched to the day shift on days 8 and 14. On these days, the model predicts that the performance of participants in the night shift group is nearly equivalent to that of participants in the day shift group. This is because the model assumes that the participants' alertness recovers when performing the task during the day. There is some evidence in associated data (not reported here) to support this, although we do not have conclusive evidence. Regardless, if the impact of degraded alertness were only on activation levels, then the knowledge should be more available during the day. As the human data illustrate, however, the deficits associated with performing the task on the night shift persisted.

Another alternative mechanism for explaining the decrements of alertness is a decrement to utility values associated with production selection and execution. This mechanism has been used to predict performance decrements due to decreased alertness in vigilance tasks (e.g. Gunzelmann, Moore Jr, Salvucci & Gluck, 2009). However, such a mechanism in the model presented here does not explain the observed data for the current task. The same issue is encountered as with the previous alternative — the model recovers to day shift levels of performance on the "day off" and "recovery" days. This is likely a result of the current task requiring constant engagement, over short periods, and thus mechanisms employed for sustaining attention throughout the task would not be stressed.

A third alternative mechanism that was explored is a variation in *procedural* learning as a function of shift. The model presented in this paper has both procedural and declarative learning enabled. It may be that the observed night shift decrement resulted from a slowing of procedural learning rather than a slowing of declarative learning. To test this, the rate of learning for productions rule utilities was varied. This made little difference in the predicted results. This lack of predictive power may result from either the way in which the model was constructed, with an emphasis on declarative knowledge, or a result of the study design, with most of the procedural learning occurring early in the protocol when all participants performed the task during the day.

Thus, the model presented here provides support for the hypothesis that variations in alertness have an impact on learning that may persist beyond immediate task performance. This is consistent with previous research that has indicated that sleep loss causes deficits in encoding declarative knowledge (see Jackson & Van Dongen, in press, for a review). In the ACT-R theory of memory, decay rate is arguably the parameter that most closely corresponds to encoding and rehearsal, as this parameter determines how much the previous exposures to knowledge will affect future retrievals. While there is no conclusive evidence in the literature to attribute either encoding or retrieval deficits to the observations, the current modeling helps support the claim that decreased alertness affects encoding.

A useful future extension to the proposed mechanism for predicting the effects of alertness on learning would be to account for the inter-session intervals. Currently the model does not specifically take into account the 2 to 26 hour intervals between consecutive sessions, which is problematic if we want to generalize the model to tasks in which the time between sessions varies. Incorporating mechanisms proposed in previous modeling to account for inter-session intervals (Anderson, et al., 1999) or practice spacing effects (Jastrzembski & Gluck, 2009) may allow the current model to predict these inter-session intervals.

Conclusion

Performance variations based on alertness have both theoretical and real-world importance. The present results illustrate how specific cognitive processes may be affected by circadian rhythms, and have implications for task training and performance in real-world contexts.

The cognitive modeling presented here illustrates how learning rates may be impaired at night, during the nadir of circadian rhythms. Because degraded learning has potential consequences that extend beyond the immediate situation, brief transitions to day shift may not result in immediate recovery. While the benefit in response time was fairly small in this study (300 ms), the modeling suggests that the effects of learning under conditions of lower alertness may accumulate over time and thus the benefit of training during the day will grow. Moreover, tasks in which exposures to declarative facts are less frequent, as seen in many real world tasks, are expected to encounter an even greater effect of decreased alertness due to a greater time between rehearsals *and* a greater (exponential) decay rate.

Several mechanisms were explored to explain the observed night shift response time decrement. Some mechanisms that have been used previously to explain observed decrements of alertness could not explain the results found in this research. We do not find this outcome particularly troublesome, or even surprising. Rather, in the current study and others, the tasks were specifically selected to ascertain the various ways in which reduced alertness may affect performance on particular mechanisms within the ACT-R architecture.

Our goal is to identify a general set of mechanisms to account for the ways in which variations in alertness impact various components of cognitive functioning. Focusing on laboratory tasks allows us to better isolate various components and evaluate particular computational mechanisms. Such an understanding is necessary in order to predict performance in more complex tasks where various cognitive functions, and mechanisms, interact in complex ways. This represents the focus of this research in the long term (e.g. Gunzelmann & Gluck, 2009; Gunzelmann, Moore, Salvucci, & Gluck, 2009; Tucker et al., 2010).

Acknowledgments

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. The research was supported in part by the Air Force Research Laboratory's Warfighter Readiness Research Division and grants 07HE01COR, 09RH06COR, and 10RH04COR from the Air Force Office of Scientific Research (AFOSR). The first author was supported by an appointment to the Postgraduate Research Participation Program at the U.S. Air Force Research Laboratory administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and USAFRL. The experimental research was supported by FMCSA grant DMC75-07-D-0006. The fourth author was supported by AFOSR grant FA9550-09-1-0136.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.

Anderson, J. R., Fincham, J. M., & Douglass, S. (1999). Practice and retention: A unifying analysis. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 25*(5), 1120-1136.

Åkerstedt, T. (1988). Sleepiness as a consequence of shift work. *Sleep, 11*(1), 17-34.

Bethell-Fox, C. E. & Shepard, R. N. (1988). Mental rotation: Effects of stimulus complexity and familiarity. *Journal of Experimental Psychology: Human Perception & Performance, 14*(1), 12-23.

Fu, W. T., Gonzalez, C., Healy, A. F., Kole, J. A., & Bourne Jr, L. E. (2006). Building predictive human performance models of skill acquisition in a data entry task. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* 1122-1126.

Gunzelmann, G., Anderson, J. R., & Douglass, S. (2004). Orientation tasks with multiple views of space: Strategies and performance. *Spatial Cognition and Computation*, *4*(3), 207-253.

Gunzelmann, G. & Gluck, K. A. (2009). An integrative approach to understanding and predicting the consequences of fatigue on cognitive performance. *Cognitive Technology*, *14*(1), 14-25.

Gunzelmann, G., Gluck, K. A., Kershner, J., Van Dongen, H. P. A., & Dinges, D. F. (2007). Understanding decrements in knowledge access resulting from increased fatigue. *Proceedings of the Annual Meeting of the Cognitive Science Society,* Austin, TX, 329-334.

Gunzelmann, G., Moore Jr, L. R., Salvucci, D. D., & Gluck, K. A. (2009). Fluctuations in alertness and sustained attention: Prediction driver performance. *Proceedings of the International Conference of Cognitive Modeling,* Manchester, UK.

Halverson, T., Gunzelmann, G., Moore Jr, L. R., & Van Dongen, H. P. A. (in press). The effects of work shift and strategy on an orientation task. *Proceedings of the Annual Meeting of the Cognitive Science Society.*

Jackson, M. L. & Van Dongen, H. P. A. (in press). Cognitive effects of sleepiness. In M. Thorpy & M. Billiard (Eds.), *Sleepiness.* Cambridge University Press.

Jastrzembski, T. S. & Gluck, K. A. (2009). A formal comparison of model variants for performance prediction. *Proceedings of the International Conference of Cognitive Modeling,* Manchester, England.

Kase, S. E., Ritter, F. E., & Schoelles, M. (2009). Caffeine's effect on appraisal and mental arithmetic performance: A cognitive modeling approach tells us more. *Proceedings of the International Conference on Cognitive Modeling,* Manchester, England, 174-179.

Lim, J. & Dinges, D. F. (2008). Sleep deprivation and vigilant attention. *Annals of the New York Academy of Science*, *1129*, 305-322.

Mednick, S., Nakayama, K., & Stickgold, R. (2003). Sleep-Dependent learning: A nap is as good as a night. *Nature Neuroscience*, *6*(7), 697-698.

Pavlik, P. I. & Anderson, J. R. (2005). Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science*, *29*(4), 559-586.

Tucker, A. M., Whitney, P., Belenky, G., Hinson, J. M., and van Dongen, H. P. A. (2010). Effects of sleep deprivation on dissociated components of executive functioning. *Sleep, 33*(1), 47-57.

Van Dongen, H. P. A. & Dinges, D. F. (2005). Sleep, circadian rhythms, and psychomotor vigilance. *Clinics in Sports Medicine,* 24(2), 237-249.

Walker, M. P., Brakefield, T., Morgan, A., Hobson, J. A., & Stickgold, R. (2003). Practice with sleep makes perfect: Sleep-Dependent motor skill learning. *Neuron*, *35*(1), 205-211.

Guidelines for Developing Explainable Cognitive Models[1]

Maaike Harbers[1,3], Joost Broekens[2], Karel van den Bosch[3], John-Jules Meyer[1]

[1]Utrecht University, The Netherlands; [2]TU Delft, The Netherlands; [3]TNO Human Factors, The Netherlands
{maaike,jj}@cs.uu.nl, joost.broekens@gmail.com, karel.vandenbosch@tno.nl

Abstract

Cognitive models can be used to generate the behavior of virtual players in simulation-based training systems. To learn from such training, the virtual players must display realistic human behavior, and trainees need to understand why the other players behave the way they do. This understanding can be achieved by explaining the underlying reasons for the virtual players' behavior. In this paper, it is discussed how to design cognitive models in such a way that they are able to explain the behavior they generate. Three users studies were carried out to assess what type of explanations are useful for training, and how that relates to cognitive model design. Several guidelines for developing explainable cognitive models are proposed.

Keywords: Explanation, Cognitive modeling, Task analysis, Virtual training.

Introduction

Virtual training systems are increasingly used for training of complex tasks such as fire-fighting, crisis management, negotiation and social skills. To create valuable learning experiences, the virtual characters in the training scenario, e.g. the trainee's colleagues, opponents or team members, must display realistic behavior. Realistic behavior can be ensured by letting humans play these roles. However, the characters in virtual training systems often have specialist tasks which can only be played by experts, and human experts are often scarcely available. Alternatively, required human behavior can be represented in cognitive models, which gives trainees the opportunity to train whenever and wherever they like (Heuvelink, 2009).

A valuable learning experience requires more than interaction with virtual players displaying realistic behavior. To learn from training, trainees must (eventually) understand the behavior of the other players. Instructors can explain the motives behind other players' behavior, but that would reintroduce the availability problems with experts just mentioned. Preferably, cognitive models representing human behavior also have the ability to explain that behavior.

There are several systems providing explanations about non-human player behavior in virtual training systems, e.g. Debrief (Johnson, 1994), XAI I (Van Lent, Fisher, & Mancuso, 2004) and XAI II (Gomboc, Solomon, Core, Lane, & Lent, 2005; Core et al., 2006). However, none of these systems obtain their explanations directly from the cognitive models of virtual players. The XAI I system only provides explanations about the physical states of virtual players, e.g. their location and health. Debrief determines what must have been the beliefs of a virtual player, but does not have access to its actual beliefs. XAI II gives explanations in terms the underlying motivations of virtual players if those are represented in simulation, but this is often not the case. Moreover, as far as we know, the explanations of these systems are not empirically evaluated.

We advocate an approach that connects behavior generation and explanation. In other words, the cognitive models used to generate behavior can also be used to explain that behavior. The models are not necessarily similar to human reasoning, as long as they generate useful explanations. In this paper, we discuss three explorative studies in which users evaluate explanations generated by explainable cognitive models on their usefulness for learning. Based on the results, we present guidelines for designing explainable cognitive models.

The paper is organized as follows. First, we discuss what is known about how people explain behavior. Second, we introduce an approach for explainable cognitive models. Then, we describe three user studies evaluating explanations of these models, and discuss the results. From this discussion, we abstract guidelines for modeling and explaining virtual player behavior. We end the paper with a conclusion and suggestions for future research.

Explaining behavior

Keil provides an extensive overview of explanation in general, in which he categorizes explanations according to the causal patterns they employ, the *explanatory stances* they invoke, the domains of phenomena being explained, and whether they are value or emotion laden (Keil, 2006). Humans usually understand and explain their own and others' behavior by adopting the *intentional stance*.

Dennett distinguishes three explanatory stances: the mechanical, the design, and the intentional stance (Dennett, 1987). The mechanical stance considers simple physical objects and their interactions, the design stance considers entities as having purposes and functions, and the intentional stance considers entities as having beliefs, desires, and other mental contents that govern their behavior. The intentional stance is closely related to the notion of *folk psychology*. Folk psychology refers to the way people think that they think, and determines the language they use to describe their reasoning about actions in everyday conversation (Norling, 2004).

Attribution theory is one of the most important theories on people's behavior explanations, and focuses on the vari-

[1]This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

ous causes that people assign to events and behavior (Heider, 1958; Kelley, 1967). External attribution assigns causality to factors outside of the person, e.g. the weather. Internal attribution assigns causality to factors within the person, e.g. own level of competence. Related to attribution theory is the concept of *explanatory style*, i.e. people's tendency to explain causes of events in particular ways (Buchanan & Seligman, 1995). People with a negative explanatory style believe that positive events are caused by things outside their control and that negative events are caused by them. People with a positive explanatory style, in contrast, believe that positive events happened because of them and that negative events were not their fault. Explanatory style is part of someone's personality.

Attribution theory is criticized for not making a distinction between the explanation of intentional and unintentional behavior (Malle, 1999). In reaction, Malle provided a framework with different explanation modes. One explanation mode considers explanations about unintentional behavior, and three explanation modes consider explanations about intentional behavior: reason, causal history, and enabling factors explanations. Reason explanations are most often used and consist of beliefs and goals, causal history explanations explain the origin of beliefs and goals, and enabling factors explanations consider the capabilities of the actor.

A lot of research on explaining computer program behavior has been done in the field of expert systems (Swartout & Moore, 1993). Usually, outcomes like diagnoses or advices are explained by the steps that lead to it, e.g. the rules that were applied. It was found that the purpose of explanation has to be taken into account during system design. The information needed in explanations must be present, even though not necessary for the generation of behavior.

Putting these findings into the perspective of cognitive modeling and virtual training: trainees should get to understand the intentional behavior of virtual players. Different explanation theories use different terms for people's explanations of (intentional) human behavior. But whether called intentional, folk or reason explanations, they all refer to explanations in terms of mental concepts like beliefs, intentions and goals. Furthermore, when a cognitive model has to determine the behavior of a virtual player, it must be executable, e.g. by implementing the model in a cognitive architecture. From explanation research on expert systems we learned that the concepts needed for explanation must be present in the design. Consequently, to develop explainable cognitive models, concepts like motivations, beliefs, and goals need to be explicitly represented in the model.

An explainable cognitive model

Virtual players in training systems usually perform relatively well defined tasks. We therefore represent their behavior in the form of task hierarchies. Hierarchical task analysis is a well established technique in cognitive task analysis, and connects internal reasoning processes to external actions (Schraagen, Chipman, & Shalin, 2000). A task hierarchy has one main task, which is divided into subtasks, which are divided into subtasks, etc. Subtasks that are not divided are actions that can directly be executed in the environment. Adoption conditions are connected to each subtask, specifying the conditions under which a subtask can be adopted. Sardina et al pointed out the similarities between task hierarchies and BDI (Belief Desire Intention) models (Sardina, De Silva, & Padgham, 2006). The tasks and adoption conditions in a task hierarchy can be seen as goals and beliefs, respectively (see Figure 1). In earlier work we have elaborated the use of goal hierarchies for the representation virtual player behavior, and shown how these models can be implemented in a BDI (Beliefs Desire Intention) architecture, and thus be made executable (Harbers, Bosch, & Meyer, 2009a).

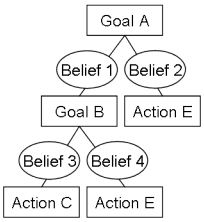


Figure 1: Example of a goal hierarchy.

There are four goal-subgoal relations: an *all* relation means that all subgoals must be achieved to achieve a goal, *one* means that exactly one subgoal must be achieved to achieve a goal, *seq* means that all subgoals must be achieved in a particular order to achieve a goal, and *if* means that a subgoal must only be achieved under certain conditions, i.e. when the player has certain beliefs. These relations yield different *action types*, i.e. the relation of an action to its parent goal.

An action can be explained by the goals and beliefs responsible for that action. However, providing the whole trace of beliefs and goals delivers long explanations with irrelevant information (Keil, 2006), in particular, with big goal hierarchies. Instead, a selection of 'explaining elements' can be provided to the trainee. For example, Action C in Figure 1 could be explained by Goal B, Goal A, belief 3, belief 1 or Action E (provided that E must follow C). More general, an action can be explained by different *explanation types*, respectively, the goal directly above an action (G+1), the goal two levels above an action (G+2), the beliefs one level above an action (B+1), the beliefs two levels above an action (B+2), and the goal or action that will be achieved after an action (Gnext).

Theories on human behavior explanation do not describe which explaining mental concepts should be part of an explanation. Malle's framework, for instance, does distinguish beliefs and goals in reason explanations, but does not (yet) describe in which situations which type is used more often (Malle, 1999). We performed three user studies to investigate which explanation types are considered useful to increase understanding of the training task. In particular, we investigated which explanation type is preferred for which action type. Our hypotheses are related to explanation stance,

length and type: 1) explanations in terms of beliefs and goals are appropriate for explaining virtual player behavior, 2) preferred explanations are relatively short and contain a selection among explaining beliefs and goals, and 3) preferred explanation type depends on the type of the action to be explained.

Three user studies

In this section we will give overviews of Study 1 (Harbers, Bosch, & Meyer, 2009b), 2 (Harbers, Bosch, & Meyer, 2010) and 3 (Broekens et al., 2010), and then discuss the results together. Only the results that are relevant for the discussion in this paper are presented. In all studies, the subjects were provided with a training scenario, and then asked to provide, select or judge explanations for several of the actions of the player(s) in the scenario. The independent variable in the studies is action type (actions with an *all*, *seq*, *one* or *if* relation to their parent) and the dependent variable is preferred explanation type (G+1, G+2, B+1, B+2, or Gnext). The explanations presented to the subjects were generated by implemented cognitive models of the virtual players.

Study 1: Onboard firefighting

Domain and task. The domain was onboard firefighting. The role to be trained was that of Officer of the Watch (OW), the person in command when there is a fire aboard a ship.

Subjects. The subjects (n=8) were instructors of the Royal Netherlands Navy and all expert on the training task.

Material. We used the CARIM system, a virtual training system for onboard firefighting (Bosch, Harbers, Heuvelink, & Van Doesburg, 2009). Three of the characters in the training scenario were modeled and implemented. The implementation was done in the programming language 2APL (Dastani, 2008). Questionnaires were administered to the subjects.

Procedure. Subjects played one scenario (approx 20 minutes), using the CARIM system, in which they were confronted with a fire aboard a Navy ship. Subsequently, they received a list with 12 actions of players in the scenario, and were asked to explain them in a way they considered useful for increasing trainees' understanding. Then, they received the same list of 12 actions, this time with four explanation alternatives (G+1, G+2, B+1, B+2) for each action. The subjects were asked to indicate which of the alternatives they considered most useful for increasing trainees' understanding.

Results. Regarding the first part of the questionnaire, we counted the number of elements in each of the subjects' own explanations, where an element is a goal, a fact, etc. Of the 88 explanations in total, 62 contained 1 element and 26 contained 2 elements. Furthermore, we categorized the elements in the subjects' explanations in different mental concepts. We were able to categorize all elements as either a belief or a goal: 52 beliefs and 62 goals. Table 1 shows the results of the second part of the questionnaire, the multiple choice questions. The agreement among the subjects for these results differed per action: for 5 actions at least 75% of the subjects preferred the same explanation, for 6 actions at least 50%, and for 1 action there was no explanation which at least 50% of the subjects preferred.

Action type	**Explanation type**			
	G+1	G+2	B+1	B+2
All (3 actions)	33%	50%	13%	4%
Seq (9 actions)	51%	21%	28%	0%

Table 1: Percentages of preferred explanation types per action type (n=8).

Study 2: Firefighting

Domain and task. The domain of this study was civil firefighting, and the role of the trainee was leading firefighter.

Subjects. The subjects (n=20) in Study 2 were unfamiliar to the training task. An advantage of non-expert subjects is that they do not have to imagine how useful the provided explanations are for understanding the training task. Instead, they can introspect to determine which explanations they consider useful. A disadvantage, on the other hand, is that non-experts cannot be expected to provide useful explanations for expert task actions themselves.

Material. A cognitive model of a leading firefighter was developed and implemented, again in 2APL. Questionnaires were used for the evaluation.

Procedure. The subjects were briefed about the training scenario, which involved a fire in a house. Subsequently, they received a list of 16 actions of the leading fire-fighter in the scenario with each four explanation alternatives (G+1, G+2, B+1, and Gnext). They were asked to indicate which explanation they considered most useful for understanding the task of leading fire-fighter.

Action type	**Explanation type**			
	G+1	G+2	B+1	Gnext
All (5 actions)	25%	16%	50%	9%
One (4 actions)	8%	8%	85%	0%
Seq (4 actions)	43%	14%	34%	10%
If (3 actions)	2%	2%	97%	0%

Table 2: Percentages of preferred explanation types per action type (n=20).

Results. Table 2 gives an overview of the results. For 7 of the actions at least 75% of the subjects preferred the same explanation, for 8 actions at least 50%, and for 1 action there was less than 50% agreement.

Study 3: Cooking

Domain and task. The domain of this study was cooking, and the training task was making pancakes. We purposely selected a simple training task, so that it was easy to find people

that could be considered experts.

Subjects. The subjects (n=30) were all familiar to this task.

Material. A cognitive model of a cook able to make pancakes was developed. The model was implemented in the programming language GOAL (Hindriks, 2009). Again, questionnaires were used for the evaluation.

Procedure. First, the subjects were briefed about the training scenario. Subsequently, they were asked to explain 11 of the cook's actions as they would to a student cook. Next, the subjects had to rate given explanations for all the 11 actions on their naturalness and usefulness on a scale of 1 to 5. The subjects were divided over condition 1, 2 and 3 in which they had to rate explanations of type G+1, B+1 and Gnext, respectively. In the last part of the questionnaire the subjects were shown the underlying goal hierarchy of the virtual player, and they were asked to indicate in the hierarchy by which beliefs and/or goals they would use to explain each of the 11 actions.

Results. The results of the subjects rating the usefulness of given explanations are shown in Table 3 (one of the actions was excluded from the analysis). The numbers are the average ratings of 10 subjects on 3 or 4 actions. The average

Action type	**Explanation type**		
	G+1	B+1	Gnext
All (3 actions)	3.2	2.5	3.4
One (3 actions)	3.0	2.4	2.0
Seq (4 actions)	2.9	2.8	1.8

Table 3: Average usefulness scores (scale 1-5) of action type per explanation type (n=30, n=10 per condition).

number of goals and/or beliefs that the subjects selected in the goal hierarchy for using in an explanation themselves was 1.7. One of the 30 subjects scored very high, and without this subject the average number of selected elements was 1.5.

Discussion

In this section we discuss the results of the user studies aiming to extract guidelines for developing and explaining cognitive models. The discussion is organized according to the three hypotheses concerning explanation stance, length and type.

From literature we learned that people adopt the intentional **explanatory stance** when they explain (intentional) human behavior. In other words, human(-like) behavior is explained by mental concepts such as beliefs and goals. The results of Study 1 show that it is possible to categorize the subjects' explanations in beliefs and goals, i.e. they are *compatible* with the intentional stance (we do not claim that this is the only way to categorize these explanations). In Study 3, the subjects' explanations were not categorized systematically, but an examination of the explanations provides a similar picture. Thus, the results confirm that people explain human-like virtual player behavior by the underlying beliefs and goals.

The results confirm our hypothesis that preferred explanations are relatively short. We expressed **explanation length** by the number of elements in an explanation, where an element is a fact, a goal, etc. In Study 1, the subjects' explanations had an average length of 1.3 elements, and in Study 3 the subjects selected an average of 1.7 elements from the goal hierarchy (1.5 if one outlier is eliminated from the data). The lower average in Study 1 might be due to the fact that the subjects had to write down complete explanations, whereas in Study 3 they only had to mark numbers of elements. So as expected, people's explanations about virtual player behavior usually only contain one or two elements.

As the results discussed so far confirm that explanations contain a selection of beliefs and goals, it makes sense to examine people's preferred **explanation type**. In Study 1, except for explanations of type B+2, all explanation types (G+1, G+2, B+1) were sometimes considered most useful by more than 50% of the subjects. In Study 2, for actions of type *one* and *if*, explanations containing a belief (B+1) were clearly preferred, and for actions of type *all* and *seq*, also explanations of other types (G+1 and G+2) were sometimes preferred by more than 50% of the subjects. These results are consistent with Study 1, in which only *all* and *seq* actions were examined. In Study 3, unlike Study 2, for all action types, explanations of type G+1 were on average rated higher than those of type B+1. Like in Study 2, for action types *one* and *seq*, Gnext explanations received relatively low ratings, and for actions of type *all*, they were highly rated. The usefulness of type Gnext explanations is closely related to the underlying cognitive model, which will be discussed in the next section. Interestingly, in the last part of Study 3, subjects often selected both a belief and a goal as their preferred explanation.

A remarkable difference between Study 1 and 3 on the one hand, and Study 2 on the other hand is that goal-based explanations were generally stronger preferred in the former, and belief-based explanations in the latter. A possible reason is that the subjects in Study 2 were unfamiliar, and those in Study 1 and 3 familiar with the training task. Data suggest that, on average, beliefs carry more idiosyncratic information and are harder to infer than goals (Malle, 1999). For subjects unfamiliar with a training task, belief-based explanations may provide more information underivable from the context than goal-based explanations. And expert subjects may not realize that goal-based explanations are easier to infer for trainees. Another explanation is that experts, more than non-experts, focus on the bigger picture of a virtual character's behavior. The subjects in Study 1 may be expected to know what would help trainees as they were instructors and had, besides being expert on the training task, didactical knowledge.

To conclude, action type is sometimes, but not always predictive for preferred explanation type. Of all studies, only Study 3 indicates to what extend explanations are preferred. The highest usefulness scores on action type *all*, *one* and *seq* are 3.4, 3.0 and 2.9, respectively. The scores are not low (all above the average of 2.5), but not very high either. In the experiments, we only provided subjects with explanations con-

taining one element, but the results seem to indicate that both beliefs and goals carry important information.

Modeling and explanation guidelines

Though the results of the three studies give no conclusive evidence, they provide directions for modeling and explaining virtual player behavior. In this section we present a set of guidelines for designing and explaining cognitive models.

The design and explanation of cognitive models are closely related in our approach. Though a virtual player's beliefs and goals remain unknown for users when a cognitive model is executed, they become visible when its behavior is explained. Thus, the elements in a cognitive model determine the content of its explanations. **Guideline**: the goals and beliefs in a goal hierarchy should be meaningful. Furthermore, two cognitive models with different underlying structures may display the same behavior, but generate different explanations. Figure 2, for instance, shows two possible positions of action E in a goal hierarchy. When both relations in this hierarchy are of the type *seq*, the position of action E does not effect the model's observable behavior, but it may influence they way it is explained, e.g. when explanations of the type G+1 are generated. Of course, developing a cognitive model always

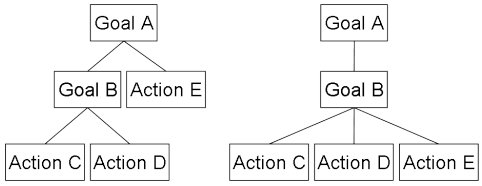


Figure 2: Same behavior, different explanations.

should be done with care, but as illustrated, this holds for explainable cognitive models in particular. **Guideline**: careful attention should be paid to the internal structure of the goal hierarchy. Though obvious, these two guidelines are crucial for developing useful explainable cognitive models.

In the previous section, we concluded that both beliefs and goals carry important information for explanations. The results showed that beliefs directly above an action (B+1) were considered most useful for explaining that action. Regarding goal-based explanations, the studies are less conclusive; several goal-based explanation types were considered useful (G+1, G+2 and Gnext) for different actions. But all together, goal-based explanations of type G+1 were most often preferred and highest rated. Moreover, people tend to use explanation types B+1 and G+1 together. **Guideline**: explanations should contain the belief(s) B+1 and the goal G+1.

The guidelines presented so far are general for all action types and supported by the results of all three studies. More specific guidelines that take action type into account can improve the default explanations. In the remainder of this section we will propose two additional, more specific guidelines.

In some cases an explanation of type Gnext can be added to the default explanation of G+1 and B+1. In contrast to G+1 and G+2 explanations, Gnext explanations do not contain goals from a particular level above the action. The level of the Gnext goal depends on the relations in the goal hierarchy. Here again, the usefulness of a Gnext explanation strongly depends on the underlying cognitive model. Consider, for instance, the two goal hierarchies in Figure 3. Goal B and C can be modeled as two neighboring goals or as goal and subgoal, e.g. when goal A, B and C represent *Report to head officer*, *Go to the head officer* and *Report new information*, respectively. In the first case, achieving goal B enables the achievement of goal C, and in the latter, goal C is achieved by achieving B. In Study 3, Gnext explanations were consid-

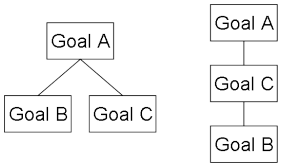


Figure 3: Neighbors or parent and sub-goal.

ered useful for actions of type *all*, where for all these *all* type actions it holds that their parents had a *seq* relation to their parents. **Guideline**: for actions of type *all*, when their parent goal has a *seq* relation, the explanation should contain the goal Gnext besides B+1 and G+1. Addition of a Gnext goal to the explanation may also be useful for other action types, but we have no evidence for that.

Another exception to the default rule concerns actions of the type *one*. The left side of Figure 4 represents a situation where action B is followed by action C or D, for example, the action *Take money* is followed by either *Cycle to the shop* or *Drive to the shop*. Action C and D are explained by goal A (G+1), e.g. *Buy ingredients*. However, a goal can only have one relation to its subgoal/actions, so the goal hierarchy in the left side is not allowed. The right side of Figure 4 shows how this situation should be represented. Goal A has a relation *seq* to its children, and a new goal X is introduced, e.g. *Go to the shop*, with a relation *one* to its children. Now, when action C and D are explained by their parent goal X, the explanation is not informative (I cycle to the shop because I want to go to the shop). In this case, it would be better to provide goal A as an explanation (I cycle to the shop because I want to buy ingredients). Although it may result in redundant goal-subgoal

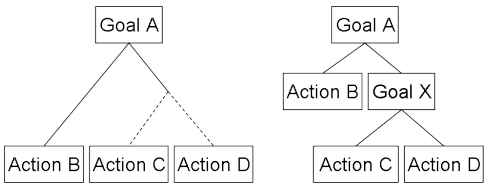


Figure 4: Explanation of actions with a one relation.

relations, we believe that from an explanation point of view a goal should have only one relation to its subgoals, as this simplifies interpretation of the cognitive model. **Guideline**: to explain actions of type *one*, instead of goal G+1, goal G+2 should be provided (i.e. B+1 and G+2).

Conclusion

In this paper we analyzed the results of three user studies investigating people's preferred explanations of virtual player behavior. From the analysis, we extracted a set of guidelines for developing and explaining cognitive models. In general, modeling should be done carefully, and by default, an action should be explained by the goal and belief directly above the action, i.e. explanation types G+1 and B+1. In addition, we introduced two guidelines for specific action types, which show how default explanations can be improved by providing extra or other elements in the goal hierarchy. More experimentation is needed for introducing more of these specific guidelines.

Another way to improve the explanations is by extending the cognitive model, for instance, by adding beliefs. Beliefs can contain information about the environment, e.g. resources that are available or events that just occured. Such beliefs are useful in particular and most often connected to *if* and *one* type actions. Beliefs can also contain information about internal reasoning processes, e.g. the given action is not yet executed, or a preceding action is executed. Such beliefs are more often connected to *all* and *seq* type actions. In these cases, it can be useful to add extra beliefs containing background information as adoption conditions. These background beliefs are always believed by the virtual player, so they do not effect the player's observable behavior, but they do add useful information to explanations.

There are many other directions in which this work can be extended. For instance, the cognitive models can be extended with emotions, a user model in which the trainee's knowledge is modeled can be used to select explanations, and the success of the approach in other domains can be examined. In future work we will first validate the present approach by comparing understanding of played training scenarios of trainees who did and did not receive explanations about virtual player behavior.

References

Bosch, K. Van den, Harbers, M., Heuvelink, A., & Van Doesburg, W. (2009). Intelligent agents for training on-board fire fighting. In *Proc. of the 2nd internat. conf. on digital human modeling* (p. 463-472). San Diago, CA: Springer Berlin/Heidelberg.

Broekens, J., Harbers, M., Hindriks, K., Bosch, K. Van den, Jonker, C., & Meyer, J.-J. (2010). *Do you get it? User evaluated explainable AI.* To appear.

Buchanan, G., & Seligman, M. (1995). *Explanatory style*. Erlbaum.

Core, M., Traum, T., Lane, H., Swartout, W., Gratch, J., & Van Lent, M. (2006). Teaching negotiation skills through practice and reflection with virtual humans. *Simulation*, *82*(11), 685-701.

Dastani, M. (2008). 2APL: a practical agent programming language. *Autonomous Agents and Multi-agent Systems*, *16*(3), 214-248.

Dennett, D. (1987). *The intentional stance*. MIT Press.

Gomboc, D., Solomon, S., Core, M. G., Lane, H. C., & Lent, M. van. (2005). Design recommendations to support automated explanation and tutoring. In *Proc. of BRIMS 2005.* Universal City, CA..

Harbers, M., Bosch, K. Van den, & Meyer, J.-J. (2009a). A methodology for developing self-explaining agents for virtual training. In Decker, Sichman, Sierra, & Castelfranchi (Eds.), *Proc. of 8th int. conf. on autonomous agents and multiagent systems (aamas 2009)* (p. 1129-1130). Budapest, Hungary.

Harbers, M., Bosch, K. Van den, & Meyer, J.-J. (2009b). A study into preferred explanations of virtual agent behavior. In Z. Ruttkay, M. Kipp, A. Nijholt, & H. Vilhjlmsson (Eds.), *Proc. of IVA 2009* (p. 132-145). Amsterdam, Netherlands: Springer Berlin/Heidelberg.

Harbers, M., Bosch, K. Van den, & Meyer, J.-J. (2010). *Design and evaluation of explainable agents.* To appear.

Heider, F. (1958). *The psychology of interpersonal relations*. New York: John Wiley Sons.

Heuvelink, A. (2009). *Cognitive models for training simulations*. Unpublished doctoral dissertation, Vrije Universiteit Amsterdam, The Netherlands.

Hindriks, K. (2009). Multi-agent programming: Languages, tools and applications. In (p. 119-157). Springer.

Johnson, L. (1994). Agents that learn to explain themselves. In *Proc. of the 12th nat. conf. on artificial intelligence* (p. 1257-1263).

Keil, F. (2006). Explanation and understanding. *Annual Reviews Psychology*, *57*, 227-254.

Kelley, H. (1967). Attribution theory in social psychology. In D. Levine (Ed.), *Nebraska symposium on motivation* (Vol. 15, p. 192-240). Lincoln: University of Nebraska Press.

Malle, B. (1999). How people explain behavior: A new theoretical framework. *Personality and Social Psychology Review*, *3*(1), 23-48.

Norling, E. (2004). Folk psychology for human modelling: Extending the BDI paradigm. In *Third internat. joint conf. on autonomous agents and multi agent systems* (p. 202-209). New York, USA.

Sardina, S., De Silva, L., & Padgham, L. (2006). Hierarchical planning in BDI agent programming languages: A formal approach. In *Proceedings of aamas 2006.* ACM Press.

Schraagen, J., Chipman, S., & Shalin, V. (Eds.). (2000). *Cognitive task analysis*. Mahway, New Jersey: Lawrence Erlbaum Associates.

Swartout, W., & Moore, J. (1993). Second-generation expert systems. In (p. 543-585). New York: Springer-Verlag.

Van Lent, M., Fisher, W., & Mancuso, M. (2004). An explainable artificial intelligence system for small-unit tactical behavior. In *Proc. of IAAA 2004.* Menlo Park, CA: AAAI Press.

A Cognitive Model of Theory of Mind

Laura M. Hiatt (laura.hiatt.ctr@nrl.navy.mil)
J. Gregory Trafton (greg.trafton@nrl.navy.mil)
Naval Research Laboratory
Washington, DC 20375 USA

Abstract

It is generally well acknowledged that humans are capable of having a theory of mind (ToM) of others. We present here a model which borrows mechanisms from three dissenting explanations of how ToM develops and functions, and show that our model behaves in accordance with various ToM experiments (Wellman, Cross, & Watson, 2001; Leslie, German, & Polizzi, 2005).

Keywords: cognitive architectures; theory of mind

Introduction

The concept of "theory of mind" (ToM) refers to one's ability to infer and understand the beliefs, desires and intentions of others, given the knowledge that one has available; without it, people can be severely impaired in their ability to interact with others (Baron-Cohen, Leslie, & Frith, 1985). A large body of research has tried to explain how this critical ability functions by studying its development in children (Wellman et al., 2001), but has led to many contradictory accounts.

We have built a model that borrows ideas from various explanations of how ToM develops and functions to form a cohesive theory of ToM, and show that it produces behavior in accordance with various ToM experiments (Wellman et al., 2001; Leslie et al., 2005). While the similarities between a model's behavior and data is not a certain indicator of cognitive plausibility (Cassimatis, Bello, & Langley, 2008), it can distinguish between models that show performance and data fit (which, to us, are preferred) and models that do not.

Theories of the Theory of Mind

There are, in general, three competing views for how ToM takes place at a cognitive level. They are typically described in the context of "belief and desire" reasoning: ToM is the understanding that different people can have different beliefs, not all of which may be actually true; people also have internal desires that cause them to act in certain ways, physically, in the world. There is also a distinction between "true-beliefs," or beliefs that are true in the physical world, and "false-beliefs," which others may have but which are not actually true. The ability to understand a false-belief task, then, indicates evidence that a person can appreciate the distinction between the mind and the world (Wellman et al., 2001).

Conceptual change (commonly called theory-theory) is one possible explanation for ToM (Wellman et al., 2001). Theory-theorists believe that children learn a set of causal laws, or theories, about the beliefs and desires of people in general (Gopnik, 1993). Children then use these causal laws to explain behavior observed in others, to predict desires and behaviors, and to perform other related ToM tasks.

Simulation theory is a second view (Gallese & Goldman, 1998). It posits that when a person ("A") tries to understand another ("B"), A simulates what he/she would do in B's place, and attributes the result to B. More specifically, the theory states that humans perform ToM by representing the mental states of others, and then using their own decision-making systems to operate on these foreign mental states to predict others' behavior; similar processes can be used to explain observed behavior, making backward inferences. Gallese and Goldman (1998) describe the distinction between this and theory-theory as, while theory-theory is performed as a "'detached' theoretical activity," simulation theory involves attempting to mimic or impersonate the mental state of another.

A third body of literature posits that the mind has two separate mechanisms that work together to provide ToM (Leslie, Friedman, & German, 2004). The theory of mind mechanism (ToMM) allows people to generate and represent multiple possible beliefs. It is argued that this mechanism is fully functional in even very young children. The second mechanism provides a selection process (SP) that uses inhibition to reason about others' beliefs, such as inhibiting a true-belief to select a false-belief answer; this processing ability, it is argued, develops in children during the pre-school years. To describe how the mechanisms work together as "ToMM-SP" to provide ToM, the authors break it down into four steps: (1) identify candidate belief possibilities; (2) provide *a priori* weights to the candidates, with true-belief receiving the highest weight; (3) adjust the weights given the belief inquiry; and (4) select the highest-weighted candidate as the answer.

A variety of experiments, primarily in developing children, have led to a range of results that supports each of these theories. We describe next some of these experiments, followed by our interpretation of the data and our overall view of ToM.

Experiments in Developing Children

The majority of experiments in this area concerns false-belief tasks. Arguably, the most well-known false-belief task (and the one on which we focus in this paper) is the Sally-Anne task (Baron-Cohen et al., 1985), in which a child is shown a play with two characters, Sally and Anne (Figure 1). The true-belief answer (to where Sally believes the marble is) is that the marble is in Anne's box (the "TB box"), since that is where the marble actually is. In contrast, the correct answer is the false-belief answer, Sally's box (the "FB box").

Variations on the Sally-Anne task have also been explored. One is the *avoidance* false-belief task (which we shorten to "avoidance task"). In a sample set-up, the marble is replaced by a kitten that crawls between boxes while Sally is out of

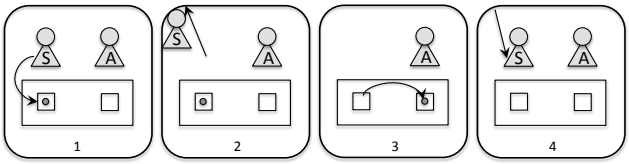


Figure 1: A diagram of the Sally-Anne task. A child watches while: (1) Sally puts a marble in her box; (2) Sally leaves the room; (3) Anne moves the marble to Anne's box; (4) Sally returns to the room. The child is then asked where Sally believes the marble is.

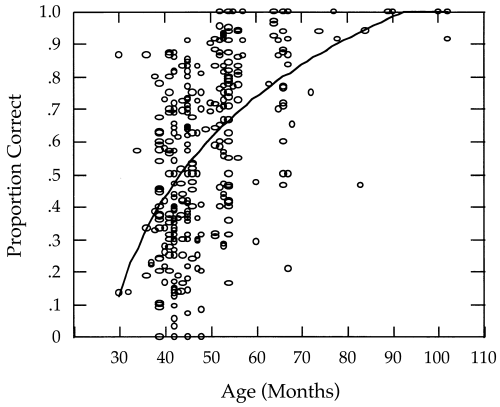


Figure 2: Results from (Wellman et al., 2001) showing a scatterplot of the results and best-fit curve.

the room; when Sally returns, she wants to put a piece of fish under the unoccupied box so that the kitten will not eat the food and get sick. Therefore, the correct answer to the question "where will Sally try to put the fish" is the TB box. This task involves not only identifying Sally's false belief, but also taking into account her avoidance desire to predict her behavior, presumably making the task more difficult.

To individually consider all the experiments in this area is nearly impossible. Instead, we focus on a meta-analysis that compiled a broad range of false-belief experiments (Wellman et al., 2001), and a more detailed experiment performed after the meta-analysis was compiled (Leslie et al., 2005). These two studies involve two developmental shifts that are believed to occur in children. The first is at about 3-4.5 years of age, when children go from being mostly incorrect to mostly correct on the standard false-belief task; this seems to correlate with the ability to recognize and identify beliefs of others. The second developmental shift is at around 4.5-6 years, when children go from having difficulty with the avoidance task to performing it mostly correctly; this seems to correlate with a child's ability to account for both beliefs and desires, and to use them to predict the behavior of others.

The meta-analysis performed by Wellman et al. (2001) provides three results pertinent to this paper. First, it identified several task components that were statistically insignificant, including the exact type of task being performed as well as the phrasing of the false-belief question (*e.g.*, whether it asks where Sally will look, what Sally believes, or what she will say). Other factors such as whether the characters in the task are dolls, photographs, etc., are also inconsequential. Our focus on the Sally-Anne task, then, and the exact experimental set-up we chose should not affect the validity of the results.

Secondly, several task components were identified as main effects, which improve performance but do not interact with age, including whether the child participated in the experiment (*e.g.*, helped to set up props), whether Sally's absence was explicitly emphasized, and in which country the experiment took place. We do not model such task variations.

Thirdly, the compiled results show a significant, if noisy, effect between age and the proportion of children that answered the false-belief query correctly ($p < 0.001$). Figure 2 shows the findings; it plots the results from each individual study considered, as well as the curve that best fits it. They found that at an age of about 44 months, the odds of answering correctly are even, or 1.0; then, the odds of being correct increase 2.94 times for every year. The linear regression model which considers only age is $y = -3.96 + 0.09 \cdot [\textit{age in months}]$, with $r^2 = 0.39$[1]. Their best statistical model, which had six variables (including age, the country in which the experiment took place, and child participation), yielded an R^2 of 0.55. The results clearly document the developmental shift that seems to happen between roughly 3 to 4.5 years of age where children go from being mostly incorrect to mostly correct on the standard false-belief task.

We also consider an experiment involving the avoidance task (Leslie et al., 2005). The experiment, performed with 4.75-year-olds on average, supports the belief that this task is more difficult than the standard task, and provides evidence for the second developmental shift. After several children were eliminated for failing the false-belief task, only 25% of 16 children correctly answered the query of "Where will Sally try to put the fish." The experiment showed, however, that by asking the question in terms of where the *first* place Sally will try to put the fish is, almost three times as many children (71%) passed the task; we refer to this as "look-first avoidance." Overall, the results suggest that children gain the ability to understand others' desires and their implications after they gain the capability to understand their beliefs.

Discussion of Experiments

The area of how children develop theory of mind remains controversial. One of the pressing questions that emerges from the literature is whether the various developmental shifts are due to learning concepts and causal laws (for clarity, we refer to this as "learning"), as the theory-theorists strongly posit, or due to increasing capabilities/functionality of mechanisms of the brain (we refer to this as "maturation"), as others argue. There is certainly evidence for both.

[1]This model transformed the proportion correct, p, via a logit transformation, $\ln(p/(1-p))$ where "ln" is the natural logarithm.

Leslie et al. (2004) argues that maturation of processing capabilities and resources, alone, can account for all ToM developments, and have designed reasonable process models (*e.g.*, ToMM-SP) demonstrating the idea's plausibility. Further evidence shows that the capabilities of specific mechanisms in the brain (such as selection processing and inhibition of beliefs) play a crucial role in ToM (German & Hehman, 2006; Carlson, Moses, & Claxton, 2004).

Wellman et al. (2001), however, makes several arguments for learning over maturation based on the results of the meta-analysis; specifically, the strong presence of task manipulations that act as main effects (*e.g.*, child participation). If maturation were true, presumably many task manipulations would interact with age since they should help younger children's processing competence more than older children's; however, they do not. The presence of such manipulations does, however, support conceptual change accounts. Overall, the authors argue that there is a potential interrelation of learning and maturation: children improve as they grow and acquire conceptual understanding of ToM but, within an age group, processing capabilities could be highly correlated with performance and could account for much of the variance.

Many of the above papers argue against simulation theory based on these results; however, much of the arguments are neither substantive nor well supported. Wellman et al. (2001) argues that, since children do not systematically err about their own false beliefs, simulation theory is not as plausible; however, this could easily be explained by children remembering their own past mental states. Leslie et al. (2004) simply says about simulation theory, "it is also hard to see a role for 'simulation' in accounting for this data... the mechanisms of theory of mind might simply figure out what one would do... there is currently no evidence that it is the first-person singular." The opposite argument could just as easily be made. Unfortunately, there are few developmental accounts available for simulation theory; (Harris, 1992) is an exception, and states that a child's inability to perform simulation early on may be due to memory limitations. In general, simulation theorists support their arguments as in (Gallese & Goldman, 1998), with the presence of mirror neurons that fire both when one views an action and when one performs it.

Overall, we agree in part with Wellman et al. (2001), who say that the ability of children to recognize false-beliefs in others is due to both learning and maturation, accounting for the first developmental shift we discussed where children gain the ability to recognize and predict beliefs in others. We argue, however, that the second developmental shift that occurs, which results in children being able to account for both beliefs and desires to predict another's behavior, is due to children gaining the ability to perform simulation. This accounts for 4.75-year-olds' inabilities to reliably answer the avoidance query: they are still in the middle of learning and maturing this ability. Note that this view is not necessarily incompatible, at the process level, with some of the others; *e.g.*, in highly complex situations, there is not much difference between Leslie et al. (2004)'s SP mechanism inhibiting everything that should *not* be used and operating only on what is left, and identifying pertinent beliefs and decision-making processes and subsequently using them in simulation.

Some recent experiments also suggest that very young children (15 months of age) can perform implicit (non-verbal) false-belief tasks (Onishi & Baillargeon, 2005). This supports the theory of processing mechanisms in the brain that work with false-beliefs and, further, suggests that the ability to recognize situations involving false-beliefs develops before the ability to explicitly reason about them. We anticipate further modeling work concerning this would be compelling.

Core Cognitive Architecture

As our core cognitive architecture we use ACT-R, a hybrid symbolic/sub-symbolic production-based system (Anderson, 2007). ACT-R consists of a number of modules, buffers and a central pattern matcher. Modules contain a relatively specific cognitive faculty typically associated with a specific region of the brain. For each module, there are one or more buffers that communicate directly with that module as an interface to the rest of ACT-R. At any point in time, there may be at most one symbolic item, or "chunk," in any individual buffer; the module's job is to decide when to put chunks into a buffer. Chunks are used to represent knowledge or memories related to any of the modules/buffers, and, in addition to symbolic information, contain subsymbolic information (*e.g.*, activation). The pattern matcher uses the contents of the buffers, if any, to match specific productions which, when fired, can modify the current contents of the buffers. Ties between competing productions are broken based on the productions' expected utilities, which can be initially set and adjusted via a reinforcement learning process; random noise can also be added in during execution to affect production selection.

The relevant modules of ACT-R to this paper are the intentional and declarative modules. In addition, ACT-R interfaces with the world through the visual, vocal, motor and aural modules. The open-source, robotic simulation environment Stage (Collett, MacDonald, & Gerkey, 2005) was used as the "world" of the model in order to enable fast model development and data collection.

ACT-R is able not only to learn new facts and rules, but also to learn which rule should fire (called utility learning in ACT-R). It accomplishes this by learning which rule or set of rules lead to the highest reward. ACT-R uses an elaboration of the Rescorla-Wagner learning rule and the temporal-difference algorithm (Fu & Anderson, 2006). This algorithm has been shown to be related to animal and human learning theory.

Any time a reward is given (*e.g.*, children being told they responded with the correct answer), a reward is propagated back in time through the rules that had an impact on the model getting that reward. Punishments are performed similarly.

Model Description

As stated above, our model is based on the conjecture that, as children grow, they learn and mature simultaneously; *i.e.*,

as they develop, they learn to take advantage of their maturing ability to select between competing beliefs. Further, we believe that being able to select between beliefs acts as a precurser for simulation, which allows people to use the beliefs and desires of others to predict and understand their behavior, and is ultimately what provides full-fledged ToM.

In our model, the Sally-Anne task takes place in the Stage simulator, which feeds the model visual information; *i.e.*, it passes the model visual locations to fixate on and, when attended to, what is at that location. This allows the model to "watch" the Sally-Anne play unfold. As the story unfolds, the model explicitly notes what happened (*e.g.*, Sally moved the marble into her box), and who saw it happen (*e.g.*, only Anne saw herself move the marble into her box). After the play completes, the model is asked several false-belief questions. If the model answers a question correctly, the model is rewarded; otherwise, it is punished.

We first describe the core mechanisms that enable ToM. Then, we describe how the model learns to effectively use these mechanisms (as well as develops the ability to use them). Although much of the description is in the context of the Sally-Anne task, as are our experiments, recall that this acts as a proxy for false-belief tasks in general and our results are not specific to this task (Wellman et al., 2001).

Theory of Mind Mechanisms

When its goal is to answer a query about someone's belief, a fully-developed model will answer the question similar to Leslie et al. (2004)'s ToMM-SP. As the story unfolds, the model generates possible beliefs for the marble's location; for the standard Sally-Anne task, then, this set is {sallys-box, annes-box}. The model first retrieves the TB answer because it has the highest activation. It realizes, however, that the answer is not correct since Sally does not know about it. To address this, it considers the various possible beliefs of the marble's location and, from these, it selects the most salient belief that Sally was known to be privy to, the FB box.

When faced with an avoidance task, a fully-developed model will first use the above process to select knowledge to use as input to its simulation. For the Sally-Anne avoidance task variant, the simulation's input would be the different boxes, as well as Sally's belief of the location of the kitten. All subsymbolic information of the knowledge, including activation levels, is preserved. The model next performs simulation by spawning a sub-model with: this input; access to the model's productions and cognitive resources; and the goal of deciding where to put the kitten (Kennedy, Bugajska, Harrison, & Trafton, 2009). Then, the sub-model can infer that, if Sally wants to put the fish under a box without the kitten, she will put it under the TB box.

Developmental Mechanisms

As stated, our model both learns and matures as it develops ToM. The learning mechanism is similar to standard ACT-R learning. The model begins with a production that answers false-belief queries simply by retrieving the belief chunk with the highest activation, and returning it. It can learn, however, to consider an alternate competing production that, upon the retrieval of the belief, considers whether the person the query is about knows about the belief. This production acts as the gateway to the selection process. Learning over time can teach the model to exclusively favor this production, as it ultimately leads to the correct answer. A similar process occurs when learning to perform simulation.

ACT-R does not normally model increasing functionality in the brain. In order to model maturation, therefore, we introduce the notion of a "maturation parameter." This parameter determines whether a model has the ability to fire certain sets of productions (*i.e.*, whether the model is mature enough to have that functionality). Since maturation is not an "all or nothing" concept, and happens gradually, the parameter acts as a guideline for how strong the model's abilities are at that moment. Any time the model attempts to fire a maturing set of productions, their availability is random according to the parameter (*e.g.*, if a randomly selected number is less than the parameter, the productions will be able to fire). Intuitively, maturation parameters should be correlated with age: the older the child, the higher the parameter.

In the case of selecting between different beliefs, the maturation parameter is called the "selection parameter" and determines the availability of the productions that select between beliefs. A model with a selection parameter of 0 would never be able to correctly select a false-belief as the involved productions would be unable to fire; a model with a selection parameter of 0.5 would be able to do so on half of its attempts; and a model would a selection parameter of 1 will always be able to fire the involved productions.

In the case of simulation, the model should be able to perform larger and larger simulations as it ages. This is in accordance with Harris (1992)'s view that children have difficulty performing simulation early on due to memory limitations. The "simulation parameter" determines the availability of the productions that perform simulation, given the size of simulation that is being attempted; for low sizes, the model is more likely to be able to do it, but at high sizes the model becomes overwhelmed and cannot process all the data, and so simulation is less likely. Specifically, any time a simulation is attempted, the probability that the simulation productions will be available is $\min(1, sp/s)$, where sp is the simulation parameter and s is the size of the attempted simulation. The size of the simulation is discussed in the subsequent section.

Modeling Developmental Progress

The model begins at approximately 2 years of age with the ability to generate multiple possible beliefs (Leslie et al., 2004). Model development mirrored the two ToM developmental phases. With respect to the first phase and the standard false-belief task, the model has a selection parameter of 0.5, but does not yet know to do the selection; *i.e.*, when it initially retrieves the most salient belief, it does not know to check whether Sally saw it and simply returns the belief. Of course, the most salient belief is likely the true-belief, and so

the model will be incorrect, leading to a punishment. This causes the model to begin to explore using the selection process. If the model is able to access that functionality (*i.e.*, if a number randomly selected at the time of the attempt is less than the selection parameter), it will attain the correct answer and receive reward; otherwise, it will default to returning the initially-retrieved belief, likely leading to punishment. Note that if this occurs, the productions leading to the selection attempt will incur lower expected utility, making it less likely that the model will attempt selection during the next trial.

Experience is simulated by engaging the model in false-belief trials and by slowly increasing the selection parameter. Therefore, as the model grows more experienced, it concurrently learns to utilize its selection mechanism and is able to more reliably perform selection: by the age of about 44 months (3.7 yrs), the selection parameter is up to 0.8, and by the age of 68 months (5.7 yrs) that parameter equals 0.95. Note that, as the selection parameter increases, so does the efficacy of learning, since more trials that attempt to select the false-belief do so successfully and receive positive reward. Learning was concentrated such that about 2 trials approximates 12 months of experience; the function relating learning trials to age was determined post hoc after comparing our results with those of (Wellman et al., 2001).

The second developmental component (concerning the avoidance task) occurs in an analogous way. Whenever the child successfully answers the standard false-belief task, it is queried about the look-first avoidance task (and, upon successfully answering that, is further queried on the standard avoidance task). The model first tries to calculate Sally's belief exactly as in the standard false-belief task; note that, especially at early ages, it may or may not be able to do so and may end up thinking about either the TB or FB box. Once a belief is in hand, the model initially does not know what to do with it; so it defaults to where it would put the kitten, the FB box, resulting in punishment. Over time, the model will start using the initial belief as input to simulation. If the model is able to simulate, it will return the box other than the belief; otherwise, it will again default to returning the FB box.

As mentioned, the model's ability to perform simulation is dependent on a simulation parameter, which in turn is dependent on the "size" of the simulation. For the look-first avoidance query, the simulation size is 1, as the child is being asked to predict Sally's actions only one step in the future. For the standard avoidance query, the simulation size is set to 3^2. When the model begins at age 2, the simulation parameter is 0 and so no simulation is possible; by age 56 months (4.7 yrs), it is 1, and by age 72 months (6 yrs), it is 5.

For all models, we kept most of the ACT-R parameter defaults. We did change the utility noise parameter (set at a moderate 1.0) to allow low-use productions to occasionally fire. Because the rate of learning is dependent entirely on the utility learning rate parameter (set at the default of 0.2), learning occurred quite quickly in this model. Utility learning rate could be scaled down substantially to match actual development and learning time. In order to do this correctly, it would be important to know approximately how often children encounter false-belief and avoidance tasks, and learn from them.

[2] Although this is ad hoc, with such limited data to match, a more pleasing parameter choice and justification is not possible.

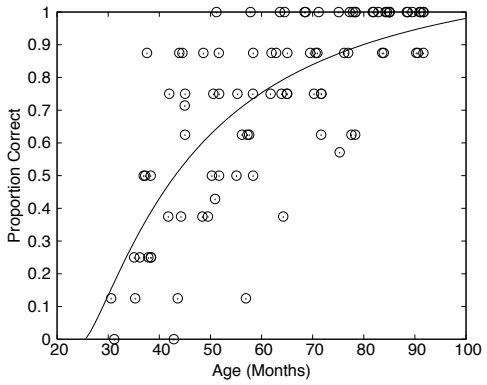


Figure 3: Model results showing a scatterplot of the standard false-belief results and best-fit curve.

Model Results

In our first experiment, corresponding to the first phase of model development, we started testing the model at age 32 months (2.7 yrs), and test roughly every 7 months until the model reaches around age 92 months (7.7 yrs), for a total of 10 tests. Each test period consisted of 8 repetitions of the Sally-Anne task, including all three queries. During these tests, learning is turned off in order to reliably test the model's abilities at that age. To simulate the variability of children's development, we randomly perturbed the models' starting ages around their *a priori* value of 2 years, selecting uniformly in the range [17, 31] months. This made the age of the models in our experiment comparable to the ages of the children in the meta-analysis (Wellman et al., 2001).

Figure 3 shows the results for the false-belief task, and plots each model's age during a test period against the proportion of correct answers the model gave during the test. The graph appears very similar, visually, to that of Figure 2, and shows a clear learning trend as well as noise which presumably stems from different maturation levels. Using Wellman et al. (2001)'s linear regression model (which considers only age) on this data, $r^2 = 0.51$ with a residual standard error of 1.73. This is considerably higher than their $r^2 = 0.39$. It also approaches the R^2 of their multi-variate model, 0.55. We argue, then, that our model is stronger since it is both a process model that learns to perform this task, as compared to a statistical model, and depends on fewer parameters.

Note that this curve is due to an interaction between the selection parameter increasing, and the model learning that attempting to select between beliefs often leads to the correct answer. We expect, therefore, that if the selection parameter increased more slowly, learning would be impeded and models' performance would not improve as quickly.

Our avoidance false-belief results were also compared to those of (Leslie et al., 2005), which showed that 71% of children around the age of 4.75 years could answer the look-first avoidance query but only 25% could answer the standard avoidance query. We were able to match these results, but further experimental data is needed in order to distinguish our parameterization from other valid possibilities.

Discussion

We have shown in this paper a cognitive model for theory of mind. Our model borrows ideas from all three main postulates of ToM to develop a cohesive explanation for how ToM functions. The model uses a selection process to identify the beliefs and knowledge others may have; then, to predict the desires and behaviors of others, it uses the identified concepts as input to its own decision-making mechanisms, simulating what the model would do in the other's place. This ToM functionality develops by concurrent learning and maturation of the required functional capabilities. The model was found to be a good match to existing data from developing children.

One of the strengths of this model is that it generalizes to many other types of false-belief and ToM tasks. The maturation parameters are very general, and can be applied with little change to other tasks. The same holds true for simulation; the cognitive mechanism which enables it can accept, and work with, any input. The learning of ToM in this paper is not as general, as it chooses between productions which are relatively task-specific; however, if the model were to have experience on a variety of ToM tasks, we expect that it would generalize what it learns into a broader concept.

Our work is also distinguished from previous work in cognitive architectures. Laird (2001)'s QuakeBot performs mental simulation of opponents to predict their behavior, for example, but to our knowledge their approach has not been matched against human cognitive data.

A future step is to explicitly address other observed ToM phenomena. One experiment added a third "neutral" box to the avoidance task, introducing a second correct answer, and had both children and adults as subjects (Leslie et al., 2004). The study showed that children have a bias towards the TB box, whereas adults have a bias towards the new neutral box. Our model does predict this phenomena for children, since the TB box is the correct box with the highest activation (it is the last box to receive a kitten, and it is identified as the true-belief of the kitten's location before the selection of beliefs begins), and so it is the answer that simulation will select. As far as the results for adults, we believe that with further learning, simple simulations can be avoided in favor of general, learned inference rules. In this case, therefore, adults are simply returning an answer that is true from anyone's perspective. The paper describes further experiments that our model can predict, but that is outside the scope of this paper.

Acknowledgements

This work was supported by the Office of Naval Research under funding document N0001409WX20173 to JGT. The views and conclusions contained in this document should not be interpreted as necessarily representing the official policies of the U. S. Navy.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

Baron-Cohen, S., Leslie, A. M., & Frith, U. (1985). Does the autistic child have a "theory of mind"? *Cognition*, *21*.

Carlson, S. M., Moses, L. J., & Claxton, L. J. (2004). Individual differences in executive functioning and theory of mind: An investigation of inhibitory control and planning ability. *Journal of Experimental Child Psychology*, *87*, 299-319.

Cassimatis, N. L., Bello, P., & Langley, P. (2008). Ability, breadth, and parsimony in computational models of higher-order cognition. *Cognitive Science*, *32*(8), 1304-1322.

Collett, T. H. J., MacDonald, B. A., & Gerkey, B. P. (2005). Player 2.0: Toward a practical robot programming framework. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA '05).*

Fu, W., & Anderson, J. (2006). From recurrent choice to skill learning: A model of reinforcement learning. *Journal of Experimental Psychology: General*.

Gallese, V., & Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, *2*(12).

German, T. P., & Hehman, J. A. (2006). Representational and executive selection resources in 'theory of mind': Evidence from compromised belief-desire reasoning in old age. *Cognition*, *101*, 129-152.

Gopnik, A. (1993). How we know our minds: The illusions of first person knowledge of intentionality. *Behavioral and Brain Sciences*, *16*, 1-14.

Harris, P. L. (1992). From simulation to folk psychology: The case for development. *Mind and Language*, *7*.

Kennedy, W. G., Bugajska, M. D., Harrison, A. M., & Trafton, J. G. (2009). "Like-me" simulation as an effective and cognitively plausible basis for social robotics. *International Journal of Social Robotics*, *1*(2), 181-194.

Laird, J. E. (2001). It knows what you're going to do: Adding anticipation to a quakebot. In *Proceedings of the International Conference on Automomous Agents.*

Leslie, A. M., Friedman, O., & German, T. P. (2004). Core mechanisms in 'theory of mind'. *Trends in Cognitive Sciences*, *8*(12), 528-533.

Leslie, A. M., German, T. P., & Polizzi, P. (2005). Belief-desire reasoning as a process of selection. *Cognitive Psychology*, *50*, 45-85.

Onishi, K. E., & Baillargeon, R. (2005). Do 15-month-old infants understand false beliefs? *Science*, *308*.

Wellman, H. W., Cross, D., & Watson, J. (2001). Meta-analysis of theory-of-mind development: The truth about false belief. *Child Development*, *72*(3), 655-684.

Task-Constrained Interleaving of Perceptual and Motor Processes in a Time-Critical Dual Task as Revealed Through Eye Tracking

Anthony J. Hornof (hornof@cs.uoregon.edu)
Yunfeng Zhang (zywind@cs.uoregon.edu)
Computer and Information Science, University of Oregon
Eugene, OR 97403 USA

Abstract

A multimodal dual task experiment that contributed to the original development and tuning of the EPIC cognitive architecture is revised and revisited with the collection of new high fidelity human performance data, most notably detailed eye movement data, that reveal the complex overlapping of perceptual and motor processes within and between the two competing tasks. The data permit a new detailed evaluation of assumptions made in previous models of the task, and contribute to the development of new models that explore opportunities for overlapping visual-perceptual, auditory-perceptual, ocular-motor, and manual-motor activities. Three models are presented: (a) A hierarchical task-switching model in which each task locks out the other; the model explains reaction time but does not account for eye movement data. (b) A maximum-perceptual-overlap model that maximizes parallel processing and predicts the trends in the eye movement data, but performs too quickly. (c) A moderately-overlapped model that introduces task-motivated constraints and predicts both reaction time and eye movement data. The best-fitting model demonstrates the complex task-constrained interleaving of perceptual and motor processes in a time-pressured dual task.

Keywords: Cognitive strategies, EPIC cognitive architecture, eye tracking, multimodal dual task, multitasking.

Introduction

A critical task domain for the research enterprise of cognitive modeling is that of multimodal (auditory and visual) multitasking. Psychologists and cognitive modelers puzzle over the question of how people engage in two or more time-pressured tasks that compete for perceptual, cognitive, and motor processes, such as for air-traffic control or in-car navigation (Byrne & Anderson, 2001; Howes, Lewis, & Vera, 2009; Meyer & Kieras, 1997; Salvucci & Taatgen, 2008). Gaining an understanding and ability to predict aspects of multimodal multitasking is of critical scientific and practical importance. This paper advances an understanding of such tasks by presenting cognitive models of time-critical multimodal multitasking and evaluates these models in detail using eye tracking data.

The Time-Critical Multimodal Dual Task

An earlier version of the experiment that forms the basis of this theoretical exploration was conducted in the early 1990s at the Naval Research Laboratory (NRL) (Ballas, Heitmeyer, & Perez, 1992). The experiment produced human speed and accuracy data that proved useful for developing detailed computational cognitive models of dual task performance (Kieras, Ballas, & Meyer, 2001). In the NRL dual task, participants use a joystick to track a moving target on one display and, in parallel, key-in responses to objects that appear on a secondary "radar" display. This paper presents an experiment that extends the original NRL dual task in numerous important ways, including that (a) eye movements are recorded, (b) eye tracking is used in some conditions to hide objects on the not-currently-looked-at display, (c) auditory cues relate more directly to required responses, and (d) participants are rigorously trained, financially motivated, and given extensive feedback so that performance approaches that of an expert.

Figure 1 shows an overview of the two displays used in the multimodal dual task modeled in this paper. Two tasks (or subtasks) were performed in parallel: a tracking task and a tactical classification task. The tracking task consisted of keeping a small circle on a moving target using a joystick. When the circle was positioned as such, it turned green, and the participant was financially rewarded at a constant rate. The tactical classification task consisted of monitoring groups of icons or "blips" (fifty-seven in a nine-minute scenario) that moved down a radar display, and keying-in the blip number and "hostile" or "neutral" as soon as the blip changed from black to red, green, or yellow, indicating that it was "ready to classify". When a blip became ready to classify, a financial bonus was awarded though it diminished at a constant rate until the blip was keyed-in, or classified. Red blips were hostile; green were neutral; yellow blips were classified based on their shape, speed, and direction, following practiced rules.

Two important factors were manipulated in the experiment: (a) peripheral visibility on or off and

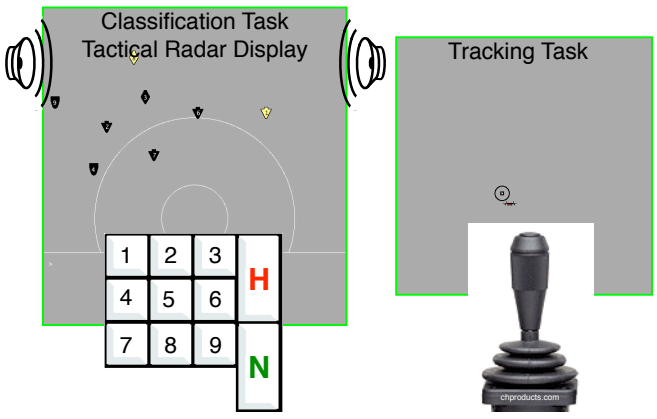


Figure 1: An overview of the visual and auditory displays and input devices used in the multimodal dual task.

(b) auditory cues present or absent. *Peripheral Visibility* manipulated whether participants could see the contents of the other display—radar or tracking—that they were not currently looking at. This simulates a task environment in which visual displays are separated by enough distance such that they cannot be monitored with peripheral vision. *Auditory Cues (Sound On)* indicates that a blip's initial appearance (as black) and color change (to red, green, or yellow) were indicated with spatialized auditory cues. Each nine-minute scenario maintained a constant setting of peripheral visible or not-visible and sound on or off.

Figure 2 summarizes the most important eye and hand movement data from the experiment, which is described in more detail in Hornof, Zhang, Halverson (2010). Figure 2 shows the time required for the four consecutive stages of classifying a blip: (a) Initiate the eye movement from the tracking display to the tactical display; (b) once on the tactical, find the target and move the eyes to it; (c) keep the eyes on the blip long enough to identify it and then move the eyes back to tracking; and (d) after the eyes are back on tracking, key-in the blip (keying-in was consistently performed *after* the eyes were back on tracking). These data serve to reveal the complex interleaving of perceptual, cognitive, and motor processing, and provide a basis for the current modeling endeavor.

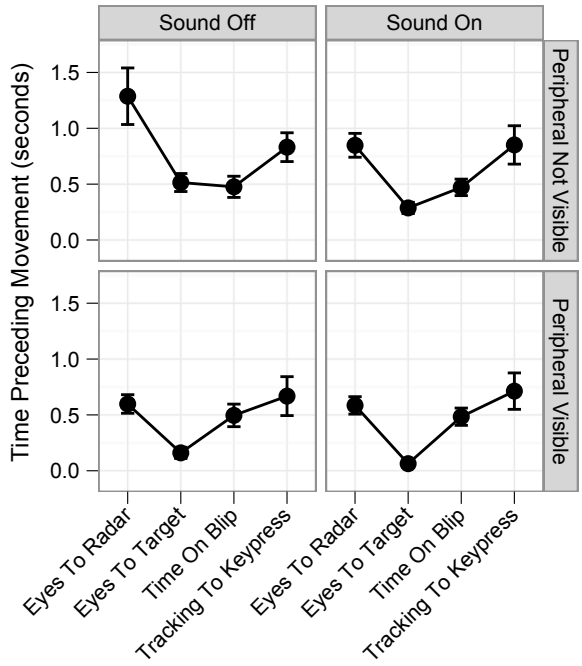


Figure 2. Time preceding eye movements across the lifetime of a colored blip. Each panel shows a unique combination of the factors of peripheral visibility and sound on/off. The *x*-axis shows a sort of timeline of the stages involved in classifying a blip.

The EPIC Cognitive Architecture

The EPIC cognitive architecture (Executive Process-Interactive Control; Kieras & Meyer, 1997) was used to model the multimodal dual task, as it was used previously to model the earlier version of the same task (ibid.; Kieras, Ballas, & Meyer, 2001). EPIC is particularly well-suited for exploring a range of explanations of multitasking performance because of its specific commitment, at the architectural level, to only enforcing sequential processing for motor activities, such as to constrain the eyes to rotate to only one point at a time, and the hands to only execute one sequence of movements at a time. Perceptual information can flow into the auditory and visual processors in parallel, and multiple production rules—IF-THEN statements that represent the strategy used to do a task—can fire in a single 50 ms cycle. Strategies can be written to permit only one rule to fire at a time (as in our initial model) or to explore the full potential of overlapping (as in our second model).

Extensions to the EPIC Cognitive Architecture

Initial sets of production rules that were constructed to put the eyes and hands through the tasks revealed two extensions to the EPIC cognitive architecture that would be needed to model this task: (a) a computational solution to the binding problem, which is the question of how people assemble perceptual stimuli to maintain a seamless conscious experience, and (b) a temporal processor to determine, entirely from within the simulated organism, when a certain amount of time has elapsed.

To address the binding problem, the visual processor in the EPIC cognitive architecture was updated (by EPIC's creator David Kieras) so that psychological objects in EPIC's visual working memory maintain their identity even as they disappear and reappear in the physical environment. In other words, if the simulated human moves its eyes so that a blip disappears (as in the peripheral-not-visible conditions), and then moves its eyes so that the same blip reappears, EPIC would previously have created a new psychological object for the reappeared blip. Now, provided that the initial psychological object associated with the blip did not fully decay, the reappeared blip is reconnected to the already-existing psychological object.

The second extension to EPIC was to add a temporal processor that replicates the temporal processor added to the ACT-R cognitive architecture (Taatgen, van Rijn, & Anderson, 2007). This gives the models a way to make self-motivated periodic checks of the tactical display when there was no peripheral visibility or auditory cuing.

Modeling Overview

Each of the models below were presented with the exact same auditory and visual stimuli in identical nine-minute scenarios that were presented to our human participants.

The following parameters were used in the models: The time required to determine the classification of a yellow blip based on its speed and direction was set to 800 ms. Alarm sounds are identified 300 ms after their onset in auditory perception rather than with their onset, to give enough time to distinguish the alarm from the blip appearance sound.

A common element within all strategies include that tracking adjustments (by moving the joystick with a Ply) were made only when the tracking circle was *not* green, consistent with a strategy that maximizes payoff.

The model development presented here follows the "bracketing" approach advocated by Kieras & Meyer (2000) in which the analyst attempts to "bracket" the human data with a slowest-reasonable and fastest-reasonable strategies. Three corresponding task strategies are developed: (a) Hierarchical task-switching (the slowest-reasonable model); (b) Maximum-perceptual-overlap (the fastest-reasonable model); and (c) Moderately-overlapped (the fastest-reasonable model slowed down based on task constraints). Models based on these three strategies, and comparisons of each model's predictions with the human data, are presented next.

Hierarchical Task-Switching Model

The hierarchical task-switching (the slowest-reasonable) model represents a straightforward translation of the multimodal dual task into a hierarchical task with strict serial processing of each subtask. Figure 3 shows the corresponding hierarchical task analysis. The production rules were generated by first creating a GOMS model (John & Kieras, 1996) of the task, and then translating that model into the corresponding production rules. Parallelism existed in the model primarily in terms of auditory and visual information getting deposited in EPIC's perceptual stores.

A key characteristic of the model includes that, once it determines that a blip is ready to classify, it holds the eyes on that blip until the keystrokes for that blip are initiated. During this period, the cognitive processor is dedicated to just classifying the blip. Tracking is completely locked out. This aspect of the model resembles the original EPIC models of the task, in which "the dual-task executive enforces mutual exclusion between the tracking task and the tactical task." (Kieras, Ballas, & Meyer, 2001, p.10)

Figure 4 shows the mean blip classification times across the four combinations of peripheral-visibility and sound-on-or-off, and for red/green versus yellow blips. The model explains the overall reaction time data very well across all eight conditions, with an average absolute error (AAE) of 4.6%. (Note that all AAEs presented in this paper are calculated using the overall observed mean as the denominator for each percentage calculation, to reduce the distortion that would otherwise result from observed and predicted values that are very close to zero.)

If an analyst were primarily interested in the classification task and hence did not proceed to model the tracking task with any degree of fidelity, and if the analyst did not have any eye movement data to work with, the modeling project would likely be done at this point, and we might declare victory—we modeled the primary data of interest with good

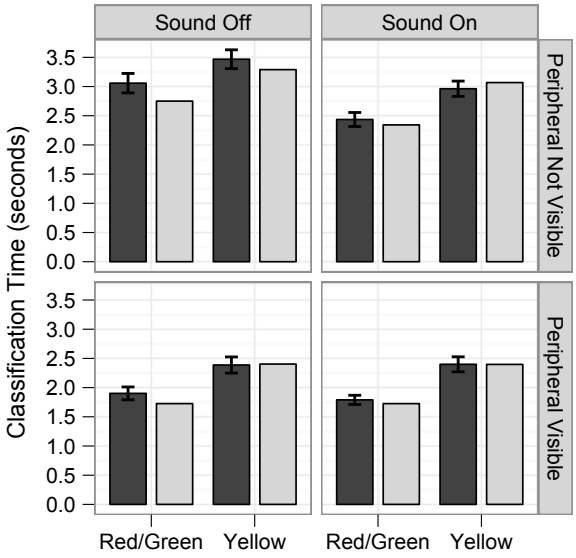


Figure 4: The mean classification time of blips as a function of blip color, observed (dark bars) and predicted (light bars) by the hierarchical task-switching model. The average absolute error (AAE) of the prediction is 4.6%.

accuracy. But a deeper look at the data that are available in this modeling exercise reveal a dark truth—the model is not accounting for the complex overlapping of visual and motor processes that participants are exhibiting with their eye movements. As well, a look at the tracking task data show that the model is performing far worse than skilled participants, predicting an overall mean tracking error of 42 pixels compared to the observed tracking error of 29 pixels.

Figure 5 shows the same observed data presented in Figure 2, along with the eye movement times predicted by the hierarchical task-switching model. As can be seen in Figure 5, the model is spending far too long looking at each blip. The tracking-to-keypress is negative (and hence a value of zero is used) because the model returns the eyes to tracking *after* the classification. Participants spent far less time on each blip, and spent substantial time with the eyes back on tracking before keying-in a classification.

The hierarchical task-switching model, though intended as a slowest-reasonable bracket, does a good job of predicting the mean classification times. But the model does not capture the interleaving of perceptual and motor processes that people clearly exhibited. The next model attempts to capture and maximize such an interleaving.

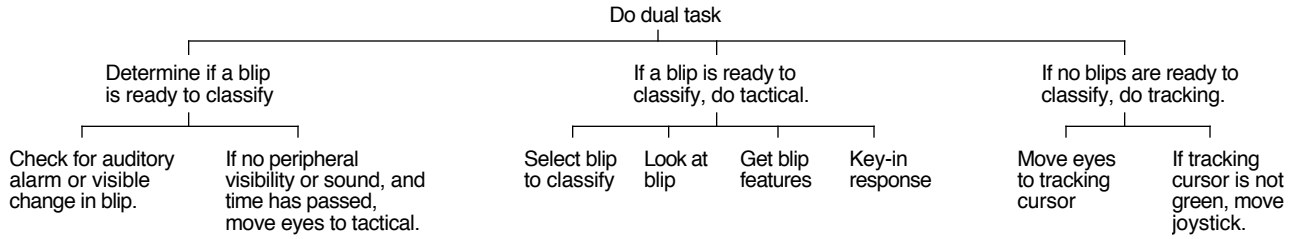


Figure 3: The hierarchical task analysis used to generate the hierarchical task-switching model.

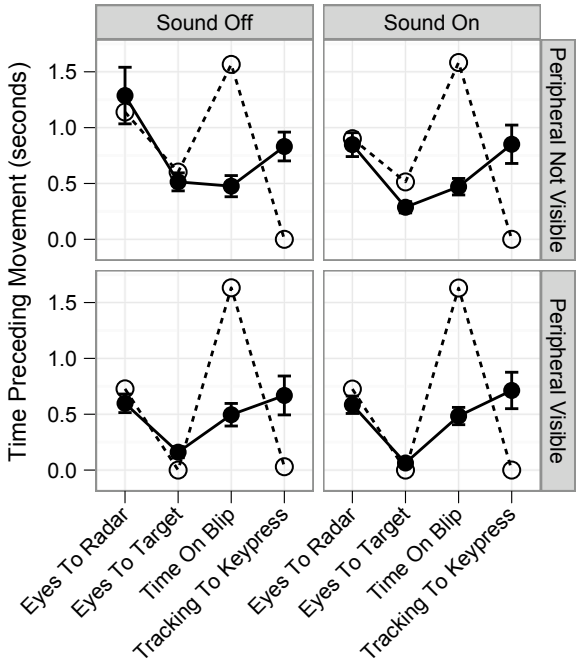


Figure 5: The time preceding eye movements observed (solid lines) and predicted (dashed lines) by the hierarchical task-switching model. (AAE = 91.4%)

Maximum-Perceptual-Overlap Model

The maximum-perceptual-overlap (fastest-reasonable) model is written to maximize all aspects of parallel processing that are built into the EPIC cognitive architecture. The production rules are written such that ocular-motor and manual-motor processing proceed entirely independently of each other, with manual-motor processing resulting from visual-perceptual features that become available based on ocular-motor activity.

Figure 6 shows two state transition diagrams that represent how one set of production rules moves the eyes between tracking and tactical to acquire visual information, and another set of rules independently shifts manual motor activity between tracking and tactical. When the model runs, both sets of rules—ocular-motor and manual-motor—spend most of their time on tracking. When a blip appears, the ocular-motor rules shift to tactical just long enough to perceive blip features, which become available to the manual-motor rules, which switch briefly to tactical to key-in a response. Each set of rules returns to tracking as soon as its tactical subtask is completed.

Figure 7 shows the classification time predictions of the maximum-perceptual-overlap model. As can be seen, the model is too fast, as would be expected for a fastest-reasonable model. Looking at the predicted eye movement times in Figure 8, however, reveals that the model does a good job predicting the overall trends in how long the eyes took to move through the stages involved in classifying a blip, especially in the peripheral-visible conditions. The comparably good fit of the eye movement data, especially when compared to the first model's poor fit with the same data, suggest that participants may truly have developed

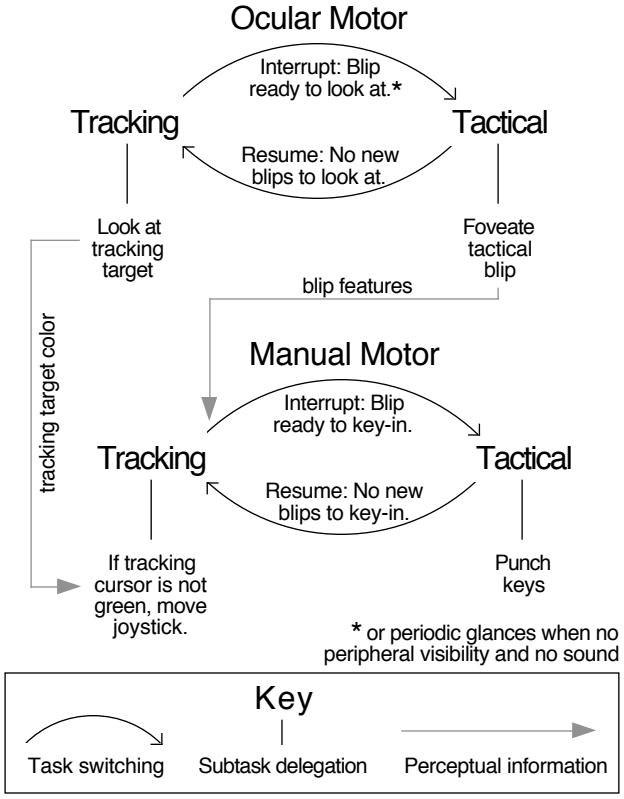


Figure 6: State transition diagrams that represent the independent ocular-motor and manual-motor processing in the maximum-perceptual-overlap model.

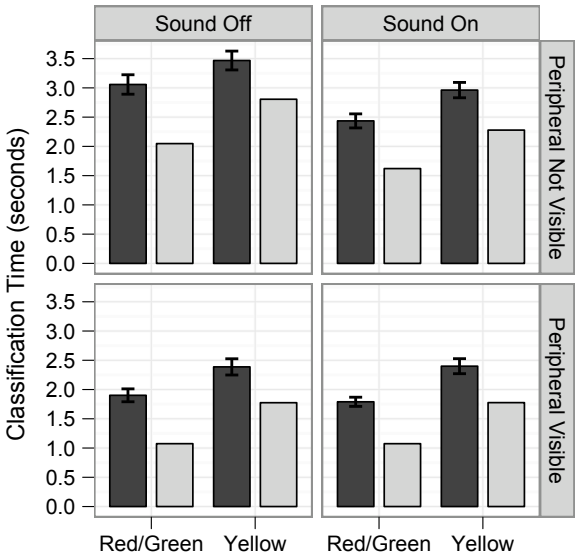


Figure 7: Classification times observed (dark bars) and predicted (light bars) by the maximum-perceptual-overlap model. (AAE = 29.2%)

expert strategies that include independent parallelism between ocular-motor and manual-motor decision making. But, as might be expected, the fastest-reasonable model is overall too fast. The predicted mean tracking error is also substantially better (20 pixels) than the observed (29 pixels).

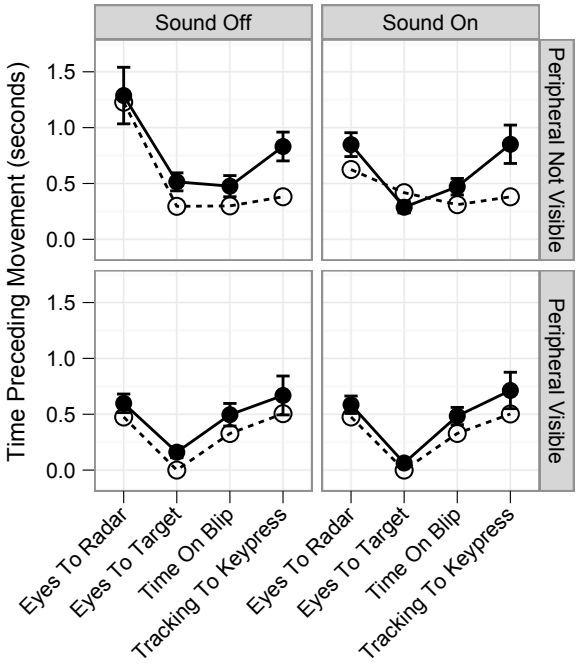


Figure 8: The time preceding eye movements observed (solid lines) and predicted (dashed lines) by the maximum-perceptual-overlap model. (AAE = 32.6%)

The final strategy explores constraints that can be introduced to the fastest-reasonable model.

Moderately-Overlapped Model

The moderately-overlapped model was constructed by starting with the maximum-perceptual-overlap (fastest-reasonable) model, presented in the previous section. Three analyses were conducted. First, the model traces and observed data were studied side-by-side to reveal subtle differences between the predicted and observed eye and hand movements. Second, opportunities were explored to adjust strategies to maximize payout (see Howes et al., 2009). Third, the manual-motor devices were examined to improve the fidelity of their simulation.

These analyses led to the following five adjustments to the model, all of which are represented by the bold italic additions in Figure 9: (a) Eye-to-radar time is delayed by having the tracking task finish any joystick Ply underway, waiting for the tracking circle to turn green, to leave that task in a money-making mode. (b) The time on yellow blips is extended to permit identification of speed and direction (set to 250 ms). (c) Tracking-to-keypress time is extended by assuming that, when moving the eyes from tactical back to tracking, people make one joystick adjustment before keying-in the blip classification; this increases tracking payment while further considering the classification. (d) The timing for a Ply was increased (to a coefficient of 300 and minimum time of 400 ms) assuming that the Ply effectively requires separate joystick movements to start and then stop the tracking circle. (e) The Punch was replaced with a Keypress to represent how the fingers are not positioned directly above the keys, but need to travel.

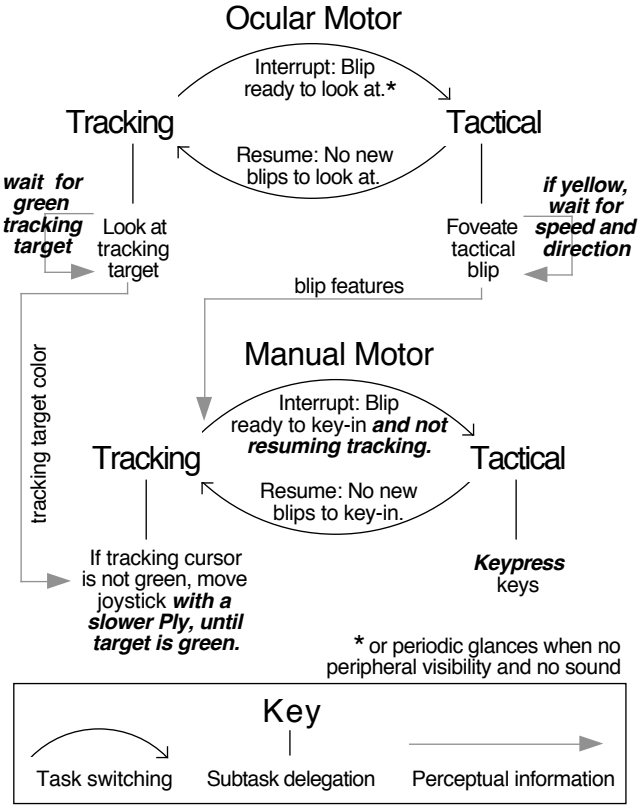


Figure 9: The moderately-overlapped model, with additions to the previous model shown in bold italics.

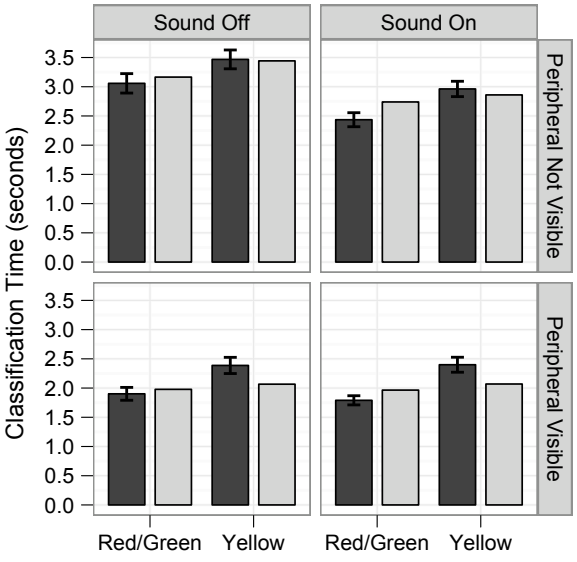


Figure 10: Times observed (dark bars) and predicted (light bars) by the moderately-overlapped model. (AAE = 7.1%)

Figures 10 and 11 show how the moderately-overlapped model does a good job of predicting both classification and eye-movement timings. The model also accurately predicts tracking error, predicting 26 pixels compared to the observed 29 pixels. Table 1 shows how this model provides the best overall fit with the observed data.

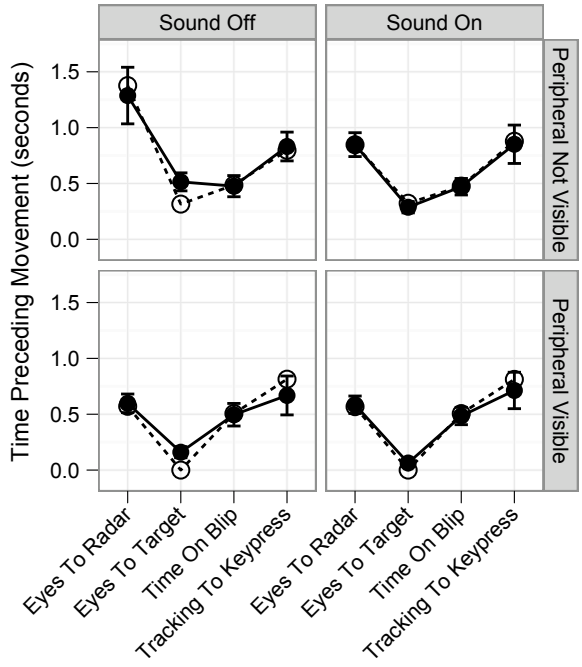


Figure 11: The time preceding movements observed (solid lines) and predicted (dashed lines) by the moderately-overlapped model. (AAE = 10.1%)

Table 1. Average absolute error of each model's predictions.

Model	Classification Time	Time Preceding Movements	Tracking Error
Hierarchical Task-Switching	4.6%	91.4%	43.6%
Maximum-Overlap	29.2%	32.6%	31.2%
Moderately-Overlapped	7.1%	10.1%	13.9%

Conclusion

The models presented here demonstrate the difficulty in accurately modeling complex multitasking behavior. First, there is the challenge of collecting enough data to evaluate the accuracy of a model; the initial hierarchical task-switching model accurately predicted the classification time, but not eye movements. Then, there is the challenge of correctly identifying opportunities for expert, overlapped behavior; the maximum-perceptual-overlap model presented here relied on the massive parallelism of the EPIC architecture's cognitive processor to demonstrate that expert strategies might manage ocular-motor and manual-motor processes largely independently. Lastly, there is the challenge of determining which task-based constraints should be introduced to govern the use of perceptual information that passes within and between two tasks that compete for motor processing; those presented for the moderately-overlapped model may or may not accurately capture the true constraints that governed behavior.

The models presented here do not clearly subscribe to the notion of an independent process that actively coordinates between two task strategies, whether that process be an executive process, as in the original models for a similar task (Kieras, Ballas, & Meyer, 2001) or an independent mechanism, as in Salvucci and Taatgen (2008). This paper explores the possibility that a dual task strategy is perhaps an altogether new, carefully interleaved strategy.

Acknowledgments

This research was funded in part by the Office of Naval Research under Grant No. N00014-06-1-0054. Tim Halverson assisted in creating the EPIC temporal processor.

References

Ballas, J. A., Heitmeyer, C. L., & Perez, M. A. (1992). Evaluating two aspects of direct manipulation in advanced cockpits. *Proceedings of ACM CHI '92: Conference on Human Factors in Computing Systems,* 127-134.

Byrne, M. D., & Anderson, J. R. (2001). Serial modules in parallel: The psychological refractory period and perfect time-sharing. *Psychological Review, 108,* 847-869.

Hornof, A. J., Zhang, Y., Halverson, T. (2010). Knowing where and when to look in a time-critical multimodal dual task. *Proceedings of ACM CHI 2010: Conference on Human Factors in Computing Systems,* New York: ACM, 2103-2112.

Howes, A., Lewis, R. L., & Vera, A. (2009). Rational adaptation under task and processing constraints: Implications for testing theories of cognition and action. *Psychological Review, 116*(4), 717-751.

John, B. E., & Kieras, D. E. (1996). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction, 3*(4), 287-319.

Kieras, D. E., Ballas, J., & Meyer, D. E. (2001). *Computational Models for the Effects of Localized Sound Cuing in a Complex Dual Task (No. EPIC Report No. 13).* Ann Arbor, Michigan: University of Michigan, Department of Electrical Engineering and Computer Science.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*(4), 391-438.

Kieras, D. E., & Meyer, D. E. (2000). The role of cognitive task analysis in the application of predictive models of human performance. In J. M. C. Schraagen, S. E. Chipman & V. L. Shalin (Eds.), *Cognitive Task Analysis* (pp. 237-260). Mahwah, NJ: Lawrence Erlbaum.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review, 104*(1), 3-65.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review 115,* 101-130.

Taatgen, N. A., van Rijn, H., & Anderson, J. R. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention, and learning. *Psychological Review, 114*(3), 577-598.

A Cognitively Bounded Rational Analysis Model of Dual-Task Performance Trade-Offs

Christian P. Janssen, Duncan P. Brumby, John Dowell, and Nick Chater
c.janssen@ucl.ac.uk, brumby@cs.ucl.ac.uk, j.dowell@cs.ucl.ac.uk, n.chater@ucl.ac.uk
University College London, Gower Street, London WC1E 6BT, UK

Abstract

The process of interleaving two tasks can be described as making trade-offs between performance on each of the tasks. This can be captured in performance operating characteristic curves. However, these curves do not describe what, given the specific task circumstances, the optimal strategy is. In this paper we describe the results of a dual-task study in which participants performed a tracking and typing task under various experimental conditions. An objective payoff function was used to describe how participants should trade-off performance between the tasks. Results show that participants' dual-task interleaving strategy was sensitive to changes in the difficulty of the tracking task, and resulted in differences in overall task performance. To explain the observed behavior, a cognitively bounded rational analysis model was developed to understand participants' strategy selection. This analysis evaluated a variety of dual-task interleaving strategies against the same payoff function that participants were exposed to. The model demonstrated that in three out of four conditions human performance was optimal; that is, participants adopted dual-task strategies that maximized the payoff that was achieved.

Keywords: multitasking; performance operating characteristic; cognitively bounded rational analysis

Introduction

Multitasking behavior often involves trade-offs in performance (e.g., time, errors, extension, etc.) between the tasks. Such trade-offs can be described graphically with Performance Operating Characteristics, which show how the performance of separate tasks vary together systematically (Navon & Gopher, 1979; Norman & Bobrow, 1975). Trade-offs reflect strategic choices and can be modified, for example, in response to instructions to prioritize one task over another (e.g., Brumby, Salvucci, & Howes, 2009; Janssen & Brumby, in press).

Consideration of the strategic choices made in multitasking (i.e., of *why a specific* way of performing the tasks is chosen) naturally supposes some optimal trade-off. Why time is allocated differentially to the tasks, and why particular patterns of interleaving are adopted, must reference the relative success of those different strategies. In this paper we use an objective payoff function to integrate into a single score the performance rewards in a tracking-while-typing dual-task situation. Such payoff functions have been used before in multitask studies, but only to show that performance is sensitive to isolated factors such as changes in reward structure (e.g.,Wang, Proctor, & Pick, 2007). Objective payoff functions have not previously been used to support explanations of multitasking strategy choices, or to assess the optimality of strategies.

Combined with a cognitive model that can perform alternative multitasking strategies (i.e., alternatives for when to interleave and execute multiple tasks), a payoff function enables an evaluation of the success of each of the strategies (Howes, Lewis, & Vera, 2009). Strategies with the highest payoff can be determined and compared with human performance in experimental settings. This can be used to explain the strategic choices participants make.

We developed a tracking-while-typing dual-task to test the hypothesis that people can hone their dual-task behavior to maximize the payoff that is achieved. The task required participants to keep a randomly moving cursor inside a circular area and to type a string of digits. Tracking tasks have been used in several multitasking studies (e.g., Gopher, 1993; Hornof, Zhang, & Halverson, 2010; Kieras, Meyer, Ballas, & Lauber, 2000; Lallement & John, 1998; Salvucci & Taatgen, 2008). For example, Gopher (1993) showed that performance trade-offs in a tracking-while-typing task can be influenced by instructions to spend more time on one of the tasks. Within the cognitive modeling literature, the work by Lallement and John (1998) is interesting as it compares performance of models developed in several cognitive architectures on a tracking and choice task. We attempt to extend this work by showing how a payoff function enables us to bind normative cognitive models with experimental observations of multitasking behavior, and specifically, to show how multitasking strategy choice can be better explained by seeing it in relation to optimal performance.

Experiment

Method

Participants Eight participants (4 female) between 20 and 35 years of age (M = 23 years) from the subject pool at UCL participated for monetary compensation. Payment was based on performance (details are provided in the Materials section). The total payment achieved by participants ranged between £7.13 and £11.45 (M = £9.14).

Materials The dual-task setup required participants to perform a continuous tracking task and a discrete typing task, presented on a single monitor. Figure 1 shows the layout of the tasks on the display. The typing task was presented on the left side and the tracking task on the right. Each task was presented within a 450 x 450 pixels area, with a vertical separation of 127 pixels between the tasks.

The tracking task required participants to keep a square cursor that drifted about the display in a random fashion inside a target circle (see Figure 1). The cursor was 10 x 10 pixels and the target had a radius of either 80 (small target) or 120 pixels (large target). A random walk function was used to vary the position of the cursor in the display every 20 milliseconds. The rate at which the cursor drifted about the display was varied between different experimental conditions. In a low noise condition the random walk had a mean of zero and standard deviation of 3 pixels per update, while in a high noise condition the random walk had a mean of zero and standard deviation of 5 pixels per update.

Participants used a Logitech Extreme 3D Pro joystick with their right-hand to control the position of the cursor in the tracking display. The drift function of the cursor was suspended whenever the joystick angle was greater than 0.08 (the maximum angle was 1). The speed was scaled by the angle, with a maximum of 5 pixels per 20 milliseconds.

The typing task required participants to enter a string of twenty digits using a numeric keypad with their left-hand. The string was made up of the digits 1 to 3, where each digit occurred at least six times in a given sequence. Digits were presented in a random order with the constraint that no single digit was presented more than three times in a row in the sequence (e.g., "11233322132123132123" as in Figure 1). When a digit was entered correctly it was removed from the to-be-entered sequence. In this way, the left-most digit on the display was always the next one to be entered.

The study used a forced interleaving paradigm, in which only one of the two tasks was visible and could be worked on at any moment. By default the typing task was visible and the tracking task was covered by a gray square. In order to see and control the tracking task, participants had to hold down the trigger of the joystick. When the trigger was released, the tracking task was covered by a gray square and the typing task revealed.

Design The study manipulated aspects of the tracking task using a 2 (cursor noise: low vs. high) x 2 (target size: small vs. large) within-subjects design. The main dependent variables were the time required to complete the typing task and maximum distance of the cursor from the center of the target circle.

Participants were remunerated based on performance, as determined by an objective payoff function that was calculated for each dual-task trial. The function was designed to encourage fast completion of the typing task, while keeping the cursor inside the target. The payoff (in pounds) for a given trial was defined as:

$$\textbf{Payoff} = \textbf{Gain} + \textbf{Digit Penalty} + \textbf{Tracking Penalty}$$

The minimum payoff for a given trial was limited to £-0.20. The gain component was based on the total time required to complete a dual-task trial (in seconds):

$$\textbf{Gain} = 0.15 * e^{-1*\textbf{TotalTrialTimeInSec}/20 + 0.25}$$

This function was determined using pilot studies, to make sure participants mostly gained money. To encourage accurate typing, a digit penalty deduced £0.01 from the total payoff whenever an incorrect digit was entered. To encourage participants to keep the cursor inside the target circle of the tracking task, a tracking penalty was applied:

$$\textbf{Tracking Penalty} = -0.1*e^{\textbf{SecOutside}/1.386 - 0.6931}$$

This penalty was crafted such that £0.10 was lost when the cursor was outside of the radius for 0.5s, and £0.20 was lost when it was outside of the radius for 1s. In the remainder of this paper we will not look at the effect of digit penalty on payoff.

Procedure Participants were informed that they would be required to perform a series of dual-task trials and that they would be paid based on their performance. A participant's payment was based on the cumulative payoff over the course of the study, in addition to their base payment of £3. Participants were told that they would gain more points by completing the typing task as quickly as possible, but that they would lose points if they made a typing error or if the cursor drifted outside of the target area in the tracking task. We chose not to give participants a formal description of the payoff function, but instead provided explicit feedback after every dual-task trial with the payoff score achieved.

After explaining how to perform each of the tasks participants performed two single-task training trials for each task and two dual-task practice trials. Participants were instructed that for dual-task trials only one of the two tasks would be visible and controllable at any moment in time, and they were instructed how to switch between tasks.

Participants then completed four blocks of experimental trials (one for each experimental condition). The order of conditions was randomized and counter-balanced across participants, with the exception that blocks of the same noise level were grouped together. The order of radius sizes was repeated across the first two blocks and the second two blocks. For each block, participants completed five single-task tracking trials, five single-task typing trials, and twenty dual-task trials. The dual-task trials were further grouped into sets of five trials, with a short pause between each set. The total procedure took about one hour to complete.

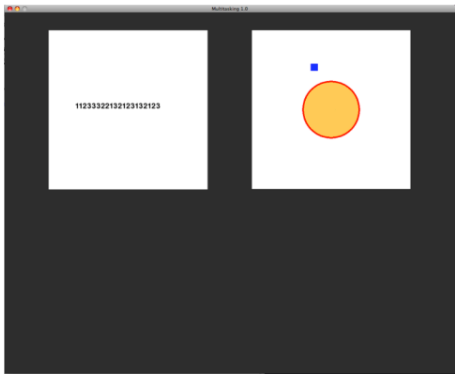


Figure 1: Position of the two tasks in the interface

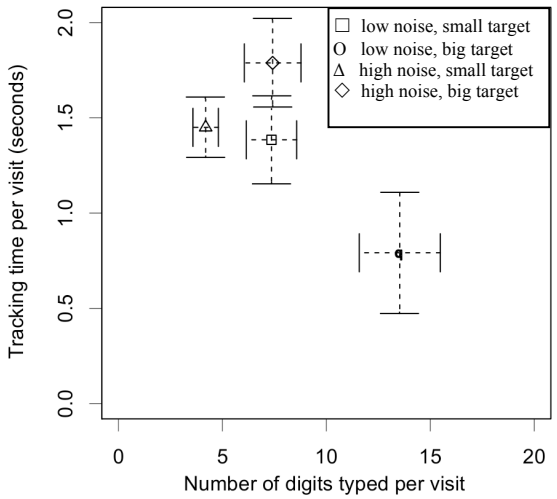


Figure 2: Number of digits typed and tracking time, both per visit. Error bars depict standard errors.

Results

We focus on performance during the last five dual-task trials of each experimental condition, as these reflect a period during which the participant has had time to adapt behavior to the objective payoff function. A 2 (cursor noise) x 2 (target size) analysis of variance (ANOVA) was used for all statistical analysis with a significance level of .05.

Overall performance We first consider the effect of varying aspects of the tracking task on the time required to complete the typing task and the maximum distance of the cursor from the center of the target circle in the tracking task. It was found that trial time was significantly longer when there was greater noise in the tracking task (M = 11.17s, SD = 4.32s) than when there was a lower level of noise in the tracking task (M = 7.51s, SD = 2.00s), $F(1, 7)$ = 15.07, $p < .01$. Trials were also longer when the target in the tracking task was smaller (M = 10.59s, SD = 4.01s) than when it was larger (M = 8.09s, SD = 3.22s), $F(1, 7)$ =11.84, p = .01. There was no significant interaction, $F(1, 7)$ = 0.22.

In the tracking task we consider the maximum distance of the cursor from the center of the target over the course of a trial. It was found that the cursor drifted more when there was a higher level of noise (M = 95 pixels, SD = 15 pixels) than when there was a lower level of noise (M = 61 pixels, SD = 8 pixels), $F(1,7)$=33.42, $p < .001$. There was no effect of target size on the maximum distance that the cursor drifted over a trial ($F(1,7)$ = 1.19, p = .31), nor was the interaction effect significant ($F(1,7)$ = 0.07).

These differences in overall task performance between conditions are somewhat expected and unsurprising because they partly reflect differences in the difficulty of the tracking task. We were far more interested in how this performance was achieved. We next consider the dual-task interleaving strategy that was adopted in each condition.

Strategies Two aspects determine a strategy: (1) the number of digits typed during each visit to the typing window and (2) the amount of time spent in the tracking window per visit to this window. Figure 2 shows these two basic strategy dimensions for each of the four conditions. It can be seen that for each experimental condition there is a unique point in this strategy space – strategies differ between conditions. The number of digits entered per visit increased with an increase in target size ($F(1, 7) = 17.4$, $p < .01$), and it also increased with a decrease in cursor noise (that is, more digits were typed when it took longer for the cursor to cross the boundary; $F(1, 7) = 15.18$, $p < .01$). There was no significant interaction ($F(1, 7) = 3.24$, $p = .12$).

It can also be seen in Figure 2 that the time spent in the tracker window per visit increased with an increase in the noise associated with the cursors movement ($F(1,7)$=14.98, p = .01). An interaction effect was present as visit time was particularly short in the low noise, large radius condition ($F(1,7)$=11.55, p = .01). There was no significant effect of radius ($F(1,7)$=0.54).

A CBRA Model of Tracking-while-Typing

The results show that participants adapted their dual-task behavior to changes in the difficulty of the tracking task. However, what these results do not show is whether participants were adopting a strategy that is *optimal* in terms of maximizing the expected payoff that could be achieved in each condition. To answer this question we developed a cognitively bounded rational analysis model (Howes, et al., 2009). This framework is particularly useful for comparing the performance of alternative strategies, allowing strategies to be discriminated based on the payoff achieved. The model developed here is inspired by our previous work in developing models of a dialing-while-driving dual-task set-up (e.g., Brumby, Salvucci, & Howes, 2007; Brumby, et al., 2009; Janssen & Brumby, in press). Both dual-task environments share core characteristics, but the current work differs in that it incorporates an explicit payoff function against which various dual-task interleaving strategies can be evaluated. In the next section, we use a model to determine whether people were selecting strategies that would maximize the financial payout that could be achieved in each condition.

Model Development

Tracking Model The crucial question for developing a model of the tracking task was at what angle participants held the joystick given their current distance from the center of the target. Figure 3 shows the mean values for discrete bins of 5 pixels for the horizontal axes (vertical data is similar). We fitted a linear function (shown as a dotted line):

Angle = -0.01 * current distance from target

The joystick had a maximum angle of (-)1. As in the experiment, the speed of the cursor is calculated by multiplying the angle of the joystick with a value of 5 pixels. Speed is calculated once every 250 milliseconds of tracking, and the cursor position is updated every 20 milliseconds based on this speed value. As in the

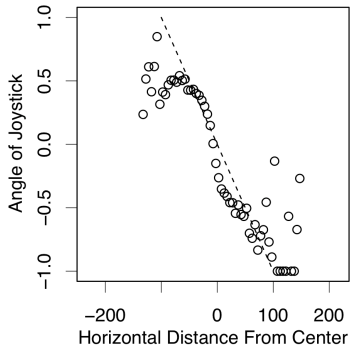


Figure 3: Angle of the joystick as a function of distance from the target. The dashed line shows a fitted function.

experiment, the cursor could only be controlled when the tracking window was open. The total time spent tracking in dual-task was varied as part of the strategy (see below).

Typing Model To model the typing task we fitted model performance to human single-task typing performance data. The time taken to type a digit (260 milliseconds) is identical to the mean inter-keypress interval measured in single-task.

Dual-Task Model The dual-task model works as followed. The model starts of with typing a series of digits (the length of which is varied as a strategy). For switching between typing and tracking a switch cost of 250 milliseconds is incurred, based on experimental data (time between last key press and pressing the trigger on the joystick: 247 milliseconds). The model then tracks the cursor for a designated amount of time (varied between runs as a strategy aspect). When it switches back to typing, a switch cost of 180 milliseconds is incurred (time between releasing the trigger and pressing the first key press minus single task inter-keypress interval: 185 milliseconds). Noteworthy, switch cost values are close to those in ACT-R models (e.g., Borst, Taatgen, & Van Rijn, 2010) and in the Cognitively Bounded Rational Analysis driving models.

Strategies We used this basic model to explore performance of a variety of strategies. A strategy is determined by the number of digits that are typed in sequence during a visit to the target window. We consider only a subset of twenty simple strategies that placed a consistent number of digits during each visit (between 1 and 20), with the exception of the last visit during which the remaining digits were placed (e.g., strategy 6-track-6-track-6-track-2, but not 6-track-4-track-6-track-4). In addition, for each visit to the tracking task, more or less time can be spent on tracking. We systematically explored performance for models that spent between 250 to 3000 milliseconds on tracking during each visit to the tracking window, using steps of 250 milliseconds (i.e., 12 alternatives). This gave a total of 229 (20 x 12 – 11) strategy alternatives.

The objective function for rating performance is similar as in the experiment with the exception that the model does not make typing errors. For each strategy alternative 100 runs were performed. Mean performance is reported.

Model Results

The first question of interest was whether the model would fit the experimental data. In particular, if we hardcode a strategy that types the same number of digits per visit and spends about the same amount of time tracking as participants did in each condition (with both measures lying within two standard errors of human means), does this then

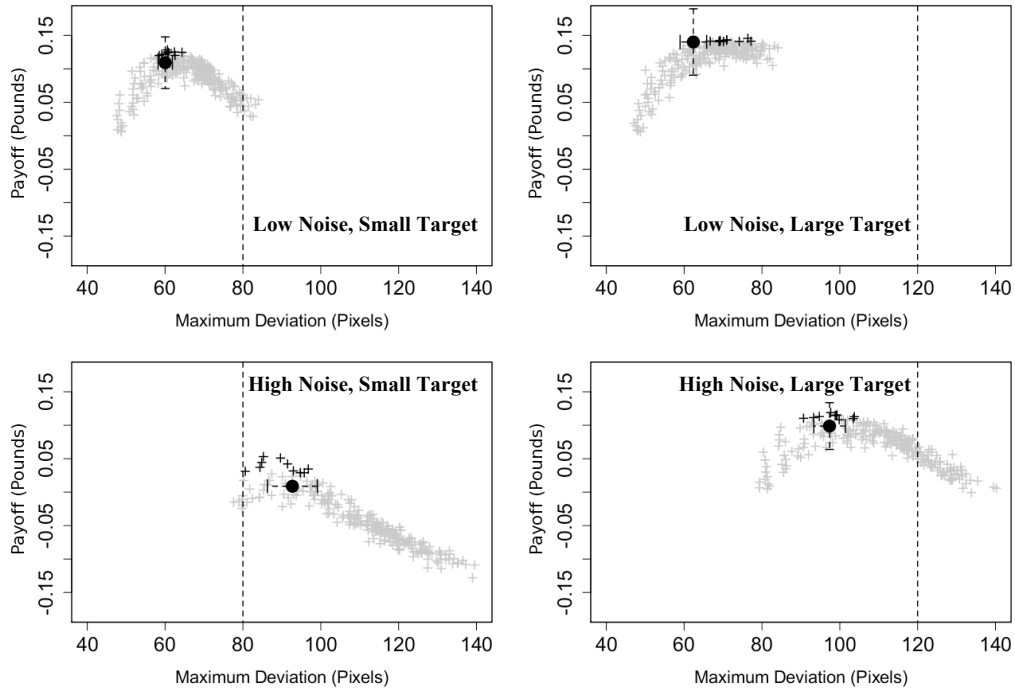


Figure 4: Maximum deviation versus predicted payoff per trial for the ten best (black crosses), and other strategies (gray crosses) per condition. Human results are shown as circles with standard error. The dotted line shows the target boundary.

result in similar total trial time and maximum deviation in each experimental condition (again with performance within two standard errors of the mean)? This is important so as to know that we have a reasonable calibration of the model's performance relative to the human data. This was the case.

Given that we can be confident that the model is reasonably calibrated to the human data on the observed strategy, we can now use the model to evaluate the payoff achieved by different (unobserved) dual-task interleaving strategies. Figure 4 shows a plot of the average maximum deviation versus payoff. We plotted the ten highest scoring strategies with black crosses, and the other strategies with gray crosses. In each condition there is a strong peak, though the shape of the distribution of scores differs between experimental conditions. In three out of four conditions the human data (black circles) lies in the region of maximum deviations that can achieve the highest performance. In each figure a vertical line shows the boundary of the target. Note that in the low noise, large radius condition participants could have let the cursor drift more to improve their score slightly (they would never cross the target boundary). Due to space limitations, we omitted a plot of total time data versus score; the pattern is similar.

Traditionally, differences in dual-task performance are plotted in Performance Operating Characteristics (POCs), in which performance on one measure or task is shown against performance on the other measure or task (Navon & Gopher, 1979; Norman & Bobrow, 1975). In Figure 5 we show the POC of total time and maximum deviation for the model and human data. The ten best performing strategy alternatives are again plotted with black crosses. There are a couple of interesting aspects to these graphs. First, the best performing models lie on the outer edge (left side, and bottom side) of the strategy space: the trade-off curve. That is, the best strategies make an optimal trade-off between performance on the two tasks. Furthermore, the position of the optimum strategies is at a *different section* (e.g., top left, or bottom right) of this curve for each condition. The model is essential for this assessment, as traditional POCs cannot predict optimal regions by themselves.

Human data again lies in the region of optimum payoff for three out of four conditions. Only in the low noise large target condition could participants have scored better by spending less time on the tracking task (increasing maximum deviation, but decreasing trial time). In all other conditions, the model illustrates that participants made good performance trade-offs to optimize their payoff.

General Discussion

In this paper we have presented an experiment and a model of a tracking-while-typing dual-task setup. A good feature of the task environment, in which participants need to track a cursor and type in digits, is that it translates performance on both tasks into a single performance score. Due to this feature, we were able to move beyond observations that participants trade-off performance in tasks, as done in classical dual-task research (Navon & Gopher, 1979; Norman & Bobrow, 1975) and in research on dual-task driving behavior (e.g., Janssen & Brumby, in press). Here, we were able to demonstrate that participants mostly made performance trade-offs in an optimal manner, so as to maximize pay-off (cf. Howes et al., 2009).

These claims are possible because of the use of a payoff

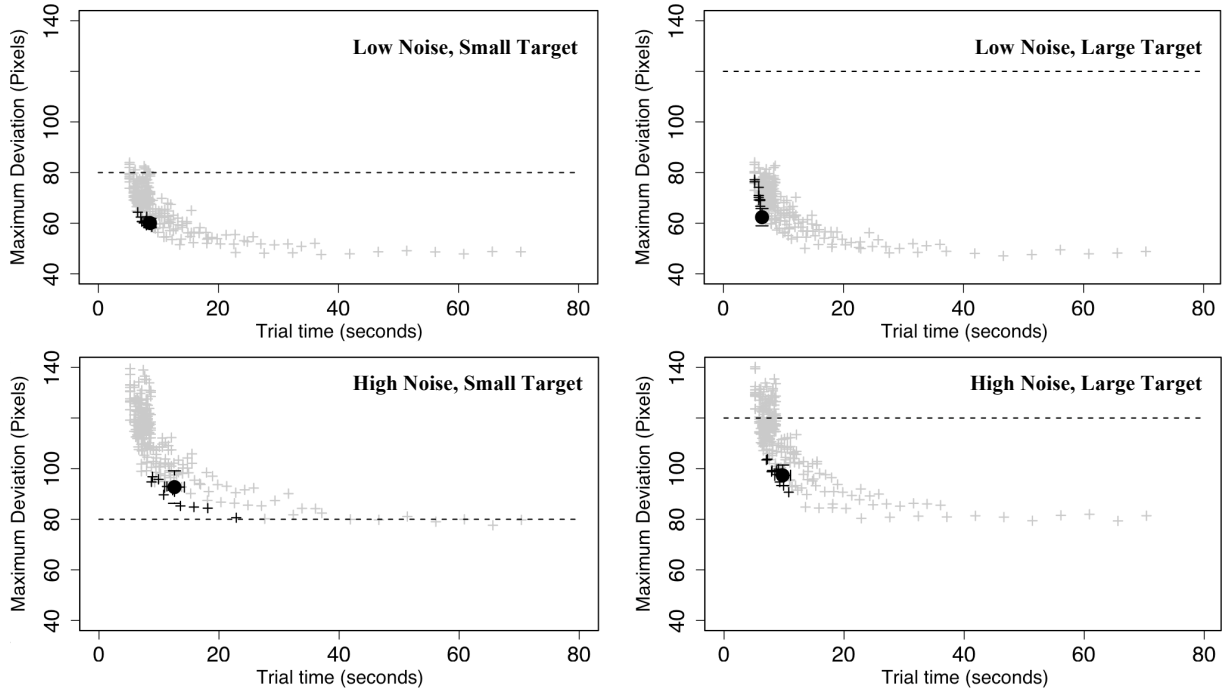


Figure 5: POCs of model performances for the ten best (black crosses), and other strategies (gray crosses) per condition. Human results are shown as circles with standard error. The dotted line shows the target boundary.

function that explicitly describes how participants ought to trade performance on each task to gain payment. The goal of this paper is not to argue that objective functions are the most prevalent aspect of performance in the real world. However, they make it possible to quantify how good performance is. This contrasts with previous work where verbal instructions on how to trade performance on each task is given (e.g., Gopher, 1993; Horrey, Wickens, & Consalus, 2006; Levy & Pashler, 2008), or where performance is sensitive to a change in payment (e.g., Wang, et al., 2007). In contrast, we can define *optimal* performance in terms of maximizing payoff.

There was however one condition (the low noise, large target condition) in which participants did not maximize the payoff that was achieved. In this condition, participants could have typed all of the digits in one sequence (i.e., without multitasking) to receive a slightly higher payoff than was actually observed. Two possible explanations for suboptimal performance are that participants overestimated the danger of the cursor crossing the boundary (which would give a severe penalty), or they were biased to switch between the two tasks (which is necessary in the other conditions). In this sense, participants not always adapt their behavior to maximize the payoff function. Further research is required to investigate such biases.

The model was developed with a minimal set of assumptions. This was already enough to demonstrate that people mostly adapt performance to an objective function. Further research can investigate how people adapt their behavior to different payoff functions, which, for instance, give greater weight to performance on one of the two tasks. Also, the model of the typing task might be refined to predict typing errors, and to predict the effect of the different times needed to type repeating digits versus non-repeating digits (cf. Janssen, Brumby, & Garnett, 2010). At a different level of analysis, the role of eye-movements can be considered to explore a wider variety of strategies (cf. Hornof, et al., 2010), such as strategies in which some visits to the typing task window are only spent on studying, and not typing digits.

Acknowledgments

This work was supported by EPSRC grant EP/G043507/1. We thank Julian Marewski and two anonymous reviewers for their valuable comments on this paper.

References

Borst, J. P., Taatgen, N. A., & Van Rijn, H. (2010). The problem state: A cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, memory, and cognition, 36*, 363-382.

Brumby, D. P., Salvucci, D. D., & Howes, A. (2007). Dialing while driving? A bounded rational analysis of concurrent multi-task behavior. In *Proceedings of the 8th International Conference on Cognitive Modeling*

Brumby, D. P., Salvucci, D. D., & Howes, A. (2009). Focus on driving: How cognitive constraints shape the adaptation of strategy when dialing while driving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1629-1638).

Gopher, D. (1993). The skill of attention control: Acquisition and execution of attention strategies. In *Attention and performance XIV* (pp. 299–322). Cambridge, MA: MIT Press.

Hornof, A. J., Zhang, Y., & Halverson, T. (2010). Knowing where and when to look in a time-critical multimodel dual task. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

Horrey, W. J., Wickens, C. D., & Consalus, K. P. (2006). Modeling drivers' visual attention allocation while interacting with in-vehicle technologies. *Journal of Experimental Psychology: Applied, 12*, 67-78.

Howes, A., Lewis, R. L., & Vera, A. (2009). Rational adaptation under task and processing constraints: Implications for testing theories of cognition and action. *Psychological Review, 116*, 717-751

Janssen, C. P., & Brumby, D. P. (in press). Strategic adaptation to performance objectives in a dual-task setting. *Cognitive Science*.

Janssen, C. P., Brumby, D. P., & Garnett, R. (2010). Natural break points: Utilizing motor cues when multitasking. In *Proceedings of the 54th annual meeting of the Human Factors and Ergonomics Society*. San Francisco, CA, USA: Human Factors and Ergonomics Society.

Kieras, D. E., Meyer, D. E., Ballas, J. A., & Lauber, E. J. (2000). Modern computational perspectives on executive mental processes and cognitive control: Where to from here? In *Attention and performance XVIII* (pp. 681-712). Cambridge, MA: MIT Press.

Lallement, Y., & John, B. (1998). Cognitive architecture and modeling idiom: An examination of three models of the wickens task. *Proceedings of the twentieth annual conference of the Cognitive Science Society*, 597-602.

Levy, J., & Pashler, H. (2008). Task prioritisation in multitasking during driving: Opportunity to abort a concurrent task does not insulate braking responses from dual-task slowing. *Applied Cognitive Psychology, 22*, 507-525.

Navon, D., & Gopher, D. (1979). On the economy of the human-processing system. *Psychological Review, 86*, 214-255.

Norman, D. A., & Bobrow, D. G. (1975). On data-limited and resource-limited processes. *Cognitive Psychology, 7*, 44-64.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review, 115*, 101-130.

Wang, D. D., Proctor, R. W., & Pick, D. F. (2007). Acquisition and transfer of attention allocation strategies in a multiple-task work environment. *Human Factors, 49*, 995-1004.

Prediction Intervals for Performance Prediction

Tiffany S. Jastrzembski (tiffany.jastrzembski@us.af.mil)
Kelly Addis (kelly.addis@mesa.afmc.af.mil)
Michael Krusmark (michael.krusmark@mesa.afmc.af.mil)
Kevin A. Gluck (kevin.gluck@us.af.mil)
Warfighter Readiness Research Division, Air Force Research Laboratory
6030 S Kent Street, Mesa, AZ 85206 USA

Stuart Rodgers (stu@agstechnet.com)
AGS TechNet, 10887 Miriam Lane
Dayton, OH 45458 USA

Abstract

The Predictive Performance Equation (PPE) is a mathematical model of learning and forgetting developed to capture performance effectiveness across training histories, and to generate precise, quantitative *point predictions* of performance by extrapolating the unique mathematical regularities indicative of the learner. This equation is implemented in the Predictive Performance Optimizer (PPO) cognitive tool, designed to help learners and instructors make principled training decisions through examination of the learning and retention tradespace. Because the point predictive nature of the model implies a high degree of certainty, decision-makers could be misled into making less than optimal decisions in applied settings; and with regards to basic science, the model lacks prediction error and uncertainty which would more accurately represent the predicted range of human performance. Implementation of prediction intervals into a point predictive model of human performance is unprecedented in the psychological literature. We must balance the competing factors of reduced performance variation as practice accumulates, and greater prediction uncertainty as time spans increase. In this paper, we explore new methodologies for incorporating prediction intervals into quantitative predictions of future performance.

Keywords: point prediction; mathematical model; prediction interval; knowledge retention; skill retention

Introduction

The Predictive Performance Equation (PPE) is a mathematical model of learning and forgetting developed to capture performance effectiveness across training histories, and to generate precise, quantitative *point predictions* of performance. This is accomplished by extrapolating unique mathematical regularities indicative of the learner from training history, while additionally accounting for the spacing at which knowledge and skills were trained to estimate the stability of performance across time. This equation is based upon robust findings in the psychological literature, and designed with the intent to be relevant in applied learning domains. As such, the PPE is implemented in the Predictive Performance Optimizer (PPO)–a cognitive tool designed to help learners and instructors make principled training decisions through examination of the learning and retention tradespace.

What the PPE currently lacks is a measure of uncertainty, because it contains no noise or error parameter in its current form. If the model is run 100 times, it will produce the same answer again and again. We know that if a human performs a task 100 times a range of performance values will be produced due to the usual suspects (e.g., distractions, fatigue, fluctuating motivation, random noise) coming into play. Thus, the point predictive nature of the model could be misleading due to the high degree of accuracy implied in its predictions. Therefore, it is necessary to incorporate principled measures of uncertainty, or *prediction intervals (PIs)*, around model point predictions. This provides the likely *range* of performance that is expected, and equips decision-makers with a more thorough picture.

Unfortunately, implementation of PIs into a *hybrid* point predictive model of human performance (to be detailed in the next section) is unprecedented in the psychological literature. By hybrid, we are referring to the notion that one step of the model functions by *calibrating* parameters to available historical data, while the other step *extrapolates* mathematical regularities beyond known data, to make true a priori predictions of performance for practical applications and purposes (e.g., Kahrs & Marquardt, 2007; Psichogios & Ungar, 1992).

Figure 1: Example of prediction uncertainty in the meteorological domain.

Other disciplines, including meteorology, econometrics, and the physical natural sciences, have well-established methods for incorporating uncertainty into time-series model predictions, such that in general, prediction uncertainty increases as time increases (see Figure 1). We may think of this trend as an expanding cone of uncertainty as lead time increases.

In the human performance domain, this is also a fair assumption to make. As the length of time between known data and a prediction increases, uncertainty would be expected to increase (see Figure 2).

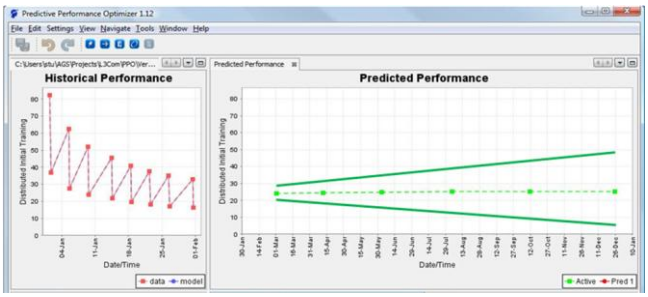


Figure 2: Notional training historical data and predicted refreshers to maintain performance from 1-10 months out.

What meteorological and econometric disciplines do not have to contend with is the fact that as practice accumulates, variability in human performance decreases (e.g., Ericsson, 1996; Rabbitt & Banjeri, 1989). Thus, model uncertainty should *decrease* as practice amasses (see Figure 3).

Figure 3: Expected levels of uncertainty for 3 regimens immediately following a 45-day lag and within a 2-4 day training block.

Furthermore, if multiple predictions are made, as shown in Figure 3, uncertainty is *conditionally* dependent on all previous model predictions. Thus, prediction uncertainty *n*-steps ahead of known empirical training history should generally grow incrementally larger-and prediction uncertainty should additionally be greater after a 12-month lag compared to a 1-month lag.

Thus, we are in the unique predicament of requiring a PI calculation method that balances the competing factors of reduced performance variation as practice amasses, and greater prediction uncertainty as lead time increases. Furthermore, to adhere to both basic and applied science demands, we need to ensure our methods are based on principle, while concurrently providing *useful* and *relevant* guidance for decision-making purposes. Before we turn our attention towards the new methodologies we are exploring to achieve alignment with these trends, we must first detail the nature of the hybrid point predictive human performance model.

The Performance Prediction Equation

The PPE is built upon the strengths of the General Performance Equation (GPE) (Anderson & Schunn, 2000), which handles effects of recency and frequency very well, but is ill-equipped to handle effects of massed versus distributed practice. As such, the PPE formally extends the GPE by capturing effects of spacing, while providing the additional capability to predict performance at later points in time in an a priori fashion. The PPE is expressed as:

$$Performance = S \cdot St \cdot N^{c} \cdot T^{-d};$$

(Equation 1a)

where free parameters include *S*, a scalar to accommodate any variable of interest, *c*, the learning rate, and *d*, the decay rate. Fixed parameters include *T*, the true time passed since the onset of training, and *N*, the discrete number of training events that occurred over the training period. The term *St*, defined in Equation 1b, is short for Stability Term and is responsible for capturing effects of spacing by calculating experience amassed as a function of temporal training distribution and true time passed.

$$St = \left[\frac{\sum lag}{P} \cdot \frac{P_i}{T_i} \cdot \frac{\sum_i^j \left(lag\,max_{i,j} - lag\,min_{i,j} \right)}{N_i} \right];$$

(Equation 1b)

Lag is computed as the amount of wall clock time passed between training events and *P* is computed as the true amount of time amassed in practice. As such, experience and training distribution attenuate performance by affecting knowledge and skill stability at the macro-level of analysis.

Descriptive Adequacy across Data

We have validated the descriptive adequacy and predictive validity of this mathematical model across multiple types of previously published datasets available in the cognitive/experimental psychology literature, including empirical studies spanning knowledge acquisition, knowledge retention, skill acquisition, and skill retention. Goodness-of-fit measures across those domains have achieved an average R^2 of 0.98 (see Jastrzembski & Gluck, 2009, for additional information).

These results are encouraging. However, the datasets available in the psychological literature are from simple laboratory tasks, possessing few data points over an extensive retention period (e.g., Bahrick et al., 1993, study measured performance at seven points over the course of eight years), or measuring performance at short timescales (e.g., Glenberg, 1976, examined monotonic versus non-monotonic effects within one paired-associate training session). These datasets are useful to include in a larger test harness of empirical data to thoroughly validate model mechanisms, but their ecological validity is questionable.

Thus, it is necessary to validate against empirical data from more applied realms - where the interplay of knowledge and skill are often inextricably linked, extended lags between practice opportunities are on the

order of several weeks to multiple months, and knowledge and skill decay across extended lags can have a real impact on mission success. These features often characterize the nature of military training, where resources are both costly and scarce. As such, we validated PPE in a team coordination Unmanned Air Systems (UAS) reconnaissance task (Cooke, 2005), and with F-16 simulator air-to-air combat data collected in the Distributed Missions Operations testbed at the Air Force Research Laboratory (see Jastrzembski, et al., 2009). These highly complex datasets possess significantly longer inter-stimulus intervals than those found in the literature, and provide excellent opportunities to evaluate the incorporation of uncertainty within training blocks and across extended lags, where the need to provide estimates of uncertainty have very clear ramifications.

Predictive Performance Equation Methodology

We will now explain the two distinct, non-stochastic sequential steps in our performance prediction methodology. The first step in using PPE deals with calibrating, or optimizing (using maximum likelihood estimation), the learning and decay parameters to the unique mathematical regularities of the learner, identified by tracking training history. The second step is extrapolating the mathematical regularities to make true a priori predictions of performance at specified future times. We make this distinction because it is commonplace for modelers in the cognitive science community to use the term *prediction* when *fitting* empirical datasets, often in a post-hoc manner; whereas we use the term *calibration* to refer to that fitting process, and prediction for out of sample calculations.

With regard to the UAS reconnaissance study (Cooke, 2005), teams of three individuals were required to coordinate to fly a UAS and attain pictures of targets. They completed five 40-minute missions on the first day of training (the training baseline used for model calibration), and returned either one or three months later to complete an additional three missions (used to validate model a priori predictions) (see Figure 4).

The design of the DMO study was similar in nature, but required teams of four F-16 pilots to fly air-to-air combat missions over a more extensive training baseline (one to two hour-long missions trained each day over for five days), allowing us to examine skill acquisition and decay patterns both within days (where prediction uncertainty should decrease) and across days (where prediction uncertainty should increase). Teams were reassessed either three or six months later and completed three hour-long missions over the course of two training days (see Figure 5 for individual team level analysis).

The need to incorporate valid PIs around model point predictions becomes extremely evident in the following potential use cases, as PPO is indeed intended to help decision-makers make informed training decisions. As shown in Figure 6, PPO may be used to help determine how many additional practice opportunities unique learners (an F-16 pilot team in this case) need to achieve a desired level of performance (denoted as achievement of a wing standard of 0.015 in this particular case).

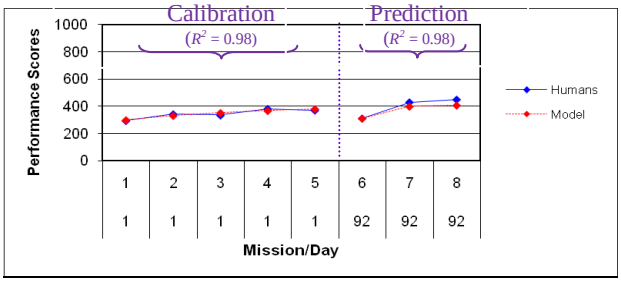


Figure 4: Aggregate team performance in a UAS task, with a three month lag.

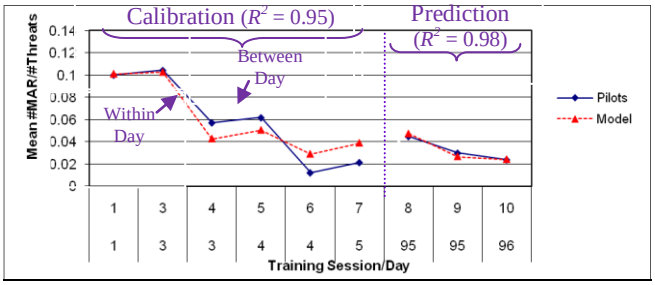
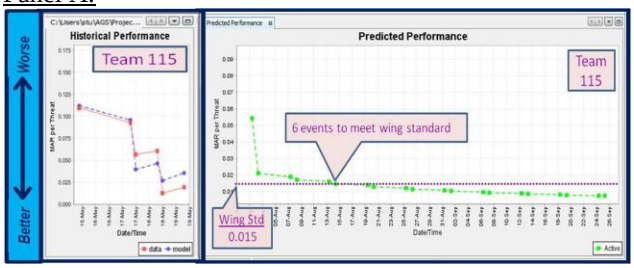


Figure 5: Number of times enemy airspace was violated by an individual F-16 team, with a lag of three months.

PPO takes in the historical data for each unique team, optimizes the learning and decay parameters to the mathematical regularities inherent in the training history, and makes customized team performance predictions by extrapolating those learning trends into the future. Thus, Team 115 (shown in Figure 6, Panel A) is predicted to require six additional training events to achieve the desired performance level, while Team 112 (shown in Figure 6, Panel B) is predicted to require 20 more events.

Panel A:



Panel B:

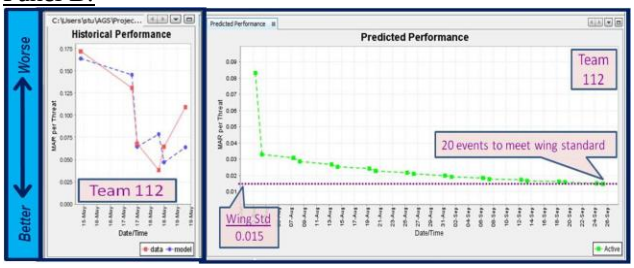


Figure 6: Model predictions for two unique F-16 pilot teams to achieve the same criterion.

In line with statistical principles, as PPO makes multiple time-series dependent predictions, significant *uncertainty* will build for predictions made farther and farther ahead in time from actual historical data. Thus, in the example above, Team 115's predicted attainment of criterion is actually more certain than Team 112's, simply due to the fact that criterion is reached with fewer timesteps ahead from the historically calibrated data.

Another potential use case that nicely demonstrates the need to incorporate "risk" into model point predictions is revealed by PPO's capability to examine performance implications across a multitude of potential future training regimens.

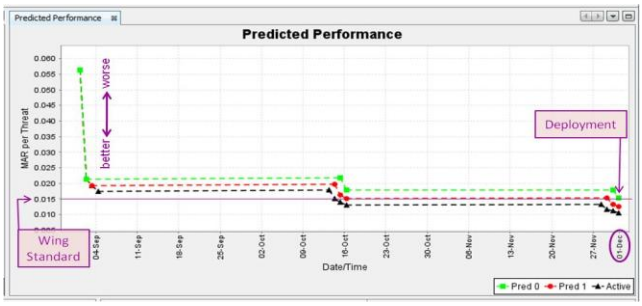


Figure 7: Future training regimen comparisons to identify which training routine best meets desired goals.

The graph revealed in Figure 7 is calibrated upon the historical F-16 pilot team performance data shown in Figure 6, Panel A, and depicts predicted levels of performance under three distinct training regimens. The green line depicts two training opportunities given in each training block (occurring every 45 days), while the red line reveals three, and the black line reveals four. Noting that a desired performance effectiveness level of 0.015 is to be reached by the intended deployment date, the learner or instructor may easily inspect and assess the efficiency and effectiveness each potential future training regimen will likely provide.

As shown in Figure 7, the red and black lines both achieve the desired performance level by deployment, while the green line does not. However, PIs for the black line should theoretically be smaller than those in the red line - because more training opportunities are provided meaning performance variability should be reduced. Thus, less risk would be involved in deploying trainees who completed the black training regimen.

Given the potential ramifications these types of prospective use case decisions entail, it becomes very clear why the incorporation of prediction uncertainty measures is needed. Further, equipping PPE with these measures will better aid decision-makers' understanding both learning and training needs, as well as the risks.

Prediction Interval Calculation Methodology

As previously expressed, there is no precedent for incorporating PIs into a human performance point prediction model of this nature. As such, we have developed and are investigating new methods to achieve our goals of both reducing variability as practice amasses, and increasing variability at longer lead times.

Extrapolation of Residuals

The first method we are investigating involves extrapolating residuals from calibrated model predictions and human empirical data to model point predictions. Residuals are often used to add uncertainty to models in other disciplines, like econometrics (see Chatfield, 2001, for a review); but as previously mentioned, other disciplines do not have the added phenomenological complexity of uncertainty decreasing as practice increases, nor do they have good solutions for estimating *how much* larger PIs should be after lags of increased length. Thus, in order to base a PI method on residuals in the human performance domain, a good deal of innovation will be required to ensure estimates stay true to expected human performance trends.

As such, we have modified the residuals by the stability term (see Expression 1) and will illustrate PI incorporation based on this method later in this paper.

$$S * St * N^{c} * T^{-d} \pm (z_{\alpha/2} * E[RMSD] * St_{i+h});$$

(Expression 1)

The Coefficient of Variation

The second method we have developed and are continuing to investigate deals with adding variability into the learning and decay parameters themselves. The amount we have chosen to vary parameters by is the coefficient of variation (CV), selected because it is a unitless measure of deviation between model predictions and human empirical data, generally ranging between zero and one (Schweickert, et al., 2003), and it has previously been used to incorporate stochasticity into other types of cognitive and task performance models (Patton, et al., 2009; Patton & Gray, submitted; Schweickert, et al., 2003). It is calculated across historical training calibration data using Equation 2:

$$CV = RMSD/model\ mean;$$

(Equation 2)

and integrated into PPE in the following way (see Expression 2):

$$S * St * N^{c \pm c*CV} * T^{-(d \pm d*CV)};$$

(Expression 2)

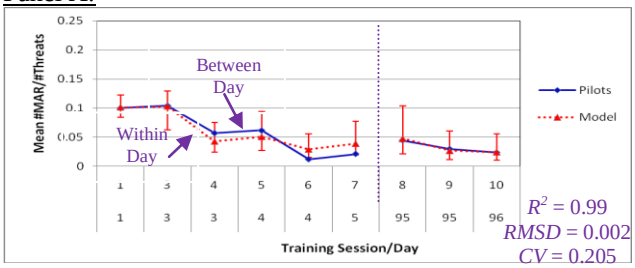
thus producing upper and lower PI bounds.

Desirable qualities of this measure include a readily available mapping to the learning and decay rates, which also range from zero to one; and greater variability being added into models that produce lower quality calibrated fits to empirical data, producing larger PIs as a result.

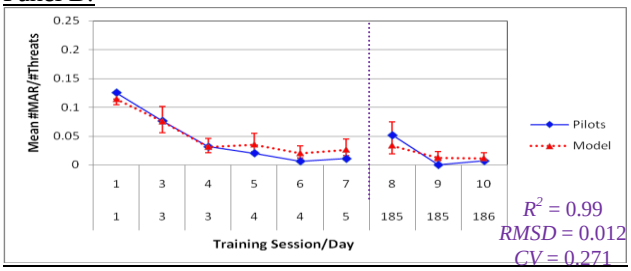
Prediction Interval Utility in the Applied Domain

We now illustrate the PI incorporation across four unique F-16 pilot teams, possessing differences in learning regularities and quality of calibration fit – leading to differences in PI spans as a result (see Figures 8 and 9).

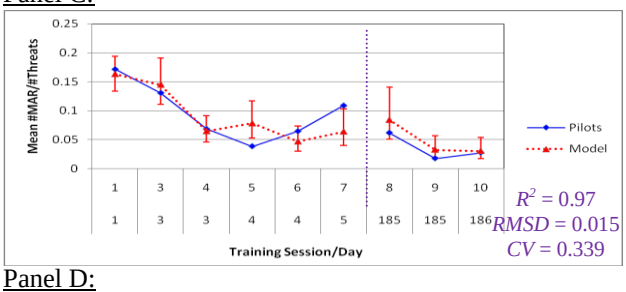
Panel A:



Panel B:



Panel C:



Panel D:

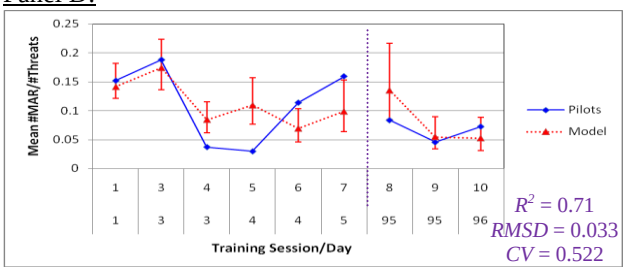
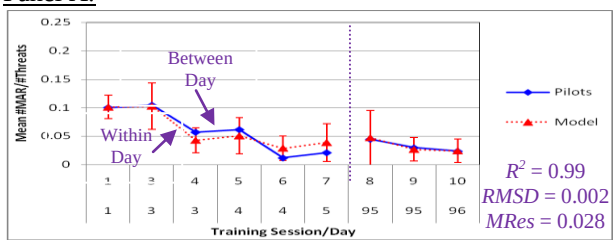


Figure 8: CV PI incorporation for F-16 pilot teams.

As revealed by Figures 8 and 9 (Figure 9 displays identical empirical data displayed in Figure 8, Panels A and D), each method produces larger PIs *between* training days and smaller PIs *within* training days – thus, mapping nicely onto human empirical findings showing that performance variation decreases as practice amasses. They also reveal wider PI bands following the three or six month lag relative to other predicted points; thereby aligning with the notion that longer lead time predictions are more uncertain than predictions at shorter lead times.

An added unexpected, but very desirable effect, of the CV method was that the PI bands are asymmetrical in nature – thereby diverging from standard symmetrical estimates of confidence or error (as revealed by the residual-based method). This is pleasing in cases where human performance is bounded by a floor or ceiling, (ceiling performance was zero on the y-axis in Figures 8 and 9). Thus, there is more room to err (the higher end of the y-axis) and less room to gain (performing closer to zero), mapping nicely to CV-based error bars having longer upper than lower whiskers.

Panel A:



Panel B:

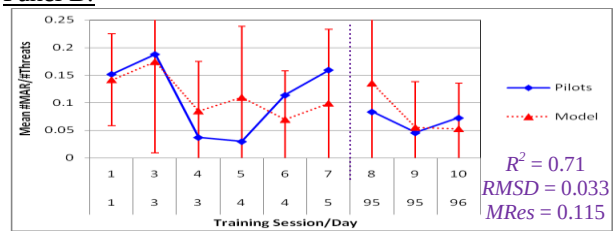


Figure 9: Residual-based PIs across unique F-16 teams.

Comparison of these PI methods to empirical data reveal that utilization of residuals, compared to the CV-based method, tends to produce larger error bars in general (it is more liberal, but covers more of the data), produces error bands outside the bounds of possible performance (below zero in this case), and is more sensitive to noisy data (see Figure 9, Panel B – the same empirical data as Figure 8, Panel D). This raises concerns for how useful a residual-based approach will be as a decision-making guide. As such, additional modifications are being examined.

Resolution of Data In our last set of analyses, we will limit our discussion to the CV PI methodology, due to limitations of the residual-based method described above. Using data collected in the UAS reconnaissance task (Cooke, 2005), we applied PIs to models aggregated at different grains of analysis. Given the intended utility of the PPO as a principled training decision guide, it is important to understand the implications of using a predictive model at the aggregate, team, and individual learner levels of performance (see Jastrzembski, et al., 2006), as aggregate data, by definition, reduces noise through averaging procedures that smooth out the shape of human performance curves. Thus, data will always be noisier at finer and finer grains of resolution, implying PIs should be wider and wider as aggregation decreases. We inspect the ability of the CV PI method to align with this phenomenon as shown in Figures 10-12 below.

As we might expect, PIs for the first point prediction after the lag are indeed larger after the long delay (PI_{range} = 146) compared to the short delay (PI_{range} = 129), revealed in Figure 9, showcasing the fact that predictions at longer lead times will be less certain than predictions at shorter lead times. This effect is generated in PPE because the upper and lower CV bounds are placed in the learning and decay exponents, which interact with the number of training opportunities accumulated, as well as the actual amount of time passed.

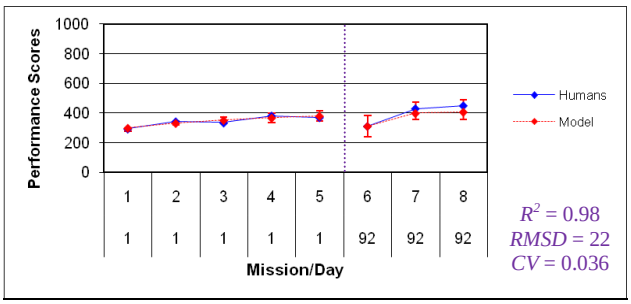


Figure 10: Aggregate performance across all teams in the UAS reconnaissance task, with lags of 30 or 91 days.

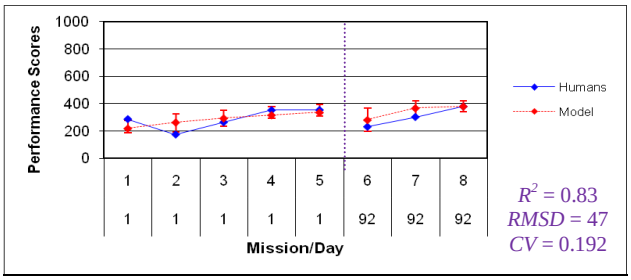


Figure 11: Individual UAS team performance.

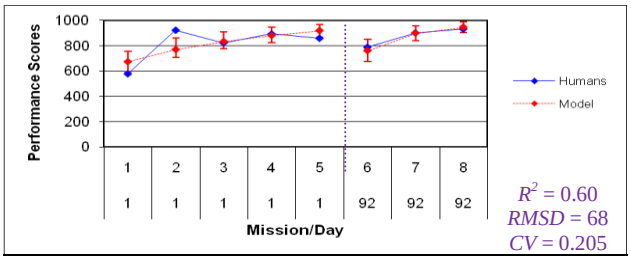


Figure 12: Individual UAS team member performance.

Finally, we note that the CV increases as we move from aggregate to team to individual levels of performance, as expected (see Figures 10-12). This is a useful property to note because it shows that decisions may be riskier at finer grains of resolution. One way to help circumvent this problem at finer grains of analysis is to in fact accumulate larger training histories to calibrate PPE upon, allowing variability and noise to be smoothed.

These illustrative exercises help lend credence to the notion that use of this newly developed CV PI calculation method may have merit as being a useful way to help guide training decisions in a way that nicely accounts for the competing trends of reduced performance variability expected with increases in practice, and increased prediction uncertainty expected for longer lead times.

Conclusions

The incorporation of estimates of uncertainty into model point predictions is a necessary extension to our point predictive model in order to provide learners and instructors with relevant and useful guidance concerning the amount of predictive uncertainty that should be expected at specific future points in time and under competing future training regimens. Because there are no precedented existing methodologies to apply to this problem, we plan to further the validation effort across the two potential solutions we proposed in this paper against human empirical data, and we are hopeful this new capability will apply not only to our modeling effort, but also for others who are working on the optimization of training (e.g., Lindsey, et al., 2009; Pavlik, & Anderson, 2008; van Rijn et al., 2009).

References

Bahrick, H., & Phelps, E. (1987). Retention of spanish vocabulary over 8 years. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 13*, 344-349.

Chatfield, C. (2001). Prediction intervals for time series. In J.S. Armstrong (Ed.), *Principles of Forecasting: A Handbook for Practitioners and Researchers*. Norwall, MA: Kluwer.

Ericsson, K. A. (Ed.) (1996). *The Road to Excellence: The Acquisition of Expert Performance in the Arts and Sciences, Sports, and Games.* Mahweh, NJ: Erlbaum.

Glenberg, A. M. (1976). Monotonic and nonmonotonic lag effects in paired-associate and recognition memory paradigms. *Journal of Verbal Learning & Verbal Behavior, 15*, 1-16.

Jastrzembski, T. S., & Gluck, K. A. (2009). A formal comparison of model variants for performance prediction. *Proceedings of the International Conference on Cognitive Modeling (ICCM)*, Manchester, England.

Jastrzembski, T. S., Rodgers, S., & Gluck, K. A. (2009). Improving military readiness: A state-of-the-art cognitive tool to predict performance and optimize training effectiveness. *Proceedings of the I/ITSEC annual meetings*, Orlando, Florida.

Kahrs, O., & Marquardt, W. (2007). The validity domain of hybrid models and its application in process optimization. *Chemical Engineering and Processing, 46*, 1054-1066.

Lindsey, R. Mozer, M., Cepeda, N., & Pashler, H. (2009). Optimizing memory retention with cognitive models. *Proceedings of the ICCM* annual meeting, Manchester, England.

Patton, E. W., & Gray, W. D. (submitted). SANLab-CM: A tool for incorporating stochastic operations into activity network modeling. *Behavior Research Methods*.

Patton, E. W., Gray, W. D., & Schoelles, M. J. (2009). SANLab-CM – The stochastic activity networking laboratory for cognitive modeling. *Proceedings of the 53rd HFES Annual Meetinge*, San Antonio, Texas.

Pavlik, P. I., & Anderson, J. R. (2008). Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied, 14*, 101-117.

Psichogios, D. C., & Ungar, L. H. (1992). A hybrid neural network – first principles approach to process modeling. *AIChE Journal, 38(10)*, 1499-1511.

Rabbitt, P., & Banjeri, N. (1989). How does very prolonged practice improve decision speed? *Journal of Experimental Psychology: General, 118*, 338-345.

Schweickert, R., Fisher, D. L., & Proctor, R. W. (2003). Steps toward building mathematical and computer models from cognitive task analyses. *Human Factors, 45(1)*, 77-103.

van Rijn, H., van Maanen, L., & van Woudenberg, M. (2009). Passing the test: Improving learning gains by balancing spacing and testing effects. *Proceedings of the ICCM* annual meeting, Manchester, England.

Exploration of Costs and Benefits of Predictive Human Performance Modeling for Design

Bonnie E. John (bej@cs.cmu.edu)
Human-Computer Interaction Institute, Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213

Tiffany S. Jastrzembski (tiffany.jastrzembski@wpafb.af.mil)
Air Force Research Laboratory
2698 G Street, Building 190
Wright-Patterson AFB, OH 45433-7604

Abstract

Human performance modeling promises to be a valuable tool for early evaluation of user interface designs, predicting different performance for different design alternatives and, recently, different performance on a single design between younger and older adults (Jastrzembski & Charness, 2007; Jastrzembski, et al., 2010). When using modeling in the development process, the costs of creating models must be traded-off against the accuracy needed to guide design choices. It is therefore a meaningful exercise to examine and weigh the costs and benefits of different modeling approaches, to provide practitioners information to help them choose the modeling approach best suited for their needs. We compare younger and older adult human performance data captured from dialing and text-messaging tasks, across two mobile phones, against age-specific GOMS (Card, Moran & Newell, 1983) and various CogTool models (John, et. al. 2004), and examine the trade-offs between time and effort required to build those models and the predictive validity each model produces.

Keywords: predictive human performance modeling, design.

Introduction

Research in computational cognitive process modeling continues to progress by creating models able to account for human data on more tasks across more domains, often through years of effort by PhD students, post-docs and/or senior researchers. However, when practitioners wish to use cognitive modeling approaches in user interface (UI) design, issues of costs and benefits become a stark reality. It is therefore often necessary for the practitioner to base the selection of a modeling approach by trading off the costs of producing the human performance models against the desired accuracy of the predictions of those models.

The costs of producing models for design include how much knowledge the practitioner must have to develop an appropriate cognitive model in the task domain of interest for the intended user group, learning and understanding the modeling theory that underlies a modeling tool, learning how to use the modeling tool itself, and the time it takes to accurately implement the models after learning the modeling theory and associated tools. Benefits include the ability of a modeling approach to detect differences between design alternatives and the ability to make accurate predictions of quantitative measures of performance (e.g., time for a skilled user to execute a task or number of errors).

As the consumers of interactive systems age it is becoming economically important to evaluate designs specifically for the older adult. Thus, an additional concern we address in this paper, are costs related to modifying existing modeling approaches and tools to account for the human processing capabilities of the older adult. Given the range of knowledge, time, and effort required to make these model modifications, this paper compares the quality of predictions against the efficiency of each approach.

To put these issues into context, consider a practitioner who is under a tight deadline to choose a final design that is efficient for both younger and older adults from among several design alternatives. A less time-intensive modeling approach may be required to fit into the development life cycle, even if use of that modeling approach comes at the cost of producing less accurate predictions. This paper begins to address cost-benefit concerns by assessing the accuracy of a variety of modeling approach predictions against empirical data, and examining the costs incurred to produce those predictions.

The Designs, Tasks & Empirical Results

We chose to examine two tasks on two mobile phones because Jastrzembski and Charness (2007) provides pre-existing empirical data for younger and older adults. The tasks are dialing a 10-digit phone number (*dialing*) and sending a text message to a person in the contact list (*texting*). Participant groups included a sample of younger adults (M_{age} = 20) and older adults between the ages of 60-75 (M_{age} = 69). The purpose of their study was to validate elemental model human processing parameters updated to account for the older adult, which had been estimated through a comprehensive literature review. These parameter values were then used to build age-specific GOMS (Goals, Operators, Methods, and Selection rules) models (Card, et al., 1983) to predict skilled performance of younger and older "average" adults in the mobile phone tasks.

Figure 1. Mobile phones studied by Jastrzembski and Charness (2007) and used in this analysis: the Nokia 3595 (left) and the Motorola C155 (right).

Predictions were compared to empirical data at each button press required by the task.[1]

Since GOMS models are designed to predict performance of skilled users on routine tasks, the participants were required to complete extensive practice sessions to ensure that they were skilled in the performance of these tasks on these devices. "Skill" was operationally defined as completing three consecutive trials with less than a 1s deviation from each other. Upon successfully achieving criterion in the practice sessions, participants were then given new stimuli to complete three repeated blocks of five different trials for each task. This allowed the authors to average the human performance data for a single stimulus over three trials – thus producing the empirical findings displayed in Figure 2.

The following results were revealed (Figure 2, Table 1).

- Older adults took significantly longer than younger adults to complete both tasks on both phones.
- Dialing completion times were not significantly different across phones for either age group.
- Text-messaging completion times were significantly longer using the Motorola compared to the Nokia phone for both age groups.

These findings give us an interesting spread of results to assess the evaluation of the designs across modeling approaches from a cost-benefit perspective. In order for a model to be useful in practice, it must account for all three results, i.e., detecting a difference between devices and age groups where this is one in the empirical data and detecting no difference where there isn't.

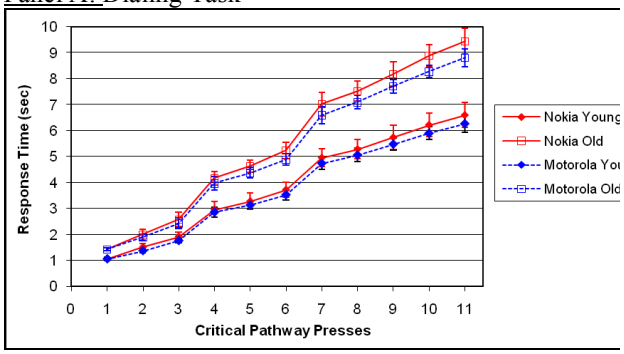
The Modeling Approaches & Their Results

Seven modeling approaches were implemented for the dialing task and four for the texting task, as described below (Table 1 displays completion time results).

GOMS-MHP. A pre-existing model by Jastrzembski & Charness (2007), this approach updated Model Human Processor (MHP) parameters through extensive literature review, to allow GOMS models to predict older adult performance. These models most closely match the "K2" models put forth by Card, et al., (1983, p. 166), where operators are at the level of hundreds of milliseconds, and map closely to MHP cycle times. The cognitive task analysis that underlies these models was informed by observing pilot participants using an eye-tracker while performing the tasks. Eye-fixation operators and subsequent decisions operators were placed in the models guided by these data. These models achieved excellent fits to the human data for tasks, phones, and age groups.

CogTool-OotB. The next modeling approach we examine is CogTool (John, et al., 2004), a tool for prototyping UI designs and automatically producing Keystroke-Level Models (KLM, Card, et al., 1983) through demonstration.

Panel A: Dialing Task



Panel B: Texting Task

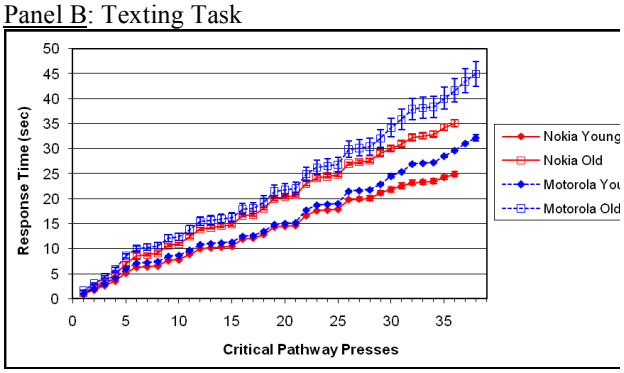


Figure 2. Empirical data for younger and older adults completing tasks on the Nokia and Motorola phones.

KLM is a simplified form of GOMS that sums each key press, *K* (including typing on a keyboard and mouse clicks); pointing movement, *P*; homing movement between devices, *H*; system response time, *R*; and "mental operator", *M* (an averaged amalgamation of visual search, perception and cognitive operations like deciding, recalling commands, etc.), required to do a task.

CogTool automates KLM model construction through a demonstration of a task on a storyboard of a UI, adding perceptual operations in line with Salvucci (2001), and cognitive operations similar to Card, Moran And Newell's Ms[2], called "Think operators." The resulting *script* approximate a KLM produced by an expert modeler. The storyboard and script together compile into an ACT-R model (Anderson & Lebiere, 1998), which runs to produce quantitative predictions of skilled performance time. CogTool allows people with no cognitive psychology or modeling background to make accurate predictions with little variance (John et al., 2004; John, 2010).

This approach used CogTool "out of the box" examining its default predictions without modifications of the script it produced or to any of its parameters. This approach resulted in far better predictions for the texting task than for the dialing task. The remaining approaches progressively add information to this "out of the box" approach.

[1] The original GOMS parameters were set with data from younger adults, therefore we will use the original GOMS parameters for younger adults unless otherwise noted in this paper.

[2] Card, et. al. (1983) set the duration of *M* to 1.35s, but CogTool uses 1.2s because it has separate processes for eye movement and visual perception, which require about 0.15s processing time.

CogTool+KLM. To improve predictions for the dialing task, our third modeling approach brought knowledge of the KLM to bear, editing out Think operators where they violated Card, Moran & Newell's M-placement rule concerning *cognitive units*. We deemed this approach reasonable because people separate US telephone numbers into cognitive units consisting of a 3-digit area code, a 3-digit exchange, and a 4-digit station number. Because *CogTool-OotB* does not automatically identify these units, analysts must use their knowledge to delete unnecessary Think operators from the scripts. (Such modification was reasonable for the dialing task, but not for the texting task where *CogTool-OotB = CogTool+KLM*.)

CogTool+KLM+RatioThink. Since CogTool generates predictions specific to younger adults, it cannot make predictions for older adults without modifications. Therefore, our fourth modeling approach augments CogTool+KLM by applying Hale and Myerson's (1995) findings that older adults take 1.5 times as long as younger adults to process linguistic information. This means that the analyst simply copies the original CogTool+KLM script for a task and edits each Think to be 1.5 times as long as the standard younger adult time (i.e., 1.8s v 1.2s). This resulted in an average absolute percent error of less than 10% for the texting task, but 36% for the dialing task – vastly over predicting the time it takes both young and old to dial a phone number (see Table 1).

CogTool+KLM+RatioThink+ExtremePractice. Reflecting on the previous method's poor fit to the dialing task data, we realized that participants in 2005 would have had almost a lifetime of experience dialing touch-tone phones and substantially less practice sending text messages on mobile devices. Prior research in extreme practice has shown that pauses indicating mental operations almost disappear. Thinking is both getting shorter with practice and also presumably happens in parallel with the perceptual and motor actions necessary to do the task (e.g., Card, et al., 1983, pp. 279-286). Simulating extreme practice is an easy process in CogTool; the analyst simply deletes every Think step in the script except the first (which is still required because the digits must be visually acquired from a sheet of paper). This resulted in predictions that better fit the younger and older adult data. However, these predictions were within 10% of each other, meaning that these models no longer detected the main effect of age.

CogTool+KLM+RatioThink+ExtremePractice+OlderWMcapacity. Our next approach examines the accuracy of a CogTool model created by analysts possessing additional information about older adult performance, as was uncovered by Jastrzembski & Charness' (2007) literature review. That review revealed that the working memory (WM) capacity of older adults is smaller than that of younger adults. This may cause a strategy change in older adults; they may spend more time with written instructions than younger adults, trading off time for accuracy. With this insight, we put the Think steps associated with looking at the paper for the area code, exchange and station digits, back into the older adult dialing task models. This reduced

Table 1. Modeling approach predictions for the mobile phone dialing task with percent deviations from empirical data.

		Nokia				Motorola			
	Abs Avg	Younger		Older		Younger		Older	
Source of data or predictions	**%diff**	**Time (s)**	**%diff**	**Time (s)**	**%diff**	**Time (s)**	**%diff**	**Time (s)**	**%diff**
Dialing Task									
Human Data		6.606		9.442		6.268		8.812	
GOMS-MHP	0.6%	6.559	-0.7%	9.369	-0.8%	6.228	-0.6%	8.804	-0.1%
CogTool-OotB	169.9%	16.451	149.0%	n/a	n/a	18.227	190.8%	n/a	n/a
CogTool+KLM	44.1%	9.171	38.8%	n/a	n/a	9.359	49.3%	n/a	n/a
CogTool+KLM+RatioThink	36.0%	9.171	38.8%	11.571	22.6%	9.359	49.3%	11.759	33.4%
CogTool+KLM+RatioThink +ExtremePractice	15.5%	5.976	-9.5%	6.576	-30.4%	6.302	0.5%	6.902	-21.7%
CogTool+KLM+RatioThink +ExtremePractice +OlderWMcapacity	5.0%	5.976	-9.5%	8.092	-14.3%	6.302	0.5%	8.387	-4.8%
CogTool+KLM+RatioThink +ExtremePractice +OlderWMcapacity +LitReviewACT-Rparameters	6.4%	5.830	-11.8%	9.505	0.7%	6.205	-1.0%	9.520	8.0%
Texting Task									
Human Data		24.905		35.127		32.186		44.991	
GOMS-MHP	0.0%	24.901	0.0%	35.126	0.0%	32.153	0.1%	44.989	0.0%
CogTool-OotB (=CogTool+KLM)	13.9%	27.582	10.7%	n/a	n/a	37.664	-17.0%	n/a	n/a
CogTool+KLM+RatioThink	9.4%	27.582	10.7%	35.382	0.7%	37.664	-17.0%	49.064	9.1%
CogTool-KLM+RatioThink +LitReviewACT-Rparameters	11.7%	27.148	9.0%	37.442	6.6%	37.118	-15.3%	52.177	16.0%

the average absolute percent error to 5% for the dialing task.

CogTool+KLM+RatioThink+ExtremePractice+OlderWMcapacity+LitReviewACTRparameters. The last modeling approach modifies the ACT-R models running under the hood of CogTool. This approach requires both more cognitive psychology knowledge and programming skill. It leverages the aforementioned literature review as well as Jastrzembski, et al.'s (2010) translation and extension of age-specific parameters to ACT-R. We ran CogTool in a development environment rather than as an executable, and edited four specific underlying ACT-R parameters identified by Jastrzembski, et. al. (2010), in order to account for age. We modified the best of the CogTool approaches previously mentioned (*CogTool+KLM+RatioThink+ExtremePractice+OlderWM capacity* for dialing and *CogTool-OotB* for texting), but results produced overall goodness-of-fit values slightly less than other approaches, for both dialing and texting tasks.

Cost and Benefit Metrics

We now assess the costs each modeling approach would incur, based upon the estimated amounts of knowledge, time, and effort required to produce predictions using each method. Benefits are assessed relative to the empirical data collected by Jastrzembski and Charness (2007), which will be considered "*the gold standard*" - that is, the "truth" against which the models will be compared. Costs are assigned a value pertaining to the length of time required to attain the appropriate knowledge base and perform the modeling itself. A *large* cost entails months of experience to learn and/or use the method; a *medium* cost requires weeks of training and use; a *small* cost requires days.

Of course, actual costs to an organization depend on both workforce and resources. For instance, empirical collection of human data is characterized as having a *large* cost in this analysis because many practitioners are not trained in experiment design, they lack data collection laboratories, and they often do not possess statistical packages or analytic know-how to properly interpret the empirical data. These costs may be much smaller for organizations like Google or Microsoft, which already have highly equipped labs, PhD-level experimentalists and statisticians, and a network for recruiting appropriate participants.

In addition, the costs are estimated for moving into a new domain or user group where parameters are not already routinely used in models or built into tools. Many of these estimates would reduce as modeling knowledge increases and tool functionality is enhanced to embody that knowledge. Given these caveats, we identified the following costs for the analyses described in this paper.

Collect Human Data. Cost = Large because of expertise and resource issues discussed above, and because participants must be trained to a skilled level of performance on the tasks and devices studied.

Literature Review. Cost = Large for a full review and meta-analysis (it took Jastrzembski approximately nine months to complete the parameter estimation alone). Cost = Small if only a rule-of-thumb 50% increase (as reported by Hale & Myerson (1995)) is used.

Program a running prototype. Cost = Large due to required programming skill expertise (UI designers often possess graphic design backgrounds rather than a computer science backgrounds to compound the problem).

Measure for Fitts's Law. Cost = Small because estimates of size and distance between all keys are required for movement times to be integrated into models. (Although it does not take days to learn or accomplish this, the sheer tedium bumps this, in our estimation, into a real cost).

Build a Storyboard. Cost = Low because building a storyboard in CogTool (John, et. al., 2004) involves only creating a frame using a picture of what the device looks like, placing button widgets on that frame, and drawing transitions to represent user actions required to accomplish the task. Storyboards for the two phones used in this investigation took the first author about 15 minutes to build.

Know GOMS/MHP. Cost = Large. In the first author's 25 years of experience teaching GOMS, it takes engineers several sessions to learn the typical version of GOMS but requires feedback on multiple exercises and often an apprenticeship with an expert GOMS model builder to be able to produce high-quality models. The GOMS-MHP models assessed here were built with PhD-level knowledge of cognitive psychology guided by eye-tracking observations (Jastrzembski, 2006).

Know KLM. Cost = Small. In the first author's 25 years of experience teaching GOMS, KLM can be taught in a single class session but requires feedback on several exercises to be able to remove mental operators appropriately to account for "cognitive units" (John, 1994). The cost increases to Medium when knowledge of different strategies due to older adults' smaller WM span and the effects of extreme practice are required in the model.

Know CogTool. Cost = Small. Recent research has shown that CogTool can be taught in one class session and novice modelers building their first model produced predictions within 4% of an expert's model prediction, with a CV of only 7% (John, 2010).

Edit ACT-R. Cost = Large. In the final approach we studied, the practitioner must edit an ACT-R file to modify specific parameters to those established for younger and older adults (Jastrzembski, et al., 2010). This requires accessing CogTool's open source code, editing the code in the Eclipse programming environment, and knowing how and where to change the parameters. Although it is only four lines of Lisp code, the knowledge necessary to perform this procedure is, in our estimation, daunting, and would be required until CogTool could be enhanced to provide a GUI to switch between user groups.

There are two types of benefits possible in our analysis: the ability to correctly detect a difference between devices or user populations, and the numeric accuracy of its predictions. An approach is assigned a *large thumbs-up* when it correctly detects a statistically-significant difference present in the human data and, just as important for design,

Table 2. Assessment of costs and benefits of empirical data collection and seven modeling approaches.

Costs									Approach	Benefits					
										Detecting Differences		Match to Observed Times			
Collect Human Data	Program a running prototype	Literature Review	Measure for Fitts's Law	Build Storyboard	Know GOMS/MHP	Know KLM	Know CogTool	Edit ACT-R		Detect difference between devices	Detect difference between age groups	Dialing-Young	Dialing-Old	Texting-Young	Texting-Old
									Empirical: Train participants to skilled level, then collect data *(Human Data)*						
									GOMS + literature parameters for all operators *(GOMS-MHP)*						
									CogTool "Out of the box", naïve use *(CogTool-OotB)*						
									CogTool + KLM knowledge to delete Think operators *(CogTool+KLM)*						
									CogTool+KLM + 1.5:1 ratio (Hale & Myerson, 1995) for Think operators *(CogTool+KLM+RatioThink)*						
									CogTool+KLM+RatioThink + deleting all but the first Think operator to account for extreme practice dialing *(CogTool+KLM+RatioThink+ExtremePractice)*						
									CogTool+KLM+RatioThink +ExtremePractice + WM capacity of Older Adults causes more looking at task description in Dialing task *(CogTool+KLM+RatioThink+ExtremePractice+OlderWM)*						
									Best CogTool from Tables 1&2 + literature review parameters for all ACT-R operators *(CogTool(Best)+LitReview ACT-R parameters)*						

Key

- Months to acquire knowledge or do this work
- Weeks to acquire knowledge or do this work
- Days to acquire knowledge or do this work
- Detects difference when there is one and not when there isn't, or numerically within 5% of human data
- Does not (or cannot) detect difference when there is one
- Numerically within 5-10% of human data
- Numerically within 10-20% of human data
- Numerically greater than 20% of human data

does not claim a difference when there is no statistically-significant difference in the human data; a *large thumbs-down* is assigned when this is not the case. With respect to numeric accuracy, we assigned each prediction to one of four categories as shown in the key in Table 2.

Discussion of Costs & Benefits

The results of our assessments appear in Table 2. As mentioned before, collecting human data is considered the gold standard in UI design practice, but its cost is high, particularly for organizations with little staff or resources for experiment design, collection and analysis. Jastrzembski and Charness's GOMS-MHP modeling produced excellent predictions, but required eye-tracking and PhD-level understanding of the psychology literature and the Model Human Processor in order to attain those levels of predictive accuracy.

CogTool-OotB is less costly to learn and use, even for people with no psychology background. It correctly detected the difference between the devices when there was one in the data (for texting), but it was not designed to detect age-related performance differences, as it applies only to the performance of younger adults. Only by augmenting that tool with levels of knowledge of KLM and age-related literature, do models constructed within CogTool approach the level of accuracy useful for UI design if age is a factor. In fact, when only consideration of extreme practice is taken into account, the CogTool models produced fail to detect the

age-related differences in the dialing task. Only when the combination of extreme practice and WM capacity for older adults were incorporated, did the predictions fall into alignment with the empirical results. This requires substantial knowledge of the psychology literature that many practitioners would likely not possess.

Finally, the addition of specialized ACT-R parameters for younger and older adults in fact *increased* the average absolute percent error, demonstrating that utilization of substantially increased requirements of knowledge and skill (ACT-R, Eclipse & Lisp) does not always improve predictions sufficiently to warrant the increased effort.

Conclusions & Future Work

This research compares the *efficiency* and *effectiveness* of a variety of modeling approaches across tasks, designs, and user populations. There is no "right answer" for any particular development project, as each will vary in their need for accuracy, the current knowledge and skill of their team, and the value placed on acquiring modeling skill for future use. For example, if a design project must have predictions for all tasks within 5% of the "gold standard", the only approaches we examined achieving that criterion are empirical data collection[3] or GOMS-MHP modeling, with their associated high costs. However, if slightly less accurate predictions are acceptable, CogTool models augmented with some knowledge of KLM and psychology may be useful. Table 2 should be considered a guide when considering modeling, not a table of definitive recommendations.

Furthermore, advocates of using models in the development process always suggest that modeling can be used in conjunction with empirical testing, i.e., quick and easy CogTool modeling could be used as a means of weeding out detectibly poor designs from an assortment of design options in a tractable amount of time, so that empirical data collection may then be used to evaluate the few remaining candidates where accuracy is of high value. No one method need stand alone.

Several areas of future tool development are suggested by this investigation, pending, of course, repeatability of these results. First, if age-specific Think values detect age-related differences on other tasks on other devices, it would be a simple matter to put a radio button in the CogTool UI to allow analysts to select younger or older adults and attain appropriate predictions without editing scripts. Likewise, if future research showed that age-specific ACT-R parameters increased accuracy in the majority of cases, they also could be brought into play without analysts touching the underlying ACT-R Lisp code. Thus, it is beneficial to examine the costs and benefits of modeling approaches periodically, because such examinations may be used to improve model tool development, and allow us, as a field, to change the costs associated with the most useful approaches.

Acknowledgments

This research was supported in part by funds from IBM, NASA, NEC, PARC, and ONR N00014-03-1-0086. The views and conclusions in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of IBM, NASA, NEC, PARC, ONR, AFRL, or the U.S. Government.

References

Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Erlbaum.

Card, S. K., Moran, T.P. and Newell, A. (1983) *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, xlvii,* 381–391.

Hale, S., & Myerson, J. (1995). Fifty years older, fifty percent slower? Meta-analytic regression models and semantic context effects. *Aging and Cognition, 2,* 132–145.

Jastrzembski, T. S. (2006). The Model *Human Processor and the Older Adult: Validation and Error Extension to GOMS in a Mobile Phone Task.* Unpublished doctoral dissertation. Psychology Department, Florida State University, Tallahassee, FL.

Jastrzembski, T. S., & Charness, N. (2007). The Model Human Processor and the older adult: validation in a mobile phone task. *Journal of Experimental Psychology: Applied*, 13, 224-248.

Jastrzembski, T. S., Myers, C., & Charness, N. (2010) A principled account of the older adult in ACT-R: Age-specific model human processor extensions in a mobile phone task. To appear in *Proceedings of the Human Factors and Ergonomics Society 54nd Annual Meeting*, (San Francisco, CA, September 27-October 1, 2010).

John, B. E. (1994) Toward a deeper comparison of methods: A reaction to Nielsen & Phillips and new data. In *Proceedings Companion of CHI, 1994* (Boston, MA, April 24-28, 1994) ACM, New York, NY. 285-286.

John, B. E., (2010) Reducing the Variability between Novice Modelers: Results of a Tool for Human Performance Modeling Produced through Human-Centered Design. *Proceedings of the 19th Annual Conference on Behavior Representation in Modeling and Simulation* (Charleston, SC, March 22-25, 2010).

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '04.* ACM, New York, NY, 455-462.

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, *1*, 201-220.

Integrating Fast and Slow Cognitive Processes

William G. Kennedy (wkennedy@GMU.Edu)
George Mason University, 4400 University Drive
Fairfax, VA 22032 USA

Magdalena Bugajska (Magda.Bugajska@NRL.Navy.Mil)
Naval Research Laboratory, Code 5515, 4555 Overlook Ave. SW
Washington, DC 20585 USA

Abstract

Human reactions appear to be controlled by two separate types of mental processes: one fast, automatic, and unconscious and the other slow, deliberate, and conscious. With the attention in the literature focused on the taxonomy of the two processes, there is little discussion of how they interact. In this paper, we focus on modeling the slower process's ability to inhibit the fast process. We present computational cognitive models in which different strategies allow a human to consciously inhibit an undesirable fast response. These general strategies include (a) blocking sensory input, (b), blocking or interrupting the fast process's response, and (c) slowing down or delaying processing by introducing additional task. Furthermore, we discuss an approach to learning such strategies based on the inference of the causes and effects of the fast process.

Keywords: dual-processes, impulse control, inhibition, social behavior

Introduction

People appear to have two processes or systems controlling their actions: one fast, unconscious, or automatic and one slow, conscious, and deliberative (Kahneman 2003). Thus far the focus in the literature has been on discussing the differences in the processes in support of developing dual process theories of cognition (Evans 2008).

Evans (2008) provides an excellent review of the dual process theories of reasoning and decision-making. Although researchers use different terms for the two systems, almost all distinguish one system as "unconscious, rapid, automatic, and high capacity" while the other as "conscious, slow, and deliberative" (Evans, 2008). Researchers also differentiate between the systems saying the faster process is implicit and automatic and the slower is explicit and controlled. Many researchers also include the point that the faster process's control of behavior occurs without our being aware of the fact. The faster processing was described as "associative" and the slower process as "rule-based". Another theme reported was that the faster process was more concrete and situation specific and the slower, rational process more abstract and general. The key concept here is the characterization of the two systems by awareness and volition.

Our focus is on building a computational model of the interaction of these processes; specifically, we look at the ability of the slow, conscious process to inhibit the faster, automatic process. Blinking, for example, is one such fast, automatic action that with some effort can be inhibited. Under normal circumstance, blinking is an unconscious process occurring periodically whose rate is influenced by environmental conditions as well as internal, emotional state. But it is also well known that we can resist blinking. However, it is best described as "resisting" because it takes cognitive effort to not blink. The maintenance of our concentration is an example of the slow, cognitive process's inhibition on the blinking behavior. But when the concentration is broken, the fast, unconscious, and automatic process is back in control.

We propose that there are general strategies that humans use to inhibit the undesirable fast processes based on our ability to infer the causes and to detect the effects of those processes. We propose that a learned conscious process can effectively control the execution of the faster process through the control of the focus of attention and the deliberate common-resource management.

With this introduction, we will first discuss how the slow process can perceive the fast process and how the slow process can inhibit the fast process. We will then propose a general model integrating the fast and slow cognitive processes, present three instantiations of that general model, and discuss learning in these models before concluding.

Perception of a Fast Process

As Evans reported, many researchers noted that the faster process occurs without our awareness. Even though we may not be cognitively aware of the faster process while it is in progress, we can note its effect and infer its cause. When physical motion is involved, we have ability to attend to our own movement. In other words, we can sometimes sense the resulting action as soon as after it has been initiated, and definitely sense it after it has been completed. This is subject to the speed and the extent of the response as well as our focus of attention. Furthermore, Gladwell (2005) provided evidence that such fast, unexplainable processes can be the result of deep expertise we cannot easily articulate, but have ability to control including using them to our benefit as well as to inhibit them.

Humans are also capable of inferring a cause of a response. Whether it is attending to an environmental stimulus resulting in a movement, or an association between a memory and our emotional state resulting in an expression change, we can make the association.

For example, consider those nearly thoughtless responses to what we see, such as ducking a fast moving object, to what we hear, such as jumping at an unexpected sound, or even what we feel, such as uttering expletives or grimacing when we stub our toe, or smiling at a pleasant memory.

The ability to detect such effects and to infer the causes of the fast processes allows us to learn strategies to inhibit these fast processes. These general strategies for inhibiting them include (a) blocking the sensory input, (b) blocking (or interrupting) the response, and (c) running an additional process concurrently with the fast process. A general model of interaction of the two processes is shown in Figure 1. The undesirable fast process is represented as a direct Sense-Act thread while the desirable but slow process is shown below as a Sense-*Think*-Act thread. In the figure, the radar circle indicates the extent of changes to the focus of attention and the vertical lines are the boundaries between the cognitive model and the outside world. Attending to our own actions including vocalizations or facial expressions (indicated by the question mark icon in the figure), supports a deliberate choice or development of a control strategy.

Control of a Fast Process

To present how we envision a slow process can control a fast process, we begin by grounding both processes within a cognitive architecture. We will present three implementations of the general model as computational models within the ACT-R cognitive architecture (Anderson, 2007; Anderson et all, 2004). ACT-R is a symbolic and sub-symbolic, production-based cognitive architecture. The internal modules of ACT-R represent relatively specific cognitive functions (and regions of the brain) including declarative and procedural memory, auditory and visual perception, vocalization, and motor functions (based on the hand).

During each cycle, modules representing sensors fill buffers with representations of the environment. Like many production systems, ACT-R repeatedly matches production conditions with the contents of the buffers, but only selects a single production to fire, and then executes that production resulting in changes to internal buffers and module requests.

ACT-R, and more recently, jACT-R (Harrison & Trafton, 2010), have been embodied on a robotic platform which necessitated extension of motor functionality to control face muscles, head and limbs movements. For this project, we also added a rudimentary "emotional module" to allow us to keep track of the internal state of the robot. The emotions are based on appraisals according to the Appraisal Theory (Scherer, 2001; Marinier, et al, 2009), which are provided during the execution of the model. For example, unexpected stimulus is recorded automatically as it is being attended to, but the modeler could also issue an appraisal within a production to signify a successful completion of a goal or a failure. The intensity of the emotion is based on the number and recency of the appraisals along the dimensions indicative of the specific emotion. Unless the emotion is fueled after the initial event, it will decay over time; we modeled the activation of the emotion on the base-level activation equation used in the recall of declarative memory (Anderson, 2004).

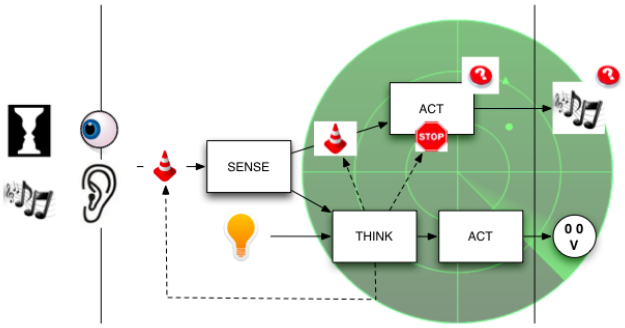


Figure 1. A General Model of Fast and Slow Process Integration.

Our theory of control of the fast process centers on the points at which its execution can be foiled. The alternative strategies leading to inhibition of the fast process are: (1) to block the perception of or attending to the relevant stimulus, and (2) to block the reaction to the stimulus, as indicated by the traffic cones graphic in Figure 1, and (3) running an additional process concurrently with the fast process, as indicated by the light bulb. It is also possible to interrupt or override, to certain degree, actions in progress, such as most large motions including face expressions.

Recall in the discussion of blinking, a slow, cognitive process could inhibit the fast, automatic blinking, but it took cognitive effort. We propose that, in general, it takes sustained cognitive effort to block fast responses. The blocking may not be completely effective in that there is evidence that like interrupting the non-blinking concentration, fleeting micro-expressions of emotion will still occur (Ekman & Friesen, 1969). An extreme example of blocking involves the patellar reflex test (the knee-jerk reaction). A patient can inhibit the normal knee jerk reaction but interrupting the patient's concentration allows the normal reaction to be observed. The common technique to break this concentration is the Jendrassik's Maneuver initially described in 1883 (Zehr & Stein 1999).

We propose that the slow process can both inhibit the faster process through the following alternative strategies:

(1) Intentionally blocking the stimulus by physically removing the stimulus, for example: by closing eyes or covering the ears, or by shifting the perceptual attention.

(2) Intentionally blocking the response by keeping the efferent processor busy, for example: performing another movement or subvocalizing to render the processor unavailable for other processes, or

(3) Intentionally performing another task at the same time.

ACT-R supports this model of process interaction through:

(a) Allowing productions of various specificities.

(b) Buffer status queries including buffer contents and status at various phases of motor processing.

(c) Serialization of processing.

Below is a sample ACT-R production implementing a fast movement in response to an unexpected sound, which could be undesirable in context of many office tasks:

```
(p fast-response-to-sound ;production name
   =aural-location>       ;aural module detects
     isa    audio-event   ;  a sound
   ?aural-location>       ;the sound was
     buffer unrequested   ;  not expected
   ?manual>               ;the motor controller
     state  free          ;  is free, (not busy)
==>                       ;THEN
   +manual>               ;initiate a manual
     isa    press-key     ; action, press
     key    "return"      ; "return" key
)
```

For this production, the strategy to block the sensory input would be any action that would block the detection of an auditory event, such as covering one's ears with one's hands. To block the reaction part of this production, one needs to engage and keep the motor module unavailable because it is busy. Furthermore, due to ACT-R's adherence to serial processing, any other production whose utility is greater than this production would decrease the probability of the undesirable response.

Note that these strategies are temporary and require continuous attention, i.e., cognitive effort, to maintain the strategy. If the cognitive focus is interrupted and the sensory input is still present, the original fast response production will be able to fire.

Model Implementation

We will demonstrate the applicability of the general model by discussing its instantiation in three different models, specifically: (1) inhibiting the Stroop Effect through deliberate shift of visual attention, (2) inhibiting the startle reflex with respect to eye blinking, and (3) inhibiting socially unacceptable response in an emotional situation. Due to space constraints, we will present the model of only one of the alternate control strategies for each of these tasks, but other strategies are applicable as well.

Task: Inhibiting Stroop effect by blocking stimulus

Stroop (1935) identified a large increase in the time taken by participants to complete the color reading in the experiment that presented the participant with incongruent ink color and text, as compared to the naming of the colors of basic shapes. Original experiment has been extended and thoroughly studied over the years to determine in excess of 18 other effects (MacLeod, 1991). In this work we focus on the interpretation of the behavior within the dual processes presented earlier.

Our ACT-R model only captures relative speed difference between the color naming and word reading. Other researchers (Lovett 2002; van Maanen, van Rijn, & Porst, 2008) provide better models of an actual response times in the task, but ability to detect one's errors and to improve the performance at the cost of the response time is a focus of our model's implementation of the dual process theory. When the fast word-reading process generates an incorrect response and it is detected due to a disparity between fast verbal response and the result of the intentional, but slower color naming process. As the response is being vocalized or as it was heard depending on the duration of the color vocalization process, an alternative strategy can be initiated. The easiest strategy simply calls for delaying, or in essence blocking the response, by pausing before giving the verbal response allowing time to reevaluate the color of the text.

As another strategy, Besner (2001) provides evidence that priming a location of a letter within the word eliminates the Stroop Effect. It stands to reason that a good, and in fact optimal, strategy would be for the participant to adjust visual attention accordingly hence blocking the word reading entirely. An easy way to achieve this is to upon or even prior to presentation of the stimulus, to shift attention to the right-most character of the text. With no competing response there is no need to confirm the answer and response can be given immediately.

To block the stimulus in our model, the automatic left-to-right visual search production competes with an intentional visual search production for the right-most symbol from the current location. As long as the expected location is attended to, the word reading (fast process) will not have a chance to happen resulting in a single and correct response.

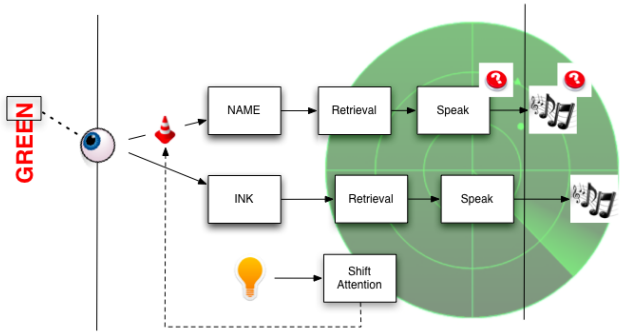


Figure 2. Inhibiting Stroop Effect by shifting gaze.

Task: Inhibiting startle reflex by blocking response

The startle reaction, also startle reflex, is the response to a sudden unexpected stimulus, such as a flash of light, a loud noise, or a quick movement near the face. These reactions include movement away from the stimulus, a contraction of arm and leg muscles, a verbal response, and often blinking. It also includes blood pressure, respiration, and breathing changes that are often described as being startled or scared.

In this section, we focus on the acoustic startle reflex, a response to an unexpected, loud, and near sound on the order of 40ms in duration. Specifically, we present an ACT-R model in which intentionally keeping eyes open inhibits blink-response to the acoustic event. Like other strategies described in this paper, muscle contraction is only a temporary strategy since it requires constant focus to maintain; any lapse in attention will result in muscle relaxing and ability for any process including the startle or routine physical maintenance reflex to control the muscle. Our ability to control blinking is often tested in a staring

contest. Due to the speed of the response, which on average, takes between 300 and 400 milliseconds to complete, this strategy works best when initiated before the stimulus is heard to act to prevent rather than override the reflex or fast response.

Our ACT-R system is capable of perceiving and attending to a sound. The general model strategy to engage the muscle in expectation of the stimulus translates in ACT-R to keeping motor module busy. Assuming the concentration can be maintained and the muscle stays engaged, the fast process's impulse to blink will be blocked. To capture the cognitive effort involved in this strategy, we allow the goal to be removed from focus of attention and the motion to be no longer than 350 ms. The model detects the unintentional motion, based on lack of the intention to move the muscle and presence of the motion.

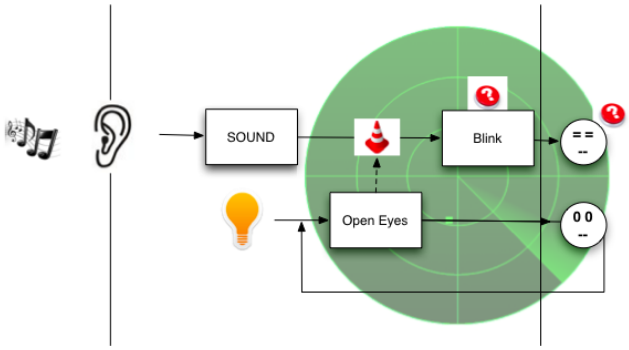


Figure 3. Preventing blinking.

This is definitely not the only strategy that can be used. Interestingly, Fillon, et al. (1993) presented an experiment which showed that an attended pre-pulse, a weaker pre-stimulus, produced greater blink inhibition at the 120 ms lead interval than an ignored pre-pulse. Obviously, covering your ears (or closing your eyes in the case of visual stimulus) is an effortless strategy and guarantees better performance, but is only feasible when task allows for it.

Both of these instantiations of the general model involve blocking the fast process. The next instantiation of the general model develops an acceptable alternative to an emotional response.

Task: Inhibiting emotional response by distraction

Thomas Jefferson is credited with having said "When angry, count to ten before you speak. If very angry, to a hundred," which even nowadays is considered a sound advice since time and distraction are key to anger management. An emotional response is a fast process behavior that rarely leads to positive result, especially in social interactions. However, given time to calm down, most people can get a handle on their initial impulses.

Evans reported that although some researchers ignore emotions in their discussions of the two systems, others place emotions within the faster process and some contemporary work includes an emotional influence in the slower, more deliberative process. Due to this lack of consistency, Evans considered emotions outside the scope of his review of dual systems theory, but we will regard the basic, spontaneous emotional responses as the fast processes.

Ekman identified basic emotions including joy and anger, as being universally recognized from facial expressions (Ekman, 1992; Ekman, 1999). The automatic nature of his basic emotions included specification that the processing was very fast, between 150 and 250 ms. Another researcher, Griffiths (1997), suggested some emotions are higher-level introspective processes, i.e., belonging to the slower, more deliberative process. Others have suggested classifying emotions based on the part of the brain that is activated by the emotion, either the amygdala or prefrontal cortex (Evans, 2001; Frank, 2009). This later differentiation is useful here because although both classifications involve the brain in the response to emotions, the separation of the high-level cognitive function from the low-level processes based on the region of the brain involved, serves our purposes.

While an emotion can be treated as either a stimulus or as a response, for the sake of our argument, we will consider an emotion state as a perceivable stimulus. The emotional responses vary widely and include changes in vocalization characteristics and content, flailing arms or legs, and obviously as facial expressions. For ease of explanation, in the current instantiation of the model, we assume that emotions can be perceived as form of an internal state akin to perception of time (Taatgen, Van Rijn, & Anderson, 2007).

In this instantiation of the general model, we simulate the behavior of an individual that is impatiently waiting for a stimulus to appear (e.g. imagine waiting for a bus or a friend while time is wasting). Since we will be focusing on blocking the undesirable response, the actual stimulus that is cause of the anger is not relevant. Upon stimulus presentation, specifically, the bus or friend's arrival, the subject vocalizes the response based on the emotional state of the model. (See Figure 4.) The model monitors its emotional state as well as the response. A negative reward is associated with the undesirable response (or positive reward is associated with the socially acceptable response).

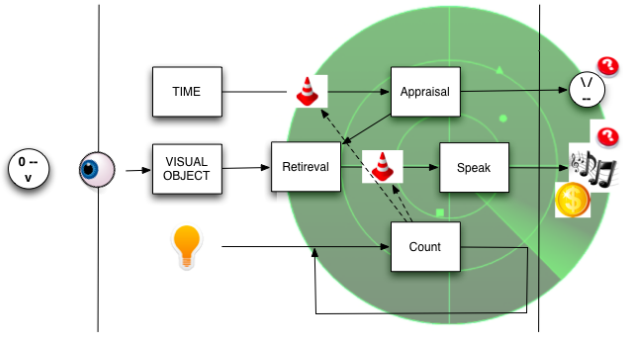


Figure 4. Preventing an emotional response.

As the passage of time is attended to, a negative appraisal is recorded and the model becomes angry. When the stimulus is detected, a fast response process is initiated. At first, the

process does not include the counting to ten and results in a negative, unacceptable response. The counting process triggered by intention to speak while angry, has the property of delaying the response to allow the emotion to decay, and it also distracts the perception of time process from "adding fuel to the fire."

A similar delay tactic can be employed during Stroop task to reinforce the color-naming process. Before giving the answer, the participant could confirm that the response is indicative of the task, which would force the color information of the stimulus to be processed independently. Detecting the conflict is resolved by the conduct (repeat) of a deliberate process to produce the correct answer. Our model of this strategy rewards the response from the deliberate process and may explain the observed brain activity associated with conflict detection and cognitive control (Egner & Hirsch, 2005).

Role of Learning

The feasibility of the strategies discussed in the previous section relies on two forms of learning. First, the alternative, slow process has to be crafted based on the input and output characteristics of the fast process. Second, the model has to learn that the alternative process is useful.

Our general model calls for learning of a control strategy upon detection of an unexpected and undesirable condition. The strategies presented in the task models were hand-crafted. We expect that a problem-solving process focused on addressing the causes of the undesired behavior can develop these strategies. Based on the realization that the causes involve both a stimulus and a response, we expect to be able to learn strategies that involve blocking both the stimulus as in the model of the first task and the response as in the model of the second task. Additionally, introducing a delay or distraction process can be learned if it can be inferred that the causes are time sensitive. This is, of course, subject for future research.

Once the control strategy, i.e., the slow, conscious process, has been crafted, it will eventually become procedurelized and compete with the fast, unconscious process productions. ACT-R utility learning provides the necessary mechanism. In accordance with the ACT-R theory, the utility of a production is determined based on its presence and position in the sequence leading to the reward; specifically, a negative reward issued upon detection of an unexpected and undesirable model behavior leads to relative increase of alternate processes. Since, in the tasks presented here, the fast process is the cause of the unexpected events, this reward mechanism results in the reinforcement of the slower processing path. For example, by punishing the sequence of productions leading up to undesirable response, we lower their utility allowing the counting process to have the higher utility and be included in execution on subsequent runs. Due to this approach, our task models make testable predictions that human error rates in experiments like the Stroop Effect should decrease over time and the response times should be representative of the shift between the two processes.

Essential to both forms of learning is detection of an incorrect or undesirable response. We define an error as an inconsistency between the fast and slow processes' responses indicating a need to decide which is the intended response. Within an ACT-R model, such inconsistencies are described by contents of the relevant buffers. For example, as we have described in the startle reflex task, the detection of a movement when none is expected indicated that a fast, unconscious process was being executed. It should be noted that attending to these cues requires additional processing and given the dynamics of the processing, such cues can be easily missed. Due to this approach, our task models make testable predictions that learning can be part of repeated tests of the Stroop Effect and that learning will not occur if the task dynamics preclude detection and adaptation.

Discussion

In the tasks modeled here, the fast process provided the wrong or undesirable response; this is not true in general. Humans have long depended on these impulses or reflexes to keep us safe as well as to provide the fast responses required in many tasks. Essentially, while slow, rational thinking has its role in our behavior, so does actually allowing the fast, irrational process guide us *in a controlled manner*. We have described how the slow process can control the fast process. However, this is only a beginning.

However, we have not yet presented evidence that our integration of the two processes matches experimental data. Several experiments are suggested by this work including re-visiting the Stroop Effect looking for learned strategies and performance over time.

Conclusions

We have shown that what has been widely discussed as a dual processes, one fast, automatic, and unconscious and the other slow, deliberate, and conscious, can be implemented within a single cognitive architecture and we provided a general model of their integration. We instantiated this general model using the ACT-R architecture and showed the slow process's control of the fast process in three different tasks. The general model's fast-process-control strategies we implemented and demonstrated included: (a) blocking the sensory input for the fast process, (b) blocking (or interrupting) the response from the fast process, and (c) substituting a slow process for the fast process. Finally, we discussed the architectural ability to reinforce the slow process's control of the fast process and an approach to learning the alternate processes.

Acknowledgments

This work is supported by the Center for Social Complexity of George Mason University and by the Office of Naval Research (ONR) under a Multi-disciplinary University Research Initiative (MURI) grant N00014-08-1-0921 and

grants N0001407WX20452 and N0001408WX30007 to J. Gregory Trafton at Naval Research Laboratory. The opinions, findings, and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the sponsors or the institutions.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglas, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of mind. *Psychological Review* 111(4): 1036-1060.

Berger, A. J. (2008). What causes muscle atonia in REM? *Sleep, 31*. 1477-1478.

Besner, D. (2001). "The myth of ballistic processing: Evidence from Stroop's paradigm." *Psychonomic Bulletin & Review, 8(2)*, 324-330.

Chase M. H. & Morales, F. R. (2005). Control of motoneurons during sleep. In M. H. Kryger, T. Roth and W. C. Dement (Eds.) *Principles and practice of sleep medicine*. 4th ed. Philadelphia: WB Saunders. 154-68.

Egner, T. & Hirsch, J. (2005). The neural correlates and functional integration of cognitive control in the Stroop task. *NeuroImage, 24, 539-547.*

Evans, D. (2001). *Emotion: The science of sentiment*. Oxford, U.K.: Oxford University Press.

Evans, J. S. B. T. (2008). Dual-processing accounts of reasoning, judgment and social cognition. *Annual Review of Psychology*, *59*, 255-278.

Ekman, P. (1992). An argument for basic emotions. *Cognition and Emotion*, *6*, 169-200.

Ekman, P. (1999). Basic emotions. In T. Dalgleish and M. Power (Eds.) *Handbook of Cognition and Emotion*. Sussex, U.K.: John Wiley & Sons, Ltd.

Ekman, P. (2003). Darwin, Deception, and Facial Expression. *Annals N. Y. Academy of Science, 1000*, 205-221.

Ekman, P. & Friesen, W. V. (1969). Nonverbal leakage and clues to deception. *Psychiatry, 32,* 88-105.

Frank, M. J., Cohen, M. X., & Sanfey, A. G. (2009). Multiple systems in decision making: A neurocomputational perspective. *Current Directions in Psychological Science*, *18*, 73-77.

Gladwell, M. (2005). *Blink: The power of thinking without thinking.* New York, NY: Little, Brown and Company.

Griffiths, P. E. (1997). *What emotions really are: the* problem of psychological categories. Chicago: The University of Chicago Press.

Harrison, A.M., & Trafton, J.G. (2010). Cognition for action: an architectural account for "grounded interaction" *Proceedings of the Annual Conference on Cognitive Science*. Aug. 11-14, 2010.

Kahneman D. (2003). A perspective on judgment and choice. *American Psychologist, 58*, 697-720.

Lovett, M. C. (2002) Modeling selective attention: Not just another model of Stroop (NJAMOS). *Journal of Cognitive Systems Research, 3*, 67-76.

MacLeod, C. M. (1991). "Half a century of research on the Stroop effect: an integrative review". *Psychological Bulletin, 109 (2)*, 163–203.

Marinier, R., Laird, J. & Lewis, R. (2009). A Computational Unification of Cognitive Behavior and Emotion. *Journal of Cognitive Systems Research, 10, 48-68*.

Salo, R., Henik, A., & Robertson, L. C. (2001) Interpreting Stroop interference: An analysis of differences between task versions. *Neuropsychology, 15(4)*, 462-471.

Scherer, K. (2001). Appraisal considered as a process of multilevel sequential checking. In K. Scherer, A. Schorr, & T. Johnstone (Eds.), Appraisal processes in emotion: Theory, methods, research. New York: Oxford University Press.

Stroop, J. R. (1935). Studies of Interference in Serial Verbal Reactions. *Journal of Experimental Psychology, 18*, 643-662.

Taatgen, N., Van Rijn, H., Anderson, J. R. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention, and learning. *Psychological Review, 114 (3)*, 577-59

Van Maanen, L., Van Rijn, D. H., & Porst, J. P. (2008). Stroop and picture-word interference are two sides of the same coin. *Psychonomic Bulletin & Review, 18*, 987-999.

Zehr, E. P., & Stein, R. B., Interaction of the Jendrassik maneuver with segmental presnaptic inhibitation. *Experimental Brain Research, 124 (4)*, 474-480.

Modeling Visual Search of Displays of Many Objects: The Role of Differential Acuity and Fixation Memory

David Kieras (kieras@umich.edu)
Electrical Engineering & Computer Science Department, University of Michigan
2260 Hayward Street, Ann Arbor, Michigan 48109 USA

Abstract

This paper describes a classic data set on visual search of 100-object displays that differ in size, shape, and color and presents a cognitive architecture model based on the active vision concept that accounts for the effects using differential visual acuity and fixation memory provided by a persistent visual store. The results provide an approximate upper bound on the duration of fixation memory, and some approximate acuity functions for modeling visual search.
Keywords: visual search; cognitive modeling; eye movements.

Introduction

Many everyday and work activities involve visual search, the process of visually scanning or inspecting the environment to locate an object of interest that will then be the target of further activity. An especially tractable form of visual search takes place in many human-computer interaction tasks in which a particular icon coded by color, shape, and other attributes must be located on a screen and then clicked on using a mouse. Such visual search takes place in a visual environment that is much simpler than natural scenes, and so is a both a good theoretical and practical domain to model visual search processes: it combines relative simplicity of the visual characteristics of the searched-for objects with practical relevance: the task is a natural one in the sense that such activities are very common in current technology; an example is current radar displays in military applications, which can contain a large number of icons and other objects (cf. Kieras & Marshall, 2006). Thus understanding in detail how visual search works in such domains can lead to better system designs.

This paper presents a model for the results of a classic study on visual search of large and dense displays of multiple items that can be searched by multiple attributes. This paper follows Kieras (2009), who presented a model for the Peterson et al. (2001) results demonstrating memory for fixations in a visual search task. In the Peterson et al. task, a single object, identified by shape, had to be located in field of a dozen objects which were very small and widely separated, meaning that each object had to be fixated before it could be identified. This paper presents a model for a task at the other extremes: A large number of objects, differing in several attributes had to be searched, but they were large enough and closely spaced enough that the properties of several objects could be considered in a single fixation. Memory for fixations still plays a role, but a critical role is also played by the differential availabilities of visual properties in extra-foveal vision, termed *differential acuity* in what follows.

Visual Search and Active Vision

In a laboratory visual search task, a display of objects is presented, and the participant must locate a particular object specified by perceptual properties and make a response based on whether such an object is present or exactly which properties it has (e.g. the specific shape). In most experiments, the display is static and contains some number of objects, only one of which is the target that must be responded to; the others are distractors. The properties of the display or the displayed objects are manipulated, and reaction time (RT) and/or eye movements are measured.

The empirical literature on visual search was dominated for a long time by studies that measured only RT, and often for tachistoscopically presented displays that ruled out eye movements. But more recently the cost of eye movement data collection has decreased to the point that it has become much more common, and deservedly so. While any behavioral measurement only indirectly reflects the mental processes that produce it, RT is clearly much less diagnostic of what goes on during visual search than eye movements. Furthermore, tasks in which the eye is free to move about a static display in a naturalistic manner, typical of eye movement studies of visual search, will be more representative of the normal operation of the visual system and the role of attention in visual activity. This point was argued eloquently by Findlay & Gilchrist (2003) in presenting an *active vision* framework for understanding visual activity; it is markedly different from traditional approaches to visual attention which have ignored both the role of eye movements and extra-foveal information.

In active vision, a key process is choosing the next object for inspection. A variety of studies (see Findlay & Gilchrist, 2003, for a review) have shown that this choice is not at all random; rather the color, shape, size, orientation, or other visual properties of objects influences which object is chosen for the next fixation; the phenomenon is called *visual guidance*. In the active vision framework, these properties are available in extra-foveal or peripheral vision to some extent, meaning that visual attention, which in the context of normal visual activity is almost synonymous with where the eye is fixated, is a process of selecting for detailed examination one of a large number of objects currently perceived to be in the visual scene, and doing this selection on the basis of the visual properties available in extra-foveal vision.

The availability of a perceptual property in extra-foveal vision depends heavily on the eccentricity (the distance in degrees of visual angle from the center of gaze) of the object, normally referred to in degrees of visual angle, and on the size of the object (also measured in degrees of visual angle), and on the specific property involved. For example,

the color of an object of a given size in the periphery is usually more likely to be visible than its shape.

The EPIC Cognitive Architecture

The EPIC architecture for human cognition and performance directly supports an active vision approach to visual search and provides a general framework for simulating a human interacting with an environment to accomplish a task. Due to lack of space, the reader is referred to Kieras (2004), Kieras & Meyer (1997), Meyer & Kieras (1997) for a more complete description of EPIC.

The EPIC architecture consists of software modules for the simulated task environment, or device, that interacts with a simulated human, which consists of perceptual and motor processor peripherals surrounding a cognitive processor. The device and all of the processors run in parallel with each other. To model human performance of a task, the cognitive processor is programmed with production rules that implement a strategy for performing the task. When the simulation is run, the architecture generates the specific sequence of perceptual, cognitive, and motor events required to perform the task, within the constraints determined by the architecture components and the task environment.

Figure 1 shows the visual system of EPIC. The *eye processor* explicitly represents differential retinal availability in terms of *acuity functions* that specify whether each visual property of each object is currently visible as a function of the size of the object and its eccentricity. The currently available visual properties for each object are represented in the *sensory store*; the *perceptual processor* then encodes the properties of each object, possibly in relation to other objects, and passes the encoded representation on to the *perceptual store* where they are available to the cognitive processor to match the conditions of production rules. The perceptual store thus contains the current representation of the visual world that cognition can reason and make decisions about, including decisions about where to move the eyes next by commanding the *ocular motor processor*. The perceptual store retains the representations for *all objects currently visible*, with more information and detail about those that have been fixated.

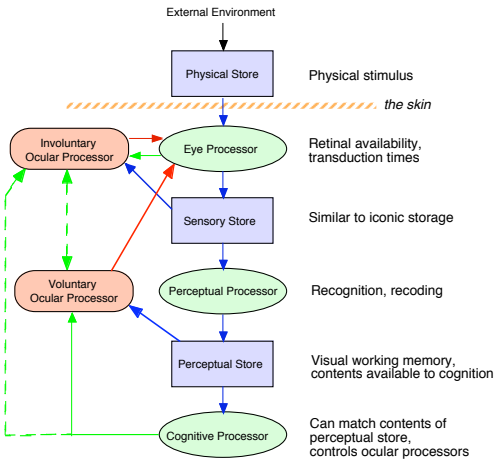


Figure 1. EPIC's visual system.

Persistence of the visual perceptual store

When the eyes move away from an object, the properties of the object persist for a short time (e.g. 200 ms) in the sensory store, and when lost, the perceptual processor notes that the corresponding property in the perceptual store no longer has sensory support. After a relatively long time, the property will then be lost from the perceptual store. But if the object disappears completely, it and all of its properties will be removed from the perceptual store fairly quickly.

The concept is that as the eyes move around the visual scene, a complete and continuous representation of the objects currently present in the visual situation will be built up and maintained in the perceptual store, allowing the cognitive processor to make decisions based on far more than the properties of the currently fixated object. The notion that this information persists for a considerable time as long as the scene is present is supported by studies summarized by Henderson & Castelhano (2005): a naturalistic visual scene is continuously present, but using a gaze-contingent forced-choice paradigm, subjects are tested for their memory of a previously fixated object; retention times at least several seconds long were observed. The model for the Peterson task (Kieras, 2009) provided a good fit to the repeat-fixation data with a retention time of at least 4 sec.

The Williams Study

A classic study using early eye-movement recording methodology was done by Williams (1966, 1967), who ventured into experimental territory commonly avoided even today. This study manipulated the size of the objects along with their color and shape, an unusual combination in the visual search literature, and used a very large number of objects, which provides an upper bound on the difficulty of search tasks of this sort.

The task required visual search of 100 objects varying in size, color, and shape, each with a unique two-digit label. The 100 objects represented all combinations of 4 sizes, 5 colors, and 5 shapes. The search task was to locate the object with the matching label. Depending on the experimental condition, additional attributes of the target object were cued; all combinations of size, color, and shape cues were tested in addition to the *Number-only* cue, which was only the object label. The hypothesis was that if a specification is an effective cue for visual guidance, more fixations should be on objects matching the cue than expected by chance.

The entire display is 39° X 39° (all degrees are degrees of visual angle), and the search objects range from 0.8° to 2.8° in size and distributed at random into the 13 X 13 grid of 3° X 3° cells. The cue specifications were shown in the center of the display. To convey an overall impression of the task, Figure 2 provides an example display produced by the model to be described later. Due to space restrictions this figure is too small for the details to be visible in a paper printing, especially in monochrome, but the details can be seen easily by zooming in with the original pdf file. In this example, the specified target is the medium-size

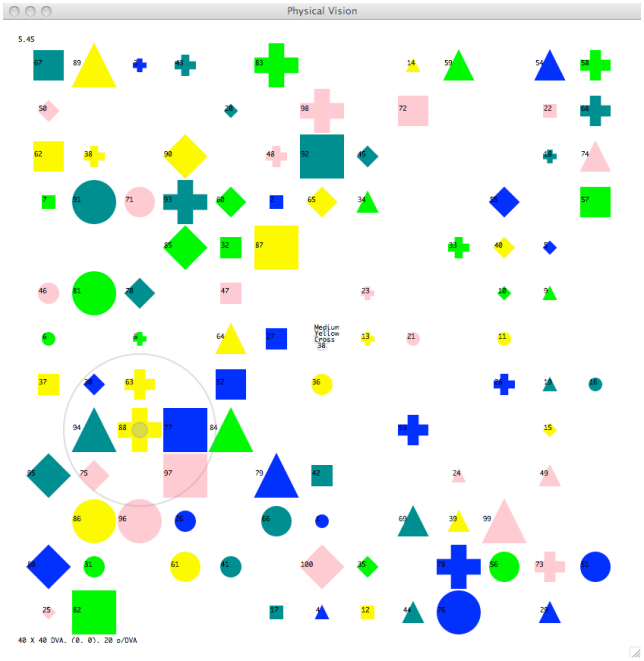


Figure 2. An example of the physical display in a Williams (1966) task trial after several fixations as depicted in EPIC's automatically-generated display. Zoom in on this figure in the pdf file to see the detail.

yellow cross labeled 38, which is in the upper-left of the display. The concentric circles at center left show the current location of EPIC's eyes; the small inner circle has a 1° diameter corresponding to the conventional fovea size; the outer circle is a calibration ring with 10° diameter. The display is shown to scale, except that to maintain legibility, the numeric labels are shown as magnified and left-justified in the object bounding boxes; in the actual stimuli and model representation, they are only 0.3° high, which would require foveation to recognize, and centered in the object.

The specification names for color and shape were the obvious names, but the four sizes were described as *small*, *medium*, *large*, and *very large*. The specifications appeared first in the center of the display; when a button is pressed, the search objects were added to the display. The participant pressed another button when he or she had located the specified object.

Eye movements were recorded with a corneal-reflection film camera system and scored by hand. The total number of fixations were counted, and classified by whether they fell on objects whose size, color, and shape matched the specifications. While 61% of the fixations were attributed to a specific object, 29% were deemed unclassifiable, a relatively large number by current methodological standards.

Unlike modern practice, Williams obtained approximate reaction times (RT) *indirectly* by counting the number of fixations and dividing by 3.25, the observed average number of fixations per second. Because the observed number of fixations and the reported RTs are perfectly correlated, the RTs will only be mentioned occasionally.

The Data

This being an early and basically descriptive study, Williams did not report confidence intervals or information sufficient for their calculation, and conventional statistical tests were not relevant. However, the data set consisted of many thousands of fixations collected from 30 participants who performed 200 trials spread over 8 conditions. Based on the original reports, it appears that a typical sample size for the statistics for any one condition as reported below is in the neighborhood of about 1000. The proportions of fixations on objects of various types are the most important results; for an observed proportion of 0.5, the 95% binomial confidence interval for a sample size of 1000 is about 0.47-0.53; this ±0.03 range can be used as an approximate confidence interval for this important subset of the data.

Figure 3 shows the observed proportion of fixations on objects that matched the cued properties (the predicted values will be discussed below). E.g., if the color was the only specified cue, about 60% of the fixations were on objects with the specified color. Figure 4 shows the number of fixations for each cue type.

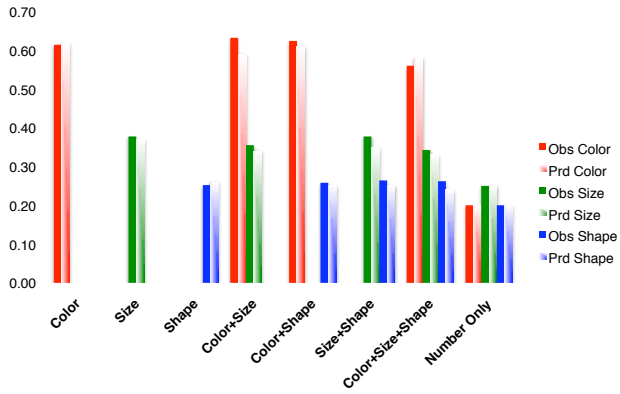


Figure 3. Observed (solid bars) and predicted (shaded bars) proportion of fixations on objects that matched each cue type. The 95% confidence intervals would be roughly ±0.03 for each observed proportion.

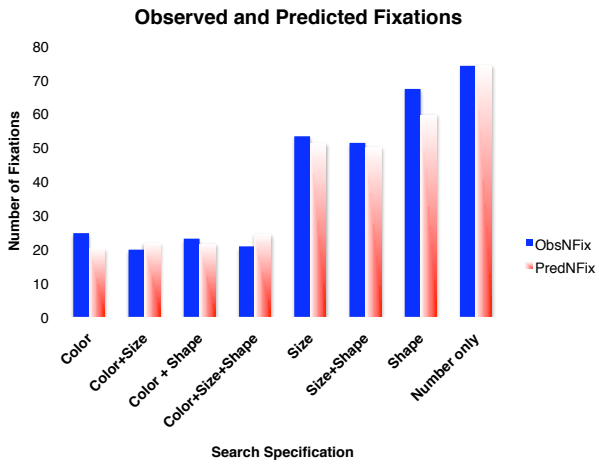


Figure 4. Observed and predicted number of fixations for each cue type.

Visual guidance produced by color, size, and shape

It is clear from the results that color is the strongest cue for visual guidance, resulting in the highest proportion of fixations on matching objects (0.61), the fewest fixations (25) and the fastest RTs (not shown, 7.6 s). Size comes next, and shape is a distant third. There is a tendency for each cue to have little or no effect if a stronger cue is also present. If only the label is provided (the *Number-only* cue), the fixations on objects that match the target properties is at chance level, the number of fixations is large (74), and the RT is quite long (23 s).

The importance of color in visual search is consistent with many results ranging from classic human factors studies (e.g. Sanders & McCormick, 1987) to recent HCI-oriented studies (e.g. Fleetwood & Byrne, 2006). But in the active vision framework, color is not specially privileged in some way, but rather, various direct measurements show that the color of an object is visible over a wide range of eccentricity and object sizes (e.g. Gordon & Abramov, 1977), and so can often serve as an effective cue about where to look next. The relative ineffectiveness of shape is likewise not due to a fundamental problem with shape, but rather that in many cases, recognizing the shape requires resolving detailed features that can only be seen close to the fovea. As an extreme of shape recognition, recognizing the text label involves detecting small features, and so requires foveation unless the text is quite large (Anstis, 1974).

Repeat fixations and memory failures

One overall feature of these results is that many more fixations are required than should be necessary if each object only received one fixation; for example, it should require no more than 50 fixations on average in the Number-only condition to find the labeled object. Williams reports a small number (3%) of immediate repeat fixations, but does not report how many repeat fixations appeared over longer time periods. Apparently objects are frequently looked at repeatedly; e.g. the 74 fixations in the Number-only condition implies a repeat rate of about 33%!

In contrast, recent observation and modeling of repeat fixations (see Peterson et al. 2001, Kieras & Marshall, 2006, Kieras, 2009) suggests that repeat fixations are relatively rare, around 5%, implying a good memory for previous fixations, and almost all are performed immediately, being due to recognition (encoding) failures rather than failures of the memory for previous fixations. The 3% immediate repeat rate reported by Williams is consistent with this, but not the much higher overall repeat rate implied by the total number of fixations.

However, the low-rate results were obtained in search tasks involving many fewer objects and that took much less time than Williams' task. Perhaps the much higher repeat rate in Williams' results is due to time decay of the fixation memory. In fact, in Peterson's task, repeat fixations at long lags become more frequent if the trial has gone on for an unusually long time (Peterson, personal communication). This issue will be important in modeling the Williams data.

Model for the Williams Task

Constructing an EPIC model for the Williams task required a choice of (1) visual acuity parameters, (2) a parameter for the decay time of visual properties in the perceptual store that are no longer sensorily supported, and (3) a set of production rules that implemented the visual search strategy. Each of these will be described briefly.

Acuity functions

Despite the many decades of research on vision, the literature does not contain a comprehensive set of parametric data on acuity for different visual properties as a function of their eccentricity and size, especially for the properties and values typical of computer displays. Space limitations do not allow even a cursory review of the available data (but see Findlay & Gilchrist, 2003). To deal with this non-definitive picture, a simple family of acuity functions were proposed, and their parameters determined by a combination of general constraints set by the literature and iterative maximization of fit in the models. A separate function was specified for each property: encoded size (*small*, *medium*, etc.), color, shape, and text label. The text acuity function was specified as text being available within 1° of the current eye position, corresponding to the conventional definition of foveal vision and the small size of text used. A psychophysical acuity function was used for the other properties: For the property to be available, its size s must exceed a threshold which increases quadratically with eccentricity e and includes a Gaussian noise component X whose variability increases with the object size and coefficient of variation v:

$$threshold = ae^2 + be + c$$
$$P(available) = P(s + X > threshold)$$
$$X \sim N(0, vs)$$

Such a function produces a wide area of highly probable availability, with a sharp tapering-off towards the periphery. The quadratic form was selected for simplicity: the parameters can be easily set to reflect a minimum size, general trend, and degree of curvilinearity, and were set to be consistent with models for other tasks not described here, and to have as much uniformity in the parameter values as possible. The function for color availability used in the model had parameter values of $v=0.7$, $a=0.035$, $b=0.1$, $c=0.1$. The acuity functions for encoded size and shape had the same values except for larger quadratic coefficients a of 0.2 and 0.3 respectively. Thus, consistent with the literature, the availability of the size and shape properties drops off with eccentricity much more rapidly than for color.

The availability for each property at the retinal and sensory store level is independently resampled for all objects whenever the eye is moved. Figure 5 shows an example of EPIC's visual sensory store after several fixations, corresponding to Figure 2, showing what is currently available around the fixation point. In EPIC's display, objects whose location, but no other properties, are known are represented as light gray open circles. Objects which are close enough to the current fixation point to have their color available, but not their shape, are represented as colored open circles. In Figure 5, the shape, color, encoded size, and label are available for the currently fixated object. The colors of several extrafoveal objects are also available, and even the shape for a nearby large object. As the eye moves around, the available properties of the same object

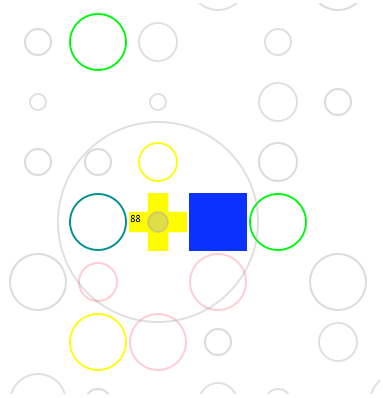


Figure 5. An example of the contents of the sensory store corresponding to the lower left corner of Figure 2, showing available properties of objects near the current fixation point.

can fluctuate, and will not be reliably available from one fixation to the next.

Perceptual store persistence time

Once a property of an object is visible, that property is attached to the object representation in the visual perceptual store where it can serve to match conditions of production rules. The visual perceptual store is persistent, in that as long as an object is within the visual field, its properties, once acquired, will persist for a long time and thus can serve as a memory for previous fixations, as described in Kieras (2009). Figure 6 shows a sample of EPIC's visual perceptual store, corresponding to Figures 2 and 5, several seconds into the visual search, showing the information persisting from previous fixations. Previously fixated objects have all properties including the label, but will eventually lose this information until fixated again. But in the meantime, their color, size, or shape can be used to guide the choice of which object to fixate next.

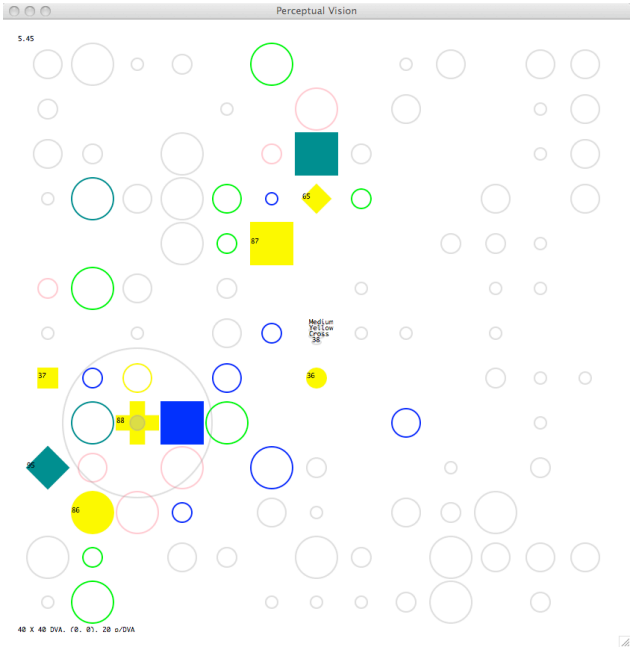


Figure 6. An example of the contents of the perceptual store after several fixations, showing the accumulated object information. Zoom in on this figure in the pdf file to see the detail.

The duration parameter was estimated iteratively by fitting the model, starting with the 4 sec lower bound determined in Kieras (2009); a good fit was found with a duration of 9 seconds.

Task strategy

The visual search strategy in the model is an application of a basic strategy, shown in Figure 8, that has been used in several EPIC visual search models. There are two threads of execution. Nomination rules in the first thread propose objects to fixate based on available visual properties, and also nominate a random choice. Choice rules then pick a single candidate from the nominated objects according to a priority scheme, and launch an eye movement to the chosen candidate. The rules in the second thread wait for all relevant properties of the fixated candidate to be fully visible and either respond if it is a target, or discard the candidate if not. Given the typical 100 ms transduction and encoding times for visual properties and the 50 ms production rule cycle time, the overlapped processing provided by the two threads enables the time between successive eye movement initiations to be short, in the range of 250 to 300 ms, which is commonly observed in high-speed visual search tasks.

For the Williams model, the strategy nominates candidate objects that have the cued properties, such as the cued color or cued shape. The fixation memory effect is implemented by only nominating objects whose text label property is currently unknown; either because the object was never fixated, or it was fixated a long time ago and has been lost from the perceptual store. The priority scheme for choosing a fixation target favors the more available information, and so chooses an object with a matching color over one with a matching size over one with a matching shape. For simplicity, given the apparent very high repeat fixation rates in the data, the mechanism for the relatively rare

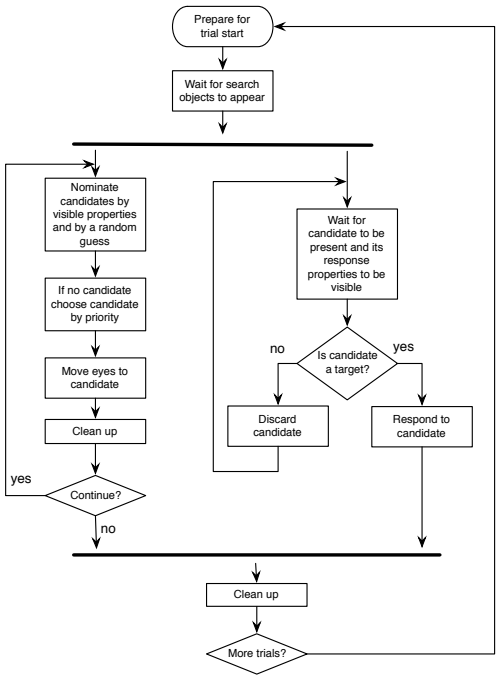


Figure 8. Flowchart for the search task strategy.

encoding failures used in previous models (e.g. Kieras, 2009; Kieras & Marshall, 2006) to trigger repeat fixations was not implemented in this model, corresponding to an assumption that most of the revisits are due to memory failure in this task.

Model Results

The model was run for 500 trials in each experimental condition, and the simulated eye movements and response time data were collected and tabulated analogously to the original experiment. Figure 3 above shows the observed and predicted proportion of fixations of each type. Clearly the fit is very good using the acuity function and perceptual store persistence parameters listed above (R^2 = .99; average absolute error (AAE) = 3%).

The observed and predicted number of fixations is shown in Figure 4 above. Again there is a very good fit (R^2 = 0.98, AAE = 12%). The observed and predicted RTs (not shown) also fit well (R^2 = 0.98 and AAE = 9%), although there is a general tendency for the model RTs to run longer than William's results. Given the unusual methodology used to determine the RTs, it is not clear that attempting to improve the fit to the absolute value would be worthwhile.

In an analysis of the model output, the proportion of repeat fixations was found to increase substantially as the perceptual store duration was decreased, and the number of fixations (or RT) increased. The persistence parameter was adjusted to produce the overall good fit on the number of fixations shown in Figure 4, and the proportion of repeat fixations on search objects was then determined with the final parameter value. The range was 11% repeats in the best condition to 33% in the Number-only condition. This proportion was highly linear with the predicted number of fixations (R^2 = 0.95). This suggests that the loss of fixation memory over time is a good account for the excess number of fixations in the data.

Conclusion

This model, along with the one in Kieras (2009), represents a realization of the active vision concept in terms of a computational cognitive architecture that incorporates differential acuity and a persistent visual store that represents the current visual situation and provides a memory of previous fixations. Two more specific points emerge: (1) Simplistic statements about which properties can guide visual search must be replaced by statements about which properties are available in a specific visual situation. For example, color should not be very effective if the objects were very small, and shape should be more effective if the objects were larger. (2) Repeat fixations have two causes: the persistent visual store is capacious and reliable at short durations, meaning that repeat fixations are due just to encoding errors, but if the search takes a very long time, information from previous fixations is lost, and more repeat fixations are the result.

This general model appears to be ready for practical application in situations where the to-be-searched display contains uniform-color objects with simple geometric shapes and very small distinguishing features such as text labels. The specific acuity functions determined here should be useful approximations in modeling such displays.

At the theoretical level, this type of model appears to be a simple and sound approach to representing visual activity, and is ready to use either as a component in models of more complex tasks that involve visual search as a subtask, or as a basis for models of more advanced visual processing.

Acknowledgment

This work was supported by the Office of Naval Research, under Grant No. N00014-06-1-0034.

References

Anstis, S.M. (1974). A chart demonstrating variations in acuity with retinal position. *Vision research*, 14, 589-592.

Findlay, J.M., & Gilchrist, I.D. (2003). *Active Vision.* Oxford: Oxford University Press.

Fleetwood, M. D. & Byrne, M. D. (2006). Modeling the Visual Search of Displays: A Revised ACT-R Model of Icon Search Based on Eye Tracking Data. *Human Computer Interaction*, 21, 153-197.

Gordon, J., & Abramov, I. (1977). Color vision in the peripheral retina. II. Hue and saturation. *Journal of the Optical Society of America*, 67(2), 202-207.

Henderson, J.M. & Castelhano, M.S. (2005). Eye movements and visual memory for scenes. In G. Underwood (Ed.), *Cognitive processes in eye guidance*. New York: Oxford University Press. 213-235.

Kieras, D.E. (2004). EPIC Architecture Principles of Operation. Web publication available at ftp://www.eecs.umich.edu/people/kieras/EPIC/EPICPrinOp.pdf

Kieras, D. (2009). The persistent visual store as the locus of fixation memory in visual search tasks. In A. Howes, D. Peebles, R. Cooper (Eds.), *9th International Conference on Cognitive Modeling – ICCM2009*, Manchester, UK

Kieras, D.E, & Marshall, S.P. (2006). Visual Availability and Fixation Memory in Modeling Visual Search using the EPIC Architecture. *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, 423-428.

Kieras, D. & Meyer, D.E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12, 391-438.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, **104**, 3-65.

Peterson, M.S., Kramer, A.F., Ranxiao, F.W., Irwin, D.E., & McCarley, J.S. (2001). Visual search has memory. *Psychological Science*, **12**, 287-292).

Sanders, M. S., & McCormick, E. J. (1987). *Human factors in engineering and design* (6th ed.). New York: McGraw-Hill.

Williams, L.G. (1966). A study of visual search using eye movement recordings. Technical Report, Honeywell Inc., Feb. 28, 1966. NTIS AD629624.

Williams, L.G. (1967). The effects of target specification on objects fixated during visual search. In A.F. Sanders (Ed.) *Attention and Performance*, North-Holland. 355-360.

Using Diverse Cognitive Mechanisms for Action Modeling

John E. Laird (laird@umich.edu)
Joseph Z. Xu (jzxu@umich.edu)
Samuel Wintermute (swinterm@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Abstract

Predicting the results of one's own actions is a powerful cognitive capability that can aid in determining which action to take in a given situation. In this paper, we describe a task-independent framework based on the Soar cognitive architecture in which rules, episodic memory, semantic memory, mental imagery, and task decomposition are available for predicting an action's consequences. We include results from two domains and make predictions for human behavior based on these results.

Keywords: Action modeling; prediction; cognitive architecture

Introduction

When faced with a decision between alternative actions, an intelligent agent may have sufficient knowledge to immediately determine which choice is best. However, in situations where directly available knowledge is insufficient or in conflict, an agent can often use predictions of how its actions will change the environment to make its decision. We call the knowledge used to make such a prediction an *action model.* Using this approach to make a decision typically involves the following steps:

1. Choose one of the alternative actions to evaluate.
2. Create an internal representation of the situation.
3. Apply the action model to the internal representation to generate a prediction.
4. Repeat for all other actions.
5. Choose the action that leads to the best predicted state.

This approach to decision making is ubiquitous in humans (de Groot, 1965; Newell & Simon, 1972) and has been used throughout artificial intelligence (AI) systems, where the agent internally simulates multiple steps into the future. A critical ingredient in this process is the action model: the means by which the results of actions are predicted. Action modeling is important because it allows an agent to move beyond reactive behavior – an agent can plan and deliberate about the implications of its actions before choosing one.

Historically, AI systems have used rule-like structures as action models, such as STRIPS operators (Fikes & Nilsson, 1972). Cognitive science research has addressed action modeling, but it has typically been isolated within specific cognitive processes, such as mental imagery (Johnson, 2000; Wintermute & Laird, 2009) or episodic memory (Atance & O'Neill 2005, Schacter & Addis 2007).

Rather than focus on one particular approach to action modeling, we investigate the problem in general. We propose that different combinations of memory and processing systems can be used for action modeling, and that domain characteristics and the agent's knowledge determine which mechanisms are used for a specific task. The mechanisms we propose include rule-based procedural knowledge, episodic knowledge, semantic knowledge, mental imagery, action decomposition, and arbitrary combinations thereof. These mechanisms vary along many dimensions including generality, reportability, learnability, computational expense, and the types of problems where they are appropriate. Forbus & Gentner (1997) have previously posited a similar diversity of processing to support mental models, although they did not focus on detailed architectural mechanisms as we do here.

Included in our work is task-independent knowledge that dynamically combines these mechanisms, implemented within Soar (Laird, 2008). Soar has the requisite representational capabilities to support the diverse forms of memories, processing units and knowledge required for action modeling. In the next section, we give an overview of Soar and our approach to using action models in support of decision making. This is followed by descriptions of the different forms of action modeling, with demonstration of them on a simple blocks world task. We then demonstrate them together on a simple board game, and analyze their relationship to human behavior.

Framework for Action Modeling in Soar

Figure 1 shows the structure of Soar, including its long-term and short-term memories and processing components. Working memory is a shared, symbolic memory that maintains the agent's primary representation of the current situation. Long-term symbolic memories hold procedural, semantic, and episodic knowledge, which are retrieved based on either the total contents of working memory (for

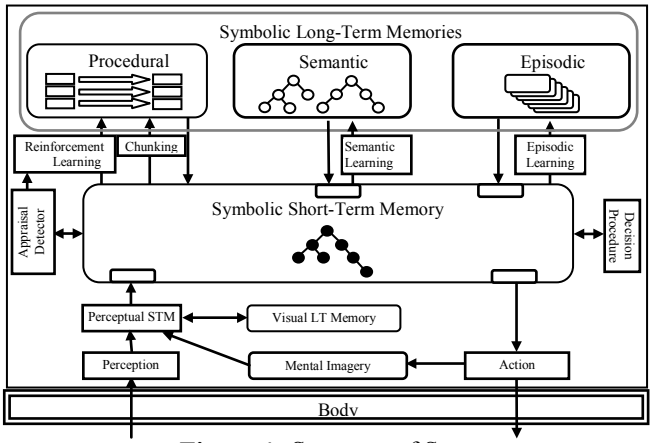


Figure 1: Structure of Soar

procedural) or cue structures created in working memory (for episodic and semantic). Soar has a non-symbolic, spatially-based perceptual short-term memory (STM) from which symbolic information can be extracted into working memory. This memory is the medium of mental imagery.

Behavior in Soar is driven by rules stored in procedural memory. Rules that successfully match the contents of working memory fire in parallel. *Operators* are the locus of sequential behavior in Soar and only a single operator can be selected at a time.[1] Operators are implemented via rules that propose, evaluate, and apply them. Rules that propose and evaluate an operator create *preferences,* while rules that apply an operator modify elements in working memory when that operator is selected.

If there is insufficient knowledge to select or apply an operator, an *impasse* arises, and a substate is created. Within the substate, operators can be proposed, selected, and applied to resolve the impasse. A side effect of resolving an impasse in a substate is that Soar builds a rule that summarizes the processing in the substate. This process is called *chunking*. The learned rule fires in similar situations so that the same impasse is avoided in the future.

Conceptually, operators are either *external*, in that they initiate action in the environment, or *internal*, in that they change the internal state of an agent. Throughout this paper, we call external operators *actions,* so that an *action model* refers to an internal model of the changes that result from the application of an external operator.

Figure 2 shows how action modeling arises in Soar. When an agent is unable to make a decision using its directly available knowledge, it internally simulates the effects of proposed actions to aid in decision making. In this example, the agent is attempting to create a stack of blocks, with A on B, B on C, and C on the table. In the upper left corner of the figure, the agent's state is shown, with the lower half corresponding to a representation of the problem state as it might be in the agent's perceptual short-term memory. The top half of the state shows the symbolic relations that the agent extracts from perception, and it is these relations that

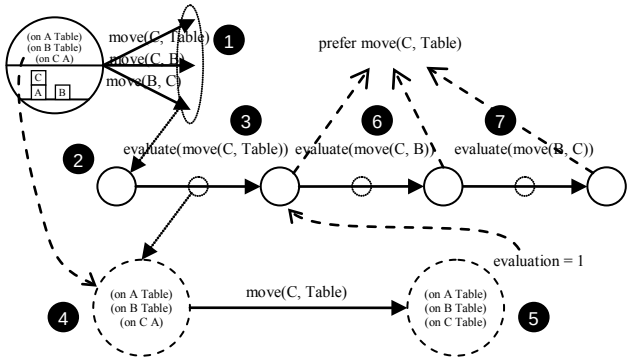


Figure 2: Soar processing using an action model.

[1] Operators in Soar correspond most closely to rules in ACT-R (Anderson, 2007); however, operators in Soar provide a richer representation for organizing action than do rules in ACT-R because of the independent representations of knowledge (as rules) for proposing, selecting, and executing the actions associated with an operator.

are available in working memory.

We assume the agent has sufficient knowledge to propose the three legal actions for this state: move B onto C, move C onto B, and move C onto the table. However, there are no rules to create preferences, so an impasse arises (1), and Soar automatically creates a substate (2).

To resolve this impasse, the agent tries out each proposed action on a copy of the state and then evaluates the quality of the result. Task-independent knowledge (TIK), encoded as rules, carries out this strategy. The *only* additional task-dependent knowledge required in this processing are action models and state evaluations, both of which can use the various forms of knowledge presented below.

As shown in Figure 2, following the impasse, operators are selected (at random) to evaluate the actions. In the example, move C to the table is evaluated first (3). In this case, the agent does not have rules to evaluate this action directly, and thus, another impasse arises. In the resulting substate (4), the TIK copies the contents of the original task state and uses a model of the action being evaluated to predict the resulting state. Once this state is computed (5), the agent must also have some knowledge (usually encoded as rules) for evaluating it. In this case, we use an evaluation that counts the number of blocks in their desired positions, which assigns the state an evaluation of 1. The creation of this evaluation terminates the evaluate operator, which is followed by the selection of operators to evaluate the remaining actions (6, 7). When all the evaluations are computed, preferences are created for the actions, leading to the selection of the action to move C to the table, and resolving the first impasse. The action is then performed. Chunking learns rules for evaluating each of the actions (from the substates where the action modeling occurs), and for creating the preferences based on those evaluations.

Different Forms of Action Modeling

In this section, we describe how action modeling can be implemented using different processing and memory systems, with the blocks world serving as an example.

Procedural Knowledge

The most direct way to encode an action model in Soar is as rules. These rules test features of the state, features of the selected action, and that the state is an internal copy of the task state. They modify the internal copy in the same way the external action would modify the real state. For complex actions, the model can be implemented with multiple rules that fire in parallel and/or in sequence.

Episodic Memory

Soar has an episodic memory that automatically stores "snapshots" of working memory over time (Nuxoll & Laird, 2007). Soar's episodic memory is an idealization of human episodic memory, and emphasizes basic functionality, such as efficient storage and associative retrieval of temporally organized episodes. For action modeling, episodic memory requires that the agent has a previous experience when the action being considered was applied in the environment.

The agent can then use its memory of that experience to make a prediction as to what will happen when the operator is applied to a similar situation (Xu & Laird, 2010).

When episodic memory is used, the behavior of the agent is as follows. The first time the agent gets to the point where the action is selected in Figure 2, an impasse would arise because there is no rule to apply the action. In the resulting substate (not shown in Figure 2), the TIK for using episodic memory selects an operator which creates a cue consisting of the task state with the action selected, in an attempt to retrieve a similar previous episode. Once the cue is created, the episodic memory system retrieves the most recent, best match to the cue and reconstructs it in working memory. If no match is found, then this approach to action modeling fails, and the agent must either try other methods, or assign a default evaluation value to the action being evaluated. Chunking does not create rules to summarize processing in substates where episodic memory retrieval failed.

If the retrieval is successful, the agent then retrieves the following episode. The agent continues retrieving subsequent episodes until it finds one where the action is no longer selected, which indicates the action has terminated. The agent then compares the task state in that episode to the current task state and modifies the internal copy of the task state to reflect any changes. Chunking creates a rule that summarizes the processing, so that in the future, the retrievals are not required.

Figures 3 and 4 compare results for using the rule-based versus the episode-based approaches to action modeling. Both figures show the progression of performance across four identical trials of the blocks world problem described above, and both use log scales for the y-axis. Figure 3 shows the number of external actions that the agent takes to solve the problem, while Figure 4 shows the number of decisions (processing cycles in Soar). These results are not intended to precisely model human behavior (for example, we are not including time for perception or motor actions); however the comparisons should be meaningful in predicting qualitative differences across methods and trials.

In Figure 3, the top line shows the average performance of an agent using episode-based action modeling where episodes are not learned, so that a random selection is always made. The next line shows the performance when episodes are being learned. Initially there are no relevant episodes, so the selections are random, but with experience, the episodes accumulate and the agent's performance improves as it is able to correctly predict future states and select the correct action, until finally it achieves optimal performance. Even the first trial gets some improvement from learned episodes. The bottom line shows the performance with the rule-based action model, which always makes the correct predictions.

Figure 4 shows the performance in terms of decisions, not just external actions. The top line corresponds to the steps required when episodes are not learned. The next line shows the performance as episodes are learned. The dashed line that starts at the same point for trial 1 shows that when chunking is used with episodic memory, it eliminates the need for episodic retrievals over time as the agent learns action models based on rules that replace those based on episodic memory. The agent eventually learns rules that choose actions directly, eliminating the need for action models. Thus, there is a combined gain with episodic memory improving solution quality, and chunking improving the efficiency of the problem solving process. Note that external actions take orders of magnitude more time to execute than internal reasoning steps, so the differences are more pronounced in real environments.

The next line shows the performance for the rule-based action model without chunking, which serves as the optimal base line for action modeling. The final line shows the impact of using chunking with the rule-based action model, where after one trial, rules are learned that eliminate the need for the action model. As these figures show, in only a few trials, the combination of episodic memory and chunking converts an agent with little task knowledge into one that solves the problem in few actions (due to episodic memory-based action modeling), while eliminating the need for purely internal decisions (due to chunking).

Semantic Knowledge

Whereas episodic memory is based on specific experiences, semantic memory consists of decontextualized facts – such as knowledge about objects and their structure, independent of when they were experienced. This makes semantic knowledge more difficult to learn than episodic knowledge,

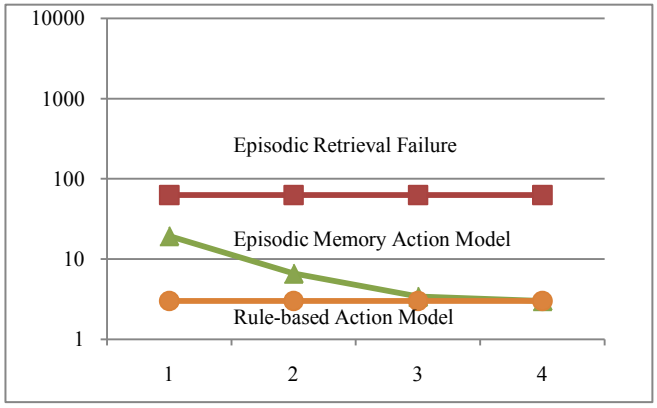


Figure 3: External actions taken across multiple trials.

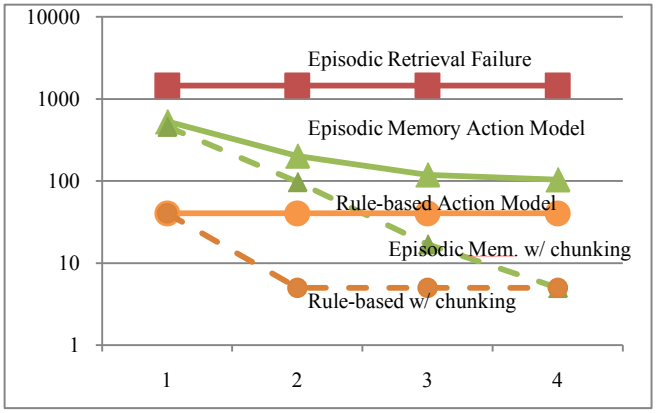


Figure 4: Total decisions taken across multiple trials.

but more useful across a variety of tasks. Soar as yet does not have a theory of how semantic memories are automatically learned, and instead Soar agents must deliberately store semantic data they encounter.

The use of semantic memory for action modeling is analogous to the use of episodic memory – when there is no action model encoded as rules, an impasse arises, and in the resulting substate, an operator is selected which queries semantic memory to retrieve knowledge that can aid in predicting the result of applying that action. Semantic memory covers a broad range of knowledge, and one can imagine many ways it can aid in action modeling. For example, the fact boiling kettles are hot can be useful when predicting the consequence of touching one. Here, we use declarative instructions that specify how to modify the internal task state to model the action.

To use semantic memory, the agent selects an internal operator that initiates a retrieval for instructions related to the action being evaluated. If the relevant instructions are retrieved, TIK selects the "interpret" operator, whose purpose is to apply the instructions to the copy of the task state. The interpret operator is not implemented directly in rules, but leads to a substate where operators are selected and applied for each of the instructions. The processing in the substate allows for arbitrarily complex implementations of instructions, and is similar in spirit to how declarative instructions are used in ACT-R (Anderson 2007; Best & Lebiere 2003); however, in those cases the instructions are interpreted to control the execution of a task, while here they are used to model the execution of an action.

The format of declarative instructions is like that of an imperative programming language or a recipe. We have developed task-independent declarative representations for common control flow instructions and state modifications. In the blocks world example, instructions specify additions and deletions of predicates. The rules that interpret those instructions assume a specific representation of predicates in working memory. Figure 5 shows the instructions for moving a block. When using semantic memory, the number of decisions decreases after one trial, as chunking creates action model and action selection rules.

Mental Imagery

Mental imagery involves the maintenance of a separate memory structure grounded in perception, which represents objects and their spatial properties. While the contents of the memory is mostly created bottom-up from perception, an agent can create new "imagined" structures and manipulate them by operations such as translation, rotation, and scaling, as well as simulate complex motions, such as the path of a car (Wintermute, 2009). The agent can extract spatial predicates from perceptual memory, such as the relative positions of objects and whether they collide. When applied to perceived structures, this can be used to create the initial symbolic representation of the problem. When applied to imagined structures, symbolic consequences of actions can be predicted. The use of mental imagery for action modeling is restricted to actions that involve spatial motion, or actions that can be mapped onto such motion.

```
Move-block(blk, dest):
  1. Del-predicate ontop(blk, x) ∀ x ≠ dest
  2. Add-predicate ontop(blk, dest)
```

Figure 5: Instructions encoded in semantic memory.

As in the use of episodic and semantic memory, mental imagery is employed when there are no rules for an action model, and an impasse arises. Mental imagery takes advantage of the spatial representation and maps the action to be modeled onto imagery operations. Making the connection between the action and mental imagery operations can involve accessing knowledge in semantic memory, or such knowledge can be encoded in rules. In our example, the agent knows that to move a block, it should imagine it centered on top of the destination block. Once the perceptual memory has changed, relevant predicates can be extracted, creating a symbolic description of the situation that serves as the resulting state.

Mental imagery involves processing that cannot be analyzed by chunking because the results of the processing are not uniquely determined by the symbolic structures available in working memory. Therefore, chunking does not create rules that summarize mental imagery processing. This is similar to ACT-R avoiding rule compilation for processing over external interactions (Anderson, 2007).

Although not as general as the other methods, mental imagery has wide applicability because of the ubiquity of spatial problems. Imagery-based action models are effective in a range of problems, from simple tasks in the blocks world (Wintermute & Laird, 2009) to complex tasks such as path planning for cars (Wintermute, 2009).

Action Decomposition

The final alternative approach is to model an action by decomposing it into simpler actions that can be modeled using any of the approaches described above. In Soar, hierarchical operator decomposition is ubiquitous, and arises when complex operators are selected, and then implemented in substates by simpler operators. In the blocks world example, when move-block is selected, it can be decomposed into pickup-block and put-down-block actions. When these actions are selected, any of the previous methods can be used as models for them, including further decomposition. One typical use of action decomposition is to take an action that involves complex spatial interactions and decompose it into simpler parts until those parts can be mapped onto imagery operations. Chunking will create rules for the action model of a complex operator as long as mental imagery was not used in any substate processing.

A Policy for Controlling Action Modeling Approaches

We have presented these action modeling approaches as alternatives, with no attention to when each would be used in an integrated agent. Inherent to Soar is that it uses rules for action modeling if they are available. That is the default behavior and it is not under control of the agent. When rules are not available, an impasse arises, and in the ensuing

substate, operators are proposed for the alternative methods, as well as any operators that decompose the selected action. This structure introduces an extra level of deliberation, which adds flexibility at minimal cost to the agent (the results in Figure 4 are without this additional layer). Although it may be possible for an agent to learn when best to use each method, that could be a difficult learning problem and we leave it to future research. As an alternative, we encoded a simple ordering preference for these approaches in the TIK and use this method in the board game demonstration below.

Integrated Demonstration

To provide additional illustration of how these approaches work, both independently and in unison, we present an agent that plays a simple board game, shown in Figure 5. In this game, the agent must slide the hexagonal marker on the left along the directional paths to numbered nodes until it gets to the end (node 10). As the marker slides along a path, it may touch one of three different objects, labeled X, Y, and $. If the marker hits an object, the agent gets points. The agent has semantic knowledge that the $ is worth 20 points, but does not initially know the values of the other objects (X is worth 10 points and Y is worth 5). The goal is to get to the end with the highest possible score, which is achieved via path A, C, F, H, I, K. We assume that the agent can sense the marker position, the paths, and the objects, but it does not a priori know whether the marker will hit a nearby object as it slides along a path.

To perform the task, the marker starts at position 1, and the agent is faced with making a decision to take path A or B. To make this decision, the agent will attempt to predict the result of each move. At this point, the agent does not have any action model rules, nor does it have any episodes or relevant information in semantic memory. However, it can use mental imagery to imagine moving the marker along each of the paths. Mental imagery predicts that if it moves along A, it will intersect with object X, while for B, it will intersect with Y. In both cases, it does not know how encountering those objects will affect its score, so it chooses at random. We assume it picks path B. It executes that action, encountering Y and getting 5 points.

Once at 3, the agent picks path D to get to 4. Here, the decision is between going along path E or F. This time, after it uses mental imagery to detect that it will encounter object Y, it then uses episodic memory to recall that the last time it encountered object Y it received 5 points. When it considers path F, it uses imagery to predict it will encounter object $, and then semantic memory to predict that it will receive 20 points. Based on these evaluations, it chooses path F. It receives 20 points, moves to 6 and then 7. At this point, it uses a combination of mental imagery and episodic memory to predict the result of moving to 8 (10 points). In imagining moving to 9, imagery shows that it will not encounter Y, so it will get a score of 0. It selects moving to 8, and then finishing by moving to 10, getting a total score of 35.

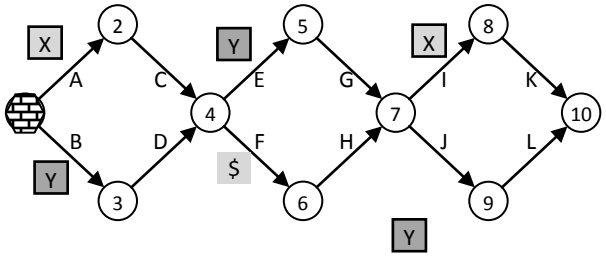


Figure 6: Board game task performed by agent.

The next time the agent plays the game, it uses episodic memory to predict the results of the paths it took the first time (B, F, I). Since it has no episodic memories of moving on paths A, E, and J, and cannot chunk over imagery action models, it must continue to use imagery for those paths.[2] Thus, in its second attempt, it will use imagery and episodic memory to predict a 10 score for A, while it will use only episodic to predict a score of 5 for B. Similar use of imagery and episodic memory will be used at nodes 4 and 7. As a result, the optimal path is taken, resulting in a score of 40.

Figure 7 shows the progression of how the agent's decisions are distributed across using imagery versus episodic memory over multiple trials. The highest line shows the total number of internal reasoning steps. The bottom two lines are the number of decisions that involve imagery and episodic memory operations. In the first trial, imagery dominates as the agent has no prior experiences it can draw on. In the second run, the agent must still use imagery for those cases where it has not taken a path, but it uses episodic memory for those cases where it had prior experiences. Although not evident in the graph, chunking replaces the use of semantic memory with a rule. For the third run, chunking decreases the total number of steps by eliminating the use of episodic memory. In the final trial, some imagery is still required for those paths the agent never actually tried, and episodic memory is no longer used as it has been replaced by rules learned through chunking.

Predictions

From these examples and an understanding of the approach, we can make some predictions about the behavior

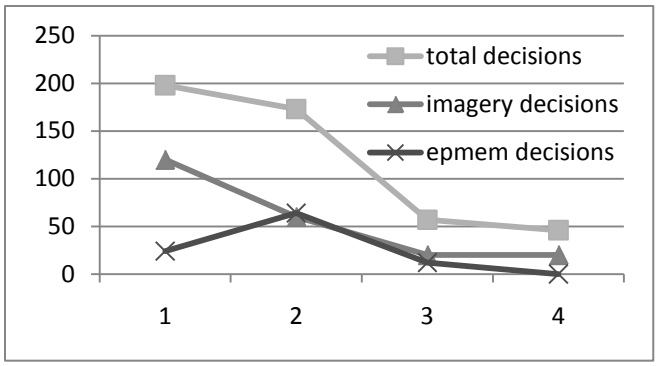


Figure 7: Agent performance over multiple trials.

[2] Soar's episodic memory does not capture subgoal processing, so the agent has no episodic memories of previous predictions. Otherwise, these steps could also be removed.

of an agent with the capabilities we described.

In a spatial environment, an agent initially relies on mental imagery for action modeling (and semantic knowledge if it is available). As the agent gains experience, it switches to using episodic memory when possible. With further experience, rules learned via chunking replace episodic memory, and eventually rules are learned that choose actions directly, eliminating action modeling.

Concurrent with learning, the agent's ability to report on its internal reasoning should change, as different structures become available in working memory (which is the basis for our predictions about reporting). Initially, for spatial problems, the agent can report imagining spatial situations, which then transitions to reports of using episodic memory (things it "remembers"). When using semantic memory, it can report on the instructions and facts it is using (things it "knows"). With practice, the agent loses the ability to report on its reasoning as intermediate structures are no longer generated in working memory and processing is done purely with rules. The rules produce behavior without the creation of a declarative trace that the agent can report.

As shown in Figure 7, our model predicts there are also changes over time in terms of which mechanisms are used in action modeling, and thus decision making. The obvious prediction is that in humans the brain areas used for action modeling, and thus decision making, will change based on characteristics of the task (whether it is spatial or symbolic) and a subject's experience (whether it has access to relevant semantic, episodic, or procedural knowledge).

Conclusions

The major claim of this paper is that intelligent agents, including humans, have a variety of available mechanisms that can be used to predict the results of their actions in service of decision making. A related claim is that internal prediction does not occur in any specific architectural module, but results from a combination of characteristics of the domain, the agent's background knowledge, prior experience, and the agent's available memories and processing elements. We have demonstrated two agents in two domains using rules, episodic memory, semantic memory, mental imagery, and action decomposition for action modeling. Although the domains are simple, the results predict significant changes in behavior as knowledge accumulates in episodic memory and is compiled into rules.

Central to achieving these results are the various memories and processing units in Soar as presented in Figure 1, as well as the task-independent knowledge that controls the use of these knowledge sources. A critical component of Soar's ability to support these methods is its employment of impasses when knowledge is incomplete. Impasses are critical for identifying when action modeling is necessary (a tie among competing actions) and for invoking alternative approaches when rule-based action modeling knowledge is missing. In addition, substates provide the representational structure needed to support retrieving and combining knowledge without disrupting the state of the problem being attempted. These components appear to be missing, or at least difficult to achieve, in other architectures, and it would be informative to attempt to duplicate the qualitative structure achieved here in other cognitive architectures.

Acknowledgment

The authors acknowledge the funding support of the Office of Naval Research under grant number N00014-08-1-0099.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Atance, C. M., and O'Neill, D. K. (2005). The emergence of episodic future thinking in humans. *Learning and Motivation* 36(2): 126-144.

Best, B. J. and Lebiere, C. (2003). Teamwork, Communication, and Planning in ACT-R Agents Engaging in Urban Combat in Virtual Environments, *International Joint Conference on Artificial Intelligence*.

de Groot, A. D. (1965). *Thought and choice in chess*. The Hague: Mouton Publishers.

Fikes, R., and Nilsson, N. (1971). STRIPS: A new approach in the application of theorem proving to problem solving. *Artificial Intelligence* 2, 189-208.

Forbus, K. and Gentner, D. (1997). Qualitative mental models: Simulations or memories? *Proceedings of the Eleventh International Workshop on Qualitative Reasoning.* Cortona, Italy.

Johnson, S. H. (2000). Thinking ahead: the case for motor imagery in prospective judgements of prehension. *Cognition*, *74*(1), 33-70.

Laird, J. E. (2008). Extending the Soar Cognitive Architecture. *Proceedings of the First Conference on Artificial General Intelligence*.

Newell, A., and Simon, H. A. (1972*). Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Nuxoll, A. M. and Laird, J. E. (2007). Extending Cognitive Architecture with Episodic Memory. *Proceedings of the 22nd National Conference on Artificial Intelligence.*

Schacter, D. L, and Addis, D. R. (2007). The cognitive neuroscience of constructive memory: remembering the past and imagining the future. *Philosophical Transactions of the Royal Society of London*. Series B, Biological Sciences 362, no. 1481 (May 29): 773-786.

Wintermute, S. (2009). Integrating Reasoning and Action through Simulation. *Proceedings of the Second Conference on Artificial General Intelligence*.

Wintermute, S., and Laird, J. E. (2009). Imagery as Compensation for an Imperfect Abstract Problem Representation*. Proceedings of the 31st Annual Conference of the Cognitive Science Society*.

Xu, J. Z. & Laird, J. E. (2010). Instance-Based Online Learning of Deterministic Relational Action Models, *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence.*

Using A* Graph Traversal to Model Conflict Resolution in Air Traffic Control

Stefan Lehmann (Stefan.Lehmann@nicta.com.au)
National ICT Australia, Queensland Research Laboratory, Level 1, McElwain Building (24A), The University of Queensland
Brisbane QLD 4072 Australia

Scott Bolland (scottb@itee.uq.edu.au)
School of ITEE, General Purpose South Building (78), The University of Queensland
Brisbane QLD 4072 Australia

Roger Remington (r.remington@psy.uq.edu.au)
School of Psychology and NICTA QRL, McElwain Building (24A), The University of Queensland
Brisbane QLD 4072 Australia

Michael S. Humphreys (mh@psy.uq.edu.au)
School of Psychology and NICTA QRL, McElwain Building (24A), The University of Queensland
Brisbane QLD 4072 Australia

Andrew Neal (Andrew@psy.uq.edu.au)
School of Psychology and NICTA QRL, McElwain Building (24A), The University of Queensland
Brisbane QLD 4072 Australia

Abstract

The efficient detection and resolution of conflicts represent the key tasks of Air Traffic Controllers in enroute environments. The complexity of these tasks imposes significant challenges on the design of cognitive models that are capable of adequately simulating them. Yet, the availability of such models is crucial for a number of applications, including the evaluation of current and future Air Traffic Control concepts. In this paper, we will propose a novel modeling approach which adopts the principles of the A* graph search scheme from Artificial Intelligence to represent the cognitive decision making process of the human operator. Results of an initial version of this model will be presented, showing that the proposed approach has promise.

Keywords: Cognitive Modeling; Cognitive Systems Engineering; Artifical Intelligence; Decision Making; Air Traffic Control.

Introduction

In most western economies, the volume of air traffic is currently growing at a rate of 4 to 6 percent per annum. According to its 2006 annual report, the US Federal Aviation Administration (FAA) acknowledges that air traffic controllers will not be able to handle traffic at 25 percent above today's level, and that traffic may increase this much by 2016 (ICAO, 2004). In response to this problem, the United States Federal Aviation Administration and Eurocontrol are currently pursuing programs to greatly increase airspace capacity (FAA, 2010; Eurocontrol, 2008), without raising either the workload or number of air traffic controllers.

Cognitive modeling could provide an important vehicle for the evaluation of new operational concepts if it is possible to simulate performance on challenging air traffic control operations. For example, models making reasonable estimates of sector workload could inform evaluations of safety and staffing. One of the more cognitively complex tasks of controllers is the detection and resolution of conflicts (Lehmann, Bolland, Remington, Humphreys, Fothergill, Hasenbosch, & Neal, 2010). The *n*-aircraft conflict resolution problem is highly combinatorial and cannot be optimally solved using classical mathematical optimization techniques. This inherent complexity imposes significant challenges on the design of corresponding models.

This paper will propose a new method that simplifies the task of modeling expert decision making in Air Traffic Control (ATC) environments by relying on domain-specific simple heuristics that humans deploy to produce accurate decisions (Todd & Gigerenzer, 2007). The conflict resolution mechanism adopts the principles of the A* search algorithm (Felner, Stern, Ben-Yair, Kraus, & Netanyahu, 2004; Lee, Osman, & Sabudin, 2009; Leigh, Louis, & Miles, 2007). The resulting scheme implements a search through a space of conflict solutions. System states are evaluated using optimization criteria encapsulating the controller's goals. Each optimization criterion is associated with a number of individual cost functions that penalize deviations of the system states from the goal states. The focus on psychologically plausible strategies, rather than representative psychological processing mechanisms, was in part a response to the complexity of decision making in ATC and the large number of unobservable factors that would need to be incorporated (*e.g.*, memories for previous or typical solutions). Moreover, the strategies we use were elicited from highly experienced controllers and thus encapsulate experts' insights and knowledge. Our working hypothesis is that the use of psychologically plausible

solution heuristics and optimization criteria in conjunction with the constraints imposed by the environment will produce human like behavior.

We first describe the conflict detection mechanism, then detail the manner in which the model selects solutions using the optimization criteria to find a path in the search tree. Finally, we present empirical tests of an initial implementation of the model showing good but not perfect fits to data from human controllers.

Conflict Detection Scheme

The current implementation of the conflict detection scheme is based on the model proposed in Loft et al. (2009). It detects pairs of conflicting aircraft in a hierarchical fashion. Its decomposition into three operational stages allows for a run-time efficient implementation. Potential conflicts are verified by extrapolating the flight paths of all aircraft that are present in the given scenario, and by subsequently identifying violations of separation standards between the flight paths. Positional aircraft uncertainty is accounted for in this process. The three stages proceed as follows:

Stage 1: Coarse check of vertical separation

A coarse check is performed to verify the vertical separation between aircraft. This stage checks if the vertical corridors of any two aircraft of interest are separated by more than 1000 ft, where the vertical corridors are defined by the aircraft's target altitude and cleared altitude respectively.

Stage 2: Lateral separation check

If the first stage (coarse check) reveals the existence of a possible vertical conflict between two aircraft, the model deploys the so-called *Trajectory Modeller* to check for a lateral conflict. At any given time *t,* the Trajectory Modeller extrapolates the flight paths up to time *t + 10 min* in discrete $\Delta T = 5\ sec$ steps. The aircraft positions at each time step are subject to positional uncertainty, where the uncertainty increases successively over time based on a step function. More specifically, the extrapolated aircraft position at a discrete time step $t_k = k\Delta T,\ k=0, 1, 2, 3,\ldots$ is associated with a discrete uncertainty interval $[a_k\Delta T,\ b_k\Delta T]$, where the coefficients a_k and b_k associated with the lower and upper limits of the interval are:

$$a_k = \text{trunc}\,(0.98 \cdot k) \qquad \text{Equation 1}$$

$$b_k = \text{trunc}\,(1.02 \cdot [k+1]) \qquad \text{Equation 2}$$

Stage 3: Final vertical separation check

If the second stage (lateral separation check) verifies a potential lateral conflict between two aircraft of interest, a third stage will be deployed to check for vertical conflicts. For this purpose, the respective flight paths are vertically extrapolated based on the maximum and minimum climb or descent rates of the aircraft. Response times of the aircraft are currently not considered. That is, the aircraft are assumed to instantaneously initiate the actions associated with the controller's interventions.

Decision Making Model

The proposed decision making model adopts the principles of the A* graph search algorithm (Felner, Stern, Ben-Yair, Kraus, & Netanyahu, 2004; Lee, Osman, & Sabudin, 2009; Leigh, Louis, & Miles, 2007). This algorithm relies on a state-space search engine to evaluate the decision alternatives in a hierarchical fashion. Hierarchical search has been shown to produce good modeling solutions to complex aeronautical problems in the past (Nason & Laird, 2005; Rosbe, Chong & Kieras, 2001).

A* finds the minimum cost path in a decision tree through a partial search in the solution space. The avoidance of an exhaustive search presents a significant advantage for its application in the ATC domain, where the decision making process poses a complex problem that typically leads to an extensive search tree in general traffic scenarios. That is, the topology of the search structure does not need to be known *a-priori.* In our model, the search space consists of *solution types*, each representing an action that could be taken to resolve the conflict. The solution types are based on simple heuristics that have been obtained from experts (using interviews and controlled experiments), and from data mining (using radar track data).

Solution Types

The current implementation of the conflict resolution model provides a set of three different solution types which may be applied to the aircraft involved in potential conflicts. Before a solution can be considered for exploration, one or more conditions of applicability must be satisfied. Each solution has a particular weight. A smaller weight corresponds to a more favourable solution. The effective weight of a solution is the sum of a base weight and a penalty value. The purpose of the penalty values is to impede the selection of solutions that would severely disturb an aircraft's intended flight path. The individual solution types and their weights are:

A. Assign closest level below or above conflict zone

The principle of this solution type is to ensure sufficient vertical separation by assigning one of the two aircraft of the conflict pair a safe altitude either beneath (*low solution*) or above (*high solution*) the other aircraft whilst they are in the region of the airspace where a loss of lateral separation is possible. More specifically, assuming two conflicting aircraft *A* and *B*, the low solution is applicable if A is not already descending through the low solution. Alternatively, the high solution is applicable if *A* is not already climbing through the high solution. This avoids direct transitions from a descent into a climb or from a climb into descent respectively.

Figure 1 illustrates an example where both aircraft *A* and *B* are on climb from Flight Level *(FL) 110* to *FL150* and from *FL120* to *FL160* respectively.

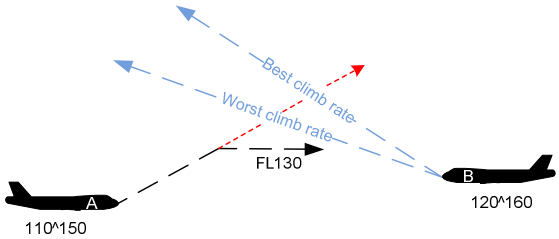


Figure 1: Assign closest level below

The climb of aircraft *A* is halted below aircraft *B* by assigning *FL130* to aircraft *A*.

The base weight of this solution type is (-0.5). Penalty values in the amount of +0.1 are additionally applied if the solution applied to A falls outside the transitional altitude band defined by A's current and cleared altitudes.

B. Assign separated levels

The second solution type involves modifying the levels of both aircraft, assuming a pair of conflicting aircraft where one aircraft is climbing and the other aircraft is descending. Figure 2 illustrates the basic concept of this solution, once again using a conflict pair of aircraft *A* and *B*. In this example, aircraft *A* is climbing from *FL110* to *FL150*, while aircraft *B* is descending from *FL150* to *FL110*.

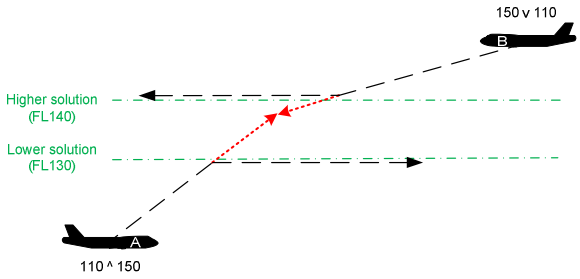


Figure 2: Assign separated levels

In this case, the applicable solution is to interrupt both the climb of aircraft *A* and the descent of aircraft *B* by assigning *FL130* to aircraft *A* and *FL140* to aircraft *B*, thereby ensuring that sufficient vertical separation between the aircraft is maintained.

The base weight of this solution type is (-0.5). Penalty values in the amount of +0.1 are added to the weight for any reverse climb or reverse descent intervention.

C. Vector behind solution

The *vector behind* solution proceeds as follows: A circle with a radius of *6nm* (nautical miles) is placed around aircraft *B* at its current position. Aircraft *A* is pointed behind aircraft *B* by vectoring it to the heading that establishes a tangent to this circle, thereby ensuring sufficient lateral separation between the two aircraft.

This solution is generally applicable to all conflicting aircraft. Its base weight is *(-0.5)*. There are no additional penalties.

Adaptation of A* to the ATC decision making task

The search space of the A* algorithm can be graphically represented by a decision tree. An example graph is shown in Figure 3. Each node in the decision tree represents a system state that, with the exception of the start node (*S*), results from the path of previous actions leading to it. The edges between the nodes represent the path of actions. Each edge has a value (shown as an integer in Figure 3) representing the cost incurred by traversing that edge. It is worthwhile to note that apart from the goal node (*G*), each node has at least one decision alternative associated with it, leading to a so-called *child node*.

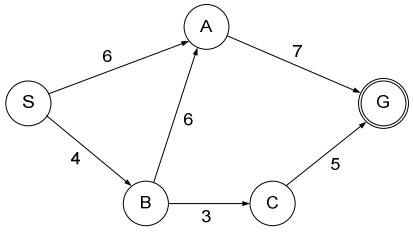


Figure 3: A* example graph

The decision making process is effectively driven by the cost function *f(x)*. That is, A* ranks each path currently under consideration based on *f(x)* to find the path with the lowest traversal cost. *f(x)* is decomposed into a so-called path-cost function *g(x)* reflecting the cost from the starting node to the node of interest, and a "heuristic estimate" *h(x)* of the distance to the goal node.

$$f(x) = g(x) + h(x), \qquad \text{Equation 3}$$

where *x* denotes some partial path. In other words, *f(x)* represents the estimated final cost of the path leading to the goal and including *x*. Under the right conditions, A* guarantees to find the path with the lowest traversal cost (Leigh, Louis, & Miles, 2007). The performance of A* relies heavily upon the heuristic estimate *h(x)*. A necessary condition for A* to find the shortest path is that the heuristic must underestimate the remaining distance.

One of the key aims in adopting the A* search scheme to the ATC conflict resolution task consists in achieving a model behavior that is closely aligned to the behavior of human controllers. For this purpose, the concept of optimization criteria was introduced. Each optimization criterion C_n encapsulates the n^{th} goal of the controller. Table 1 shows three examples for possible optimization criteria:

Table 1: Three exemplary optimization criteria

n	**Optimization criterion C_n**
1	Minimization of total number of aircraft interventions
2	Minimization of disruption to aircraft flow
3	Minimization of the controller's workload

Each optimization criterion C_n is associated with a set of descriptive attributes, A_{nk}. These attributes are represented by corresponding cost functions

$$f_{nk} = g_{nk} + h_{nk}. \qquad \text{Equation 4}$$

Summing up all the cost contributions across the individual attributes yields the final cost function for the individual criterion C_n:

$$f_n = \sum_k \left(g_{nk} + h_{nk}\right) \qquad \text{Equation 5}$$

Our initial version mainly aims at the implementation of optimization criterion C_1 from Table 1. That is, it tries to resolve all conflicts given in the scenario with the fewest aircraft interventions. However, the second criterion listed in Table 1, C_2, was additionally integrated into the model, to account for the attempts of controllers to minimize unfavorable flight maneuvers. Table 2 shows the individual attributes for C_1 and C_2.

Table 2: Attributes of the optimization criteria as per the current model implementation

C_n	k	Attribute A_{nk}
C_1	1	Preference of graph nodes of lower depth level
C_1	2	Preference of nodes showing fewer remaining conflicts
C_1	3	Number of conflicts of the aircraft subject to intervention
C_1	4	Number of occurrences of the solution of interest
C_2	1	Obstruction of unfavorable flight maneuvers

As Table 2 shows, C_1 is represented by four attributes and C_2 by one attribute respectively. The aim of the attribute A_{11} in Table 2 is to prioritize the selection of solutions that belong to graph nodes at low depth levels. The depth level of a node is determined by the number of subsequent nodes lying in the decision path, that is, by the number of actions leading to it. Therefore, the node depth defining the corresponding cost function $g_{11}(x)$ represents the number of interventions that have already occurred in the path of interest x, and that have consequently already imposed a penalty on the achievement of optimization criterion C_1.

Generally, the number of remaining conflicts in a given node establishes a good indicator for the expected number of remaining interventions. Consequently, this measure was taken to define the cost component $h_{12}(x)$ for the corresponding attribute A_{12} in Table 2. The metric was encapsulated in the heuristic component h of the cost function f as it represents a *predictive* cost estimate. The number of conflicts that the aircraft the solution acts upon is involved in represents an additional indicator for the efficiency of the solution with respect to achieving criterion C_1 in the remaining path to the goal. The number of remaining conflicts therefore forms the cost component $h_{12}(x)$ corresponding to attribute A_{12}. The underlying idea is that in comparison to solutions that are applied to aircraft that are involved in a single conflict only, solutions applied to an aircraft involved in multiple conflicts have a greater than zero probability of resolving multiple conflicts this aircraft is subject to in one go. This likelihood of efficiently minimizing the intervention count is further increased if in addition to acting on aircraft involved in multiple conflicts, the particular solution is suggested multiple times by the solution logics for resolving different conflicts. The number of total occurrences of the solution under consideration was therefore taken to define cost component $h_{13}(x)$ corresponding to attribute A_{13}.

The cost function for attribute A_{21} is simply the sum of the base weights of the solutions and the respective penalties as described in the subsection entitled *Solution Types*. While the base weights for the individual solutions are identical for all solution types in the current implementation, the additional penalties depend on the situational context. Their purpose is to prevent the selection of solutions yielding unfavorable aircraft maneuvers, such as reverse climbs and reverse descents.

Based on this set of individual cost components, the cost functions $f_0(x)$ and $f_1(x)$ are computed using Equation 5. The final cost function $f(x)$ is then just formed by adding $f_0(x)$, $f_1(x)$, and a Gaussian noise term that accounts for the probabilistic nature of the human decision maker. This noise term is characterized by a relatively small variance and therefore predominantly influences the selection of solutions belonging to the same search tree level. Impacts of this noise on solutions belonging to different tree levels are very unlikely. All parameters required for the formulation of the cost functions, including the variance of the noise, were empirically chosen in the current implementation. The effective cost $f(x)$ establishes the basis for the decision making process in the ATC search tree. This process will be discussed in the following subsection.

ATC Search Tree

An example of the resulting ATC search tree is depicted in Figure 4.

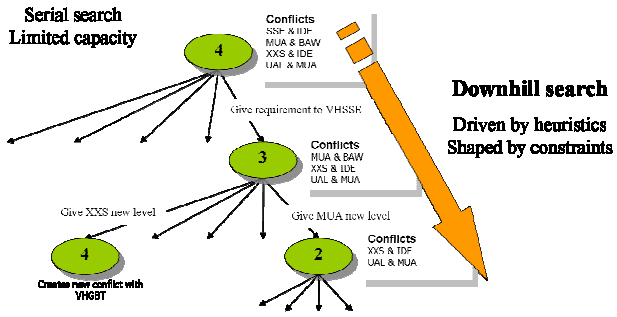


Figure 4: ATC search tree

In this example, the conflict detection model initially detects four potential conflicts between aircraft pairs in the scenario, as depicted in the root node within Figure 4. A set of potential solutions is then constructed for each of the potential conflicts present in this node. The entire set of potential solutions is then evaluated by assigning individual cost values $f_{i,j}$ to the solutions, where i *(*$i = 0$ for root node*)* and j denote the indices of the current node and the solution under consideration respectively. The solution having the smallest cost value will finally be selected and applied, creating a new child node with an associated set of conflicts. In the example in Figure 4, the solution selected in the root node resolves one of the four problems, leaving the respective child node with three remaining problem pairs. The process applied to the root node is then repeated for the child node in a recursive fashion. Figure 4 also demonstrates that solutions selected via *a-priori* evaluation may be deemed to be inefficient via *a-posteriori* evaluation. For example, the solution entitled '*Give XXS new level*' creates a new conflict, which leads to back-tracking behavior in the search process. That is, the subsequent search evaluation step may select a solution associated with the parent node, rather than propagating further down from the child node produced by the previous, inefficient solution. The overall optimization scheme effectively leads to a downhill search which is driven by the available set of solution types (heuristics) and shaped by the situational context (constraints).

Experiments

Aim and Methodology

To compare the model's behavior against the behavior of controllers, we simulated performance on a set of four different scenarios of varying complexity that were also presented to *14* En-Route, radar endorsed air traffic controllers from Brisbane Centre. Figure 5 shows the scenario with the highest complexity.

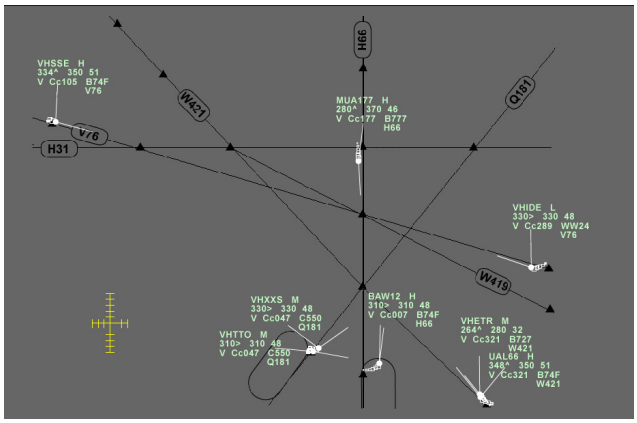


Figure 5: Scenario of highest complexity

The time participants had been endorsed as a controller ranged from *10* to *20* years. Controllers were asked to resolve the scenario by issuing restrictions to one or more of the aircraft. They were instructed to work through the scenario step by step, and to explain their actions in detail, including the evaluation of potential problems, and the processes of considering options and deciding on actions or priorities. The interviews were based on the critical decision method (Klein, Calderwood & MacGregor, 1989).

The simulation consisted of *100* runs of our decision making model for each scenario. Our interest centers on the degree to which the model used the same intervention rates and types as the human controllers. Table 3 shows the intervention types.

Table 3: Intervention types

Type	Description
H0	Intervention other than H1, H2,…, H8
H1	Vector aircraft to the left
H2	Vector aircraft to the right
H3	Issue climbing instruction
H4	Issue descent instruction
H5	Extend an existing climb
H6	Extend an existing descent
H7	Interrupt an existing climb
H8	Interrupt an existing descent

Results

The results for the scenario with the highest complexity are presented in Figures 6 and 7. Figure 6 shows the total average intervention rates for the individual aircraft for both controllers and model runs. Figure 7 shows the selection rates of the individual intervention types.

It can be seen from Figure 6 that there is a reasonable agreement between controllers and the model in selecting the aircraft that are subject to intervention. However, controllers appear to intervene with a wider range of aircraft than the model, at more variable intervention rates: Aircraft '*VHETR*' is excluded by the model in Figure 6.

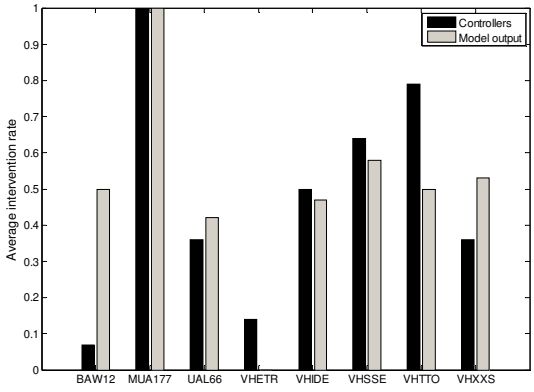


Figure 6: Total average intervention rates for the aircraft

Figure 7 demonstrates a reasonable agreement between controllers and the model in the selection of the intervention types. However, a reduced variability of the model can be observed: In contrast to controllers, the model essentially excludes the generation of intervention types *H0* ('*Intervention other than H1, H2,..., H8*') and *H5* ('*Extend an existing climb*').

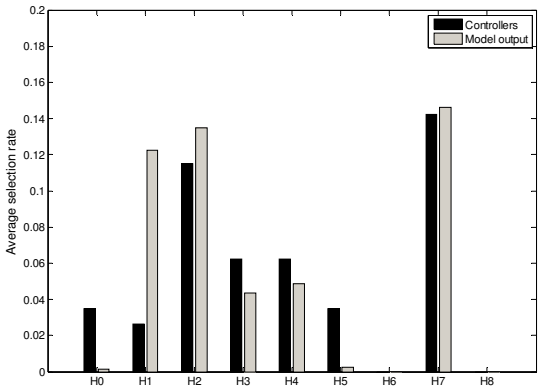


Figure 7: Average selection rates of the intervention types

Conclusions and Outlook

This paper describes a novel approach for modeling the Air Traffic Control (ATC) task using intelligent graph search. The A* algorithm was adopted to model human decision making under uncertainty and environmental constraints. This model relies on the definition of optimization criteria and associated attributes, where the attributes are represented by corresponding components of the overall cost function. The optimization criteria encapsulate properties of the situational context that influence the decision strategies of a human controller. They can consequently enable the model to alter its behavior accordingly. An initial implementation of this model is proposed that aims at minimizing the total aircraft intervention count under preservation of the realism of the generated solutions. Empirical tests demonstrate good but not perfect fits to data from human controllers. A reduced variability of the model over controllers was observed, in the selection of both the aircraft for intervention and the actual types of intervention. This variability might be induced by psychological processes that the model does not capture, such as human attention and perception.

The results suggest that the modeling concept has promise for its application to decision making in complex, dynamic task environments. We therefore plan to extend the approach in our future work by incorporating additional optimization criteria; by advancing the current decision making mechanisms; and by integrating adaptive behavior into the model.

Acknowledgments

This research project is supported by Airservices Australia. The authors acknowledge their contribution to this research in providing access to subject matter experts and facilities. The authors additionally thank Ms Selina Fothergill for providing access to the data that was generated as part of her PhD program in the School of Psychology, The University of Queensland.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Eurocontrol (2008). *Introducing the next generation of air traffic control*. Available online: http://www.eurocontrol.int/muac/gallery/content/public/docs/Introducing%20the%20next%20generation%20of%20ATC.pdf [Last accessed: *19-April-2010]*

FAA (2010). *FAA's NextGen Implementation Plan*. Available online: http://www.faa.gov/about/initiatives/nextgen/media/NGIP_3-2010.pdf [Last accessed: 19-April-2010]

ICAO (2004). *Outlook for Air Transport to the Year 2015*. (Circular 304). Montreal, Canada: ICAO.

Lee, W. P., Osma, M.A., & Maziani, S. (2009). Design of an Intelligent Route Planning System using an Enhanced A*-search Algorithm. *Third Asia International Conference on Modelling & Simulation* (pp. 40-44).

Lehmann, S., Bolland, S., Remington, R., Humphreys, M.S., Fothergill, S., Hasenbosch, S., & Neal, A. (2010). Evaluation of a Model of Expert Decision Making in Air Traffic Control. *Proceedings of the 9th Conference of the Australasian Society for Cognitive Science* (pp. 204-209).

Leigh, R., Louis, S.J., & Miles, C. (2007). Using a Genetic Algorithm to Explore A*-like Pathfinding Algorithms. *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007)* (pp. 72-79).

Loft, S., Bolland, S., Humphreys, M., & Neal, A. (2009). A Theory and Model of Conflict Detection in Air Traffic Control: Incorporating environmental constraints. *Journal of Experimental Psychology: Applied*, *15(2)*, 106-124.

Rosbe, J., Kieras, D., & Chong, R. (2001). *Modeling with perceptual and memory constraints: an EPIC-Soar model of a simplified enroute air traffic control task* (Tech. Rep. AFRL-HE-WP-TR-2002-0231). Ann Arbor, MI: Soar Technology Inc.

Todd, P. M., & Gigerenzer, G. (2007). Environments that make us smart: Ecological rationality. *Current Directions in Psychological Science*, *16(3)*, 167-171.

Computational Models of Perceptual Learning Across Multiple Auditory Tasks: Modeling Daily Learning Limits as Memory Decay

David Little (d-little@u.northwestern.edu)
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, IL 60208, USA

Bryan Pardo (pardo@northwestern.edu)
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, IL 60208, USA

Abstract

Humans have a remarkable ability to adapt their perceptual acuity to the task at hand, commonly referred to in the literature as perceptual learning. Understanding this ability at a computational level may have important implications across a wide variety of different psychological phenomena. There is evidence suggesting this ability plays an important role in speech comprehension, mathematics, and perceptual expertise, for instance. Computational models of perceptual learning have largely focused on hypothesizing how one or more mechanisms might explain the observed perceptual learning for a single task. Here we explore how a single model might explain the learning curves across two auditory perceptual learning tasks. Our results suggest that an ideal observer model with noisy input can predict learning when daily limits are not reached, and that daily limits on learning can be modeled by a decay of memory for trials observed on the current day of practice.

Keywords: perceptual learning; ideal observer; plasticity vs. stability; frequency discrimination; duration discrimination; temporal interval discrimination

Introduction

Humans have a remarkable ability to adapt their perceptual acuity to the task at hand, commonly referred to in the literature as perceptual learning (Fahle and Poggio, 2002). Perceptual learning has been demonstrated in many different experiments. In vision for instance, there are studies of vernier hyper-acuity (Poggio et al., 1992), orientation discrimination, and spatial frequency discrimination (Fiorentini and Berardi, 1980). Examples in the auditory domain include results for frequency discrimination (Demany, 1985), and temporal interval discrimination (Wright et al., 1997). Perceptual learning is often characterized as being highly specific both to the task (Fiorentini and Berardi, 1980), and to the specific location or range within a dimension (Wright and Zhang, 2009; Poggio et al., 1992).

There is evidence that perceptual learning is important for a great variety of real world tasks humans face (Kellman and Garrigan, 2008). There is data suggesting that perceptual learning helps us during speech comprehension (Norris et al., 2003), that it can help children with dyslexia (Hayes et al., 2003) and that it has an important role to play in the comprehension of mathematical formulae (Kellman et al., 2008).

Computational models of perceptual learning have the potential to enable better predictions and to help us better understand human data. Past computational work studying perceptual learning has largely focused on how specific mechanisms might explain the particular properties of perceptual learning for a single task (e.g. Poggio et al., 1992; Petrov et al., 2005; Jacobs, 2009). Such studies focus on the question of how and/or where perceptual learning occurs within the human brain for a *single* perceptual task. Our goal here is to develop a model of *multiple* perceptual learning tasks. By looking across several tasks we can ultimately constrain our model by requiring that a single parameter explain qualitatively different results across several tasks. Our research also differs from past work in that, to the best of our knowledge, there are no computational studies of perceptual learning for auditory tasks.

Here we model auditory perceptual learning across two tasks: temporal interval discrimination and frequency discrimination, as discussed in Wright and Sabin (2007). By modeling learning across several tasks our goal is to gain a better understanding of why learning does or does not occur under various training conditions. Our focus here is on modeling the daily limits of learning: it was observed in Wright and Sabin (2007) that training beyond some point in a single day does not yield extra learning. Our results suggest that limits on daily learning can be modeled by a decay of the memory of trials observed on the current day of practice. This decay is consistent with numerous studies of consolidation suggesting newly acquired information in a day begins in a volatile state, and is not made permanent until memories are consolidated (e.g. McGaugh, 2000).

Human Data

This section reviews the human data and results originally described in Wright and Sabin (2007). In this paper, they examined how varying the number of training trials practiced per day affected learning over multiple days on two auditory discrimination tasks: frequency discrimination and temporal-interval discrimination. The basic question asked in the paper was "how much daily training is sufficient for learning to occur?" The set of relevant findings we model here is that extra trials practiced per day, past a certain point, do not appear to lead to any further learning.

During the experiments, subjects practiced either a temporal interval discrimination task or a frequency discrimination task for a single session each day of practice, for six days over no more than two weeks. Each task was a two inter-

val forced choice: on each trial participants must pick which of two stimuli is longer (higher) for the interval (frequency) discrimination task. The stimuli were adjusted adaptively as practice continues. As subjects do better, the difference between the standard (shorter) and comparison (longer) stimulus gets smaller. This is a common procedure used in psychophysics to find a performance threshold. The experiments consisted of a two-by-two design over number of trials in a day (360 or 900) and task type (frequency or interval). Each of the four conditions used a different set of participants. Further details of the training procedure can be found in Wright and Sabin (2007).

The data suggest there are important within-day limitations on human perceptual learning: extra practice past some point does not improve learning any further and insufficient practice in a day yields little to no learning across days. Further, the number of trials needed for learning is task dependent. Specifically, if a subject practiced the temporal interval task for 360 trials per day this yielded the same amount of learning as 900 trials per day. During the practice of frequency discrimination, 900 trials of practice produced significantly more learning than 360 trials. All the above observations were statistically verified. Details can be found in Wright and Sabin (2007).

Here our focus will be on modeling this first observed limit within a day: past a certain point no further trials within a day appear to yield further learning.

Method

This section describes and justifies the basic principles of our model (which is evaluated in our Results section).

In terms of Marr's (1982) levels of analysis, we restrict ourselves largely to the informational level. When operating at this level we make no claims about what algorithm is used internally or how that algorithm is implemented in the human brain. Since the informational constraints are not yet fully understood for the modeled experiments, we believe this is an appropriate level of analysis for the time being.

Specifically, we utilize an ideal observer analysis (Geisler, 2003). The idea is to consider human performance in reference to an ideal observer, which processes information in a way that is 'optimal' in some sense. This can help to avoid conflation between two distinct types of limitations on human behavior. These are, respectively, informational and psychological limits. Informational limits are those limits that are inherent to the task: even if an observer were to be perfect they would still be subject to informational limits. An example of an informational limit would be noise in the input: any learner, no matter how smart, would have to deal with the problems introduced by noise. Psychological limits on the other hand are a product of resource limitations on the part of the observer: if the observer was 'smarter' they might be able to improve their behavior. An example of a psychological limit would be memory: with limited memory only so many units of information can be stored, but a smarter learner would be able to store more, and so improve behavior.

Since any observer is subject to informational limits, we always assume these are present. Psychological limits are then only hypothesized as necessary: if a behavior could be explained solely in terms of informational limits, then no additional psychological limits would be hypothesized. Throughout our discussion we make a distinction between the ideal observer and the proposed psychological limits.

Based upon this principle we designed a system capable of modeling the observed limits on the amount of useful daily practice, as observed in Wright and Sabin (2007). We begin by describing the commitments we made regarding what information is available to humans when performing this task. We then describe an ideal observer model, and then identify the ways in which our model of human performance differs from the ideal observer.

Input

The input to our model is consistent with the following properties, which are explained in more detail below. These choices represented a number of educated guesses as to the form of the information humans receive, based on psychophysical and physiological findings.

1. Differentiation along task relevant dimensions: e.g. 1 kHz is represented differently than 2 kHz.
2. Corruption by noise.
3. Range specificity: e.g. energy near 1 kHz is encoded separately from energy near 2 kHz.
4. Weber's law.

Each of these properties is based on many observations. Clearly the input is differentiated along task relevant dimensions: if there was no differentiation at all along a task relevant dimension, different stimuli of a task would appear the same to us. Second, there are many evident sources of noise to perceptual data, from noise in the world, noise during the transduction of sound to neural impulses, and noise in the nervous system itself. Range specificity is consistent with the narrow generalization patterns observed during perceptual learning tasks (e.g. Poggio et al., 1992; Fiorentini and Berardi, 1980; Wright and Zhang, 2009) and with the great multitude of physiological data suggesting that neurons are responsive to specific, limited ranges of stimuli (e.g. Brugge, 1992; De Valois and De Valois, 1980). Range specificity is distinct from differentiation: for instance a single source of information can differentiate between 1000 Hz and 200 Hz by using a single number, 1000 or 200, which would not be specific to a particular range; range specificity means that the sources of information (e.g. neurons) representing 1000 Hz and 200 Hz would be at least somewhat disjoint.

Weber's law—which states that the minimum discernible difference (or just noticeable difference) between stimuli along a particular dimension is proportional to the magnitude

of the stimuli along that dimension—has long been established as a useful rule of thumb for perceptual data (Moore, 2006).

In addition we make a number of simplifying assumptions. We assume that, prior to perceptual learning, the input has been correctly broken down into the various experimentally relevant units (i.e. each input to our model represents a single stimulus). How this happens in humans is not the focus of this modeling experiment. Our second assumption is that the dimensions of the stimulus are independent cues for the tasks in question, which is correct for the two tasks we consider.

Frequency and temporal interval are represented on a log scale. The frequency representation is found directly from the model described in (Wang and Shamma, 1994)[1]. Our interval representation is found based on a windowed autocorrelation of the stimulus onsets. Both of these choices yield a representation consistent with our above assumptions. The input to the observer is a vector $\mathbf{x}$ of 228 terms: 128 features representing frequency and 100 features representing temporal interval. There are 128 bins for frequency because this is the resolution of the model from (Wang and Shamma, 1994). The number 100 for the interval representation was chosen aribtrarially. The observations made in the Results section did not change when this number was changed to 50 or 200.

We permute the input by an experimentally determined amount of noise specific to each dimension of the stimulus (σ_t^2 for the interval noise and σ_f^2 for the frequency noise). Note that since the representation is deterministic, when it is applied directly to an ideal observer it would always respond correctly. Choosing to represent all error in the system as input noise is conservative in the sense that the ideal observer will do more poorly under these conditions than if some of the error was modeled as output noise, for instance.

Ideal Observer

We implement the ideal observer using a Bayesian approach to learning: a probabilistic model which is learned during the course of practice is used to determine the correct response on each practice trial. This model is not meant to be a psychologically plausible model of perceptual discrimination. It is an optimal decision maker for this task, whose performance can thus be used to identify in what ways humans are different from an optimal choice.

For a single trial, there are two stimuli, and each stimulus is encoded as a vector, $\mathbf{x}$, of 228 terms: 128 features for the frequency representation and 100 for the interval representation. Since we know that this input is permuted by Gaussian noise the likelihood of each stimulus type—the standard (or longer) and the comparison (or shorter)—can be modeled using a Normal distribution. We calculate the posterior model analytically by assuming a conjugate prior (Gelman, 2004). Learning and use of this model then follows a straightforward application of Bayes rule and conjugate priors, described below.

[1]An implementation of this model can be found at `http://www.isr.umd.edu/Labs/NSL/Register.htm`.

Specifically the ideal observer learns a model of the standard (e.g. shorter) stimulus, S, and one for the comparison (e.g. longer) stimulus, C for each task. Each model is a multivariate Normal distribution, describing the probability of observing a given input vector $\mathbf{x}$. This distribution is specified by the mean vector μ_S for the standard model and μ_C for the comparison. Each mean has 228 terms (one for each frequency and interval value) and covariance matrix Σ_S,or Σ_C with 228 rows and columns. Hence, the probability of observing a given input vector, assuming it is the standard is as follows.

$$p(\mathbf{x}|\mu_S,\Sigma_S) \propto \exp\left[(\mathbf{x}-\mu_S)^T\Sigma_S^{-1}(\mathbf{x}-\mu_S)\right] \quad (1)$$

To learn the model of S and C the observer must be provided with examples of the standard and the comparison. These can be used to determine the probability of a given μ_s and Σ_S, using Bayes rule. Below $\mathbf{x}_t$ represents the example of the standard (shorter) stimulus observed at time t. On each practice trial, feedback is given to the observer after it responds, so on each trial the observer is provided with another example of both the standard and the comparison.

$$p(\mu_S,\Sigma_S|\mathbf{x}_1) \propto p(\mathbf{x}_1|\mu_S,\Sigma_S)p(\mu_S,\Sigma_S) \quad (2)$$

$$p(\mu_S,\Sigma_S|\mathbf{x}_1,\mathbf{x}_2) \propto p(\mathbf{x}_2|\mu_S,\Sigma_S)p(\mu_S,\Sigma_S|\mathbf{x}_1) \quad (3)$$

$$\vdots$$

$$p(\mu_S,\Sigma_S|\mathbf{x}_t,\cdots,\mathbf{x}_1) \propto p(\mathbf{x}_t|\mu_S,\Sigma_S)p(\mu_S,\Sigma_S|\mathbf{x}_{t-1},\cdots,\mathbf{x}_1) \quad (4)$$

Equation 2 requires that the prior probability $p(\mu_S,\Sigma_S)$ be known, which we will discuss shortly. Subsequent equations show how an example $\mathbf{x}_t$ updates the distribution of parameters for S. Given a set of training examples, the probability of $\mathbf{x}$ for model S is defined as follows:

$$p(\mathbf{x}|S) = \iint p(\mathbf{x}|\mu_S,\Sigma_S)p(\mu_S,\Sigma_S|\mathbf{x}_t,\cdots,\mathbf{x}_1)\,d\mu_S\,d\Sigma_S \quad (5)$$

Equation 5 can be calculated given that conjugate priors are used. Once $p(\mathbf{x}|S)$ and $p(\mathbf{x}|C)$ are known, Bayes rule can be used to find the probability that the model should indicate that the first (or second) stimulus is the longer of the two stimuli presented on a trial.

To use this Bayesian learner we must define the prior of the model ($p(\mu,\Sigma)$), representing what people know before they practice the task. There are many deep questions that might be asked about what humans know about task before practice and how they know it. Here we choose a simple approach to selecting a prior: starting with a naive model (with mean vector 0, and an identity matrix for covariance) the learner is presented an experimentally determined number of trials of each task (N_t trials of the interval task, and N_f trials of the frequency task).

Psychological Limits

We consider two modifications of the ideal observer described in the previous section to model psychological limits. The first is a direct result of the observation in (Wright and Sabin, 2007) that for these tasks people do not appear to learn within a day but only across days, hence our 'daily' model. The 'daily' model learns as per the ideal observer, but *responds* based only on data from previous days of practice, and not from the current day. This is used as a baseline model during our evaluation in the next section. Our second modification models the hypothesis that there is a daily limit on training: it does this by introducing a decay on the knowledge obtained from trials on the current day. The 'decay' model incorporates this limit, in addition to the limits of the 'daily' model. This proposed decay is a novel contribution of this paper in that it has not been considered as an explanation for the observed daily limit in these tasks before.

The decay in the model is implemented as follows. Given a new example, $\mathbf{x}_{t+1}$ at trial $t+1$, normally the model of the standard (or comparison) stimulus is updated according to Bayes rule in the following manner.

$$f_{t,d}(\mu,\Sigma \mid D_{t+1},C) \propto p(\mathbf{x}_{t+1} \mid \mu,\Sigma) f_{t,d}(\mu,\Sigma \mid D_t) f_{T,d-1}(\mu,\Sigma \mid C) \quad (6)$$

In Equation 6, the function $f_{t,d}$ is the distribution over stimulus parameters μ and Σ, on trial t of day d. D_t represents all training examples observed for the current day, and C represents all examples observed on previous days (i.e. the consolidated information). T is the maximum number of trials observed in a day. This expresses the same relation expressed in Equation 4. However, with memory decay, this optimal update is changed to the following rule.

$$f_{t,d}(\mu,\Sigma \mid D_{t+1},C) \propto p(\mathbf{x}_{t+1} \mid \mu,\Sigma) f_{t,d}(\mu,\Sigma \mid D_t)^{1-L} f_{T,d-1}(\mu,\Sigma \mid C) \quad (7)$$

Equation 7 means that memory decay occurs for trials observed on the current day. The distribution learned from a previous day of practice remains in the same state it was at the end of that day of practice (as determined by $f_{T,d-1}$), including any decay that occurred on that day. This decay is a reasonable representation of loss of information within a day. If $L = 0$ then the model is equivalent to the 'daily' model. If $L = 1$ the daily practice has no effect on the model. Values between 1 and 0 represent a continuum between these two extreme conditions.

Note that it's possible the decay should be over some shorter period of time, rather than including all trials within a day. For instance, it has been suggested that if a short nap is taken this has the same benefit as a night of sleep for purposes of perceptual learning (Mednick et al., 2003). This could easily be explained by our model by having D_t contain only those trials that occur after the last period of sleep, and C contain all other trials. However, this is beyond the scope of the experiments modeled in this paper.

Results

Our hypothesis is that the observed daily limits on learning can be modeld as a decay of the memory of trials on the current day (while leaving memory of previous days' trials untouched). We compared a computational model that had this hypothesized limit (the 'decay' model) to one that did not (the 'daily' model). To compare these models to human data we ran the same adaptive track blocks used in (Wright and Sabin, 2007) to determine thresholds. On each trial the original audio input was represented to the model and a response was given, and then feedback about the correct answer was used by the model to learn. This procedure was repeated 30 times, to simulate 30 different experimental subjects. This number was chosen to yield satisfactory statistical power for our analysis.

Results for the two models are discussed below. Figure 1 displays the results of these two models alongside human performance, as observed in (Wright and Sabin, 2007). From the graphs it appears that both models appear to fit the results well for the 360 trials/day interval discrimination condition and the 900 trials/day frequency discrimination condition. The decay model appears to also fit the data for the 900 trials/day interval discrimination condition better than the daily model.

Our statistical tests supported this observation. For each iteration, condition and day of a model we found the squared error to the mean human performance on that day. Table 1 shows the mean squared errors across conditions and models. Because the human and model data were qualitatively different in the 360 trial/day frequency condition we excluded it from the below analysis, since any differences between the two models in this condition will not be meaningful. A 3x2x6 ANOVA across conditions and models and within days of these squared errors showed a main effect across condition and model ($p < 0.028$). A Tukey's HSD test suggested that the decay model's mean squared error was significantly less than the daily model's mean square error ($p < 0.014$).

	Interval		Frequency	
	360	900	360	900
daily	2.68(0.32)	3.40(0.37)	18.09(1.1)	1.24(0.11)
decay	2.77(0.29)	2.03(0.19)	24.60(1.2)	1.19(0.14)

Table 1: Mean squared errors for the daily and decay model. Errors are the difference between a model threshold and the mean for the human data on a given day and condition. Numbers in parenthesis indicated standard errors.

Model parameters (which determined noise and prior knowledge) were adjusted so that the daily model matched human performance on day 1 and day 6 of all conditions except the 360 trials/day frequency condition, using the optimization algorithm described in Huyer and Neumaier (2008). These conditions were chosen because this was where learning appeared to occur. Since the noise of the model strongly influences the final performance of our model on day 6 (after learning), it should be fit to those conditions where learning

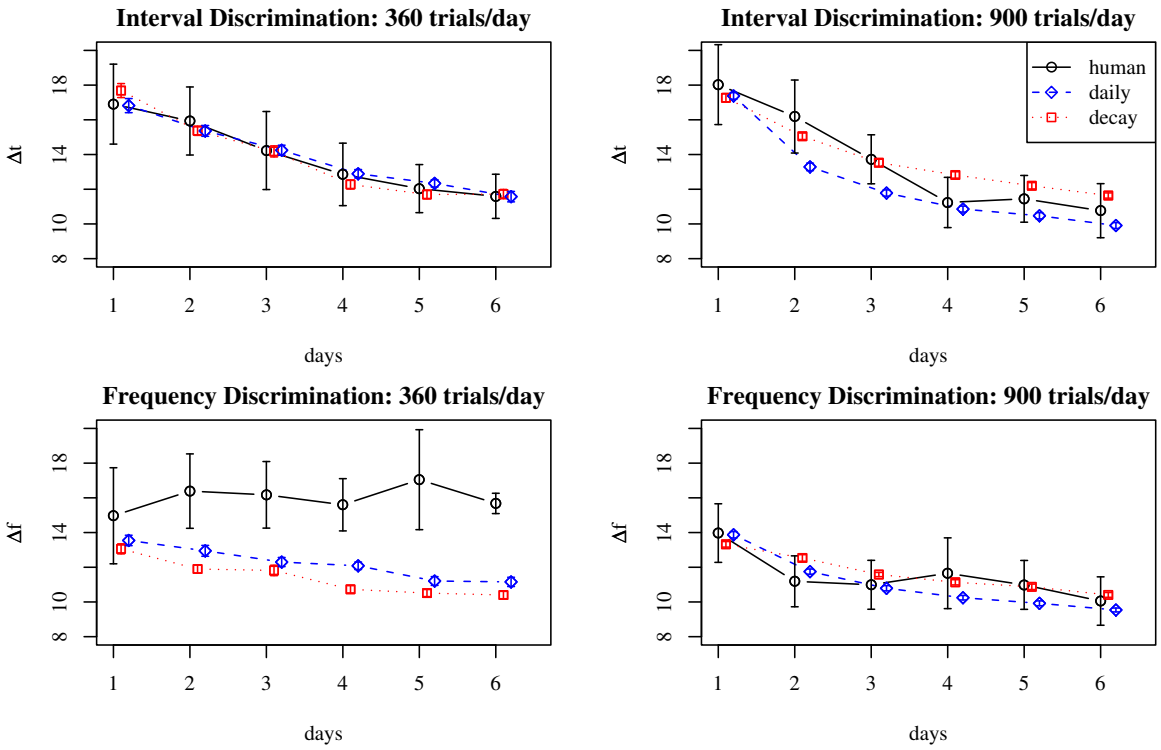


Figure 1: Results for 'daily' and 'decay' models compared to human performance. Results are averaged across 30 runs of each model. Δf represents the difference between the standard (lower) and comparison (higher) frequency stimuli for the frequency task, and Δt the difference between the standard (shorter) and comparison (longer) stimuli for the interval task. The adaptive track method used finds the 79% accuracy of a subject or model. Lower delta's indicate that the human participants are performing better. A model is accurately predicting the human data if its curve is closer to the human curves. Bars indicate standard errors.

appears to occur. The parameters for prior knowledge are dependent on this noise and so we fit it jointly and under the same conditions as the noise. For reasons that will become clear below we also matched this data to human performance on day 2 of the 900 trials/day interval discrimination task.

An analogous procedure was used for the decay model except that the decay parameter (L) was also adjusted, and fit to the same days as above. The data was fit to day 2 for the 900 trials/day interval. This single day was chosen so as to be minimal (to avoid overfitting) and such that it was a place where L might cause an observable change in the results. This same day was used for the daily model above so that both procedures had access to the same information. All parameters were selected so as to maximize the posterior probability of the selected days given the human thresholds (assuming thresholds on a day are Normally distributed, which is consistent with the analysis in Wright and Sabin (2007)).

Discussion & Conclusions

In this paper we evaluated a model of learning across two simple auditory tasks. Our goals differed from that of previous work (e.g. Poggio et al., 1992; Petrov et al., 2005; Jacobs, 2009) in that we considered auditory tasks rather than visual tasks, and in that we considered a single model that could explain results across several tasks. To the best of our knowledge, ours is the first computational model of auditory perpetual learning.

Our contributions in this paper were to show that our 'daily' model could accurately model two of the four considered experimental conditions and that our 'decay' model (which included a decay of memory for the trials observed on current days) could model an additional condition (900 trials/day of interval discrimination). This result suggests that the minimal difference in learning for this condition and the 360 trials/day of interval discrimination could be caused by memory loss.

Modeling this condition using memory decay is consistent with numerous studies of consolidation suggesting newly acquired information begins in a volatile state, and is not made permanent until consolidation occurs after practice is complete (McGaugh, 2000). In cases where consolidation is interfered with, perhaps what happens is that the memory of observed trials on a task decays before it can be stored in long term memory. The 900 trial/day interval discrimination condition would then represent an intermediate case where consolidation has yet to occur (perhaps because practice is still

ongoing), and hence memory decay degrades part of what has been learned. Once practice is complete consolidation can commence given that no other interfering effects occur.

The model presented here does not explain one of the experimental conditions we considered (the condition with 360 a trials of frequency discrimination a day). In this condition people did not appear to learn but our model did, suggesting that the human results cannot be explained simply by the fact that fewer trials were observed, which is consistent with the observations made in Wright and Sabin (2007). We have considered several possible factors that might explain this condition, but as of yet, no factor we have considered can explain both the 360 trial interval discrimination task and the 360 trial frequency discrimination task using a single parameter. Any model using a different parameter per condition would be meaningless in that any such model would fit the data. This suggests to us that more perceptual learning tasks must be considered before a meaningful model for this condition and others like it can be proposed, and is a goal of future work. In the future, it is also our plan to consider conditions where people practice several tasks at once, to help us understand why learning does or does not occur, such as in (Banai et al., 2009).

This paper thus represents a first step toward developing a model that can explain learning across a number of perceptual learning tasks, rather than modeling behavior on a single task. Such a model must consider more constraints than one that doesn't, which can help provide a better understanding of how and when perceptual learning occurs and why.

Acknowledgements

We'd especially like to give thanks to Beverly Wright for her helpful comments, and to Andy Sabin for his help providing access to the data from Wright and Sabin (2007). This research was supported, in part by Northwestern University's Cognitive Science program and by US National Science Foundation grant 0643752.

References

K. Banai, JA Ortiz, JD Oppenheimer, and BA Wright. Learning two things at once: differential constraints on the acquisition and consolidation of perceptual learning. *Neuroscience*, 2009.

J Brugge. *The Mammalian Auditory Pathway: Neuroanatomy*, volume I, chapter An overview of central auditory processing., pages 1–22. Springer Verlag, Wiesbaden, Germany, 1992.

RL De Valois and KK De Valois. Spatial vision. *Annual Review of Psychology*, 31(1):309–341, 1980.

Laurent Demany. Perceptual learning in frequency discrimination. *The Journal of the Acoustical Society of America*, 78(3):1118–1120, 1985. doi: 10.1121/1.393034. URL http://link.aip.org/link/?JAS/78/1118/1.

M. Fahle and T. Poggio. *Perceptual learning*. MIT Press Cambridge, MA:, Cambridge, MA, USA, 2002.

A. Fiorentini and N. Berardi. Perceptual learning specific for orientation and spatial frequency. *Nature*, 287:43–44, 1980.

A. Gelman. *Bayesian data analysis*. CRC press, 2004.

E.A. Hayes, C.M. Warrier, T.G. Nicol, S.G. Zecker, and N. Kraus. Neural plasticity following auditory training in children with learning problems. *Clinical Neurophysiology*, 114(4):673–684, 2003.

W. Huyer and A. Neumaier. SNOBFIT–Stable Noisy Optimization by Branch and Fit. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):9, 2008.

RA Jacobs. Adaptive precision pooling of model neuron activities predicts the efficiency of human visual learning. *Journal of Vision*, 9(4):22, 2009.

P.J. Kellman and P. Garrigan. Perceptual learning and human expertise. *Physics of Life Reviews*, 2008.

P.J. Kellman, C. Massey, Z. Roth, T. Burke, J. Zucker, A. Sawa, K.E. Aguero, and J.A. Wise. Perceptual learning and the technology of expertise Studies in fraction learning and algebra. *Pragmatics & Cognition*, 16(2):356–405, 2008.

D. Marr. *Vision: A Computational Approach*. Freeman & Co., San Francisco, 1982.

James L. McGaugh. Memory–a century of consolidation. *Science*, 287(5451):248–251, 1 2000.

S. Mednick, K. Nakayama, and R. Stickgold. Sleep-dependent learning: a nap is as good as a night. *Nature Neuroscience*, 6(7):697–698, 2003.

Brian C. J. Moore. *An Introduction to the Psychology of Hearing*. Elsevier, London, UK, 5th edition, 2006.

D. Norris, J.M. McQueen, and A. Cutler. Perceptual learning in speech. *Cognitive Psychology*, 47(2):204–238, 2003.

A.A. Petrov, B.A. Dosher, and Z. Lu. The Dynamics of Perceptual Learning: An Incremental Reweighting Model. *Psychological Review*, 112(4):715, 2005.

Tomaso Poggio, Manfred Fahle, and Shimon Edelman. Fast perceptual learning in visual hyperacuity. *Science*, 256(5059):1018–1021, 1992. URL http://www.jstor.org/stable/2877128.

Kusan Wang and Shihab Shamma. Self-normalization and noise-robustness in early auditory reprersentations. *IEEE Trans. On Speech and Audio Processing*, 2(3):421–435, 1994.

B.A. Wright and Y. Zhang. A review of the generalization of auditory learning. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1515):301, 2009.

B.A. Wright, D.V. Buonomano, H.W. Mahncke, and M.M. Merzenich. Learning and generalization of auditory temporal-interval discrimination in humans. *Journal of Neuroscience*, 17(10):3956–3963, 1997.

Beverly Wright and Andrew Sabin. Perceptual learning: how much daily training is enough? *Experimental Brain Research*, 180(4):727–736, 07 2007.

A Human-Markov Chain Monte Carlo Method For Investigating Facial Expression Categorization

Daniel McDuff (djmcduff@mit.edu)
MIT Media Laboratory
Cambridge, MA 02139 USA

Abstract

This paper demonstrates how a human-Markov Chain Monte Carlo (MCMC) method can be used to investigate models of facial expression categorization. Data were collected from four participants. At each step participants were asked to select a representation from a pair, that most resembled a particular emotional state; this was repeated iteratively. As such, they formed a component in the MCMC process. The representations were line drawn facial images with 10 nodes and four degrees of freedom. The judgements formed samples for a set of interleaved Markov Chains. These were mapped to a two-dimensional plane using Generalized Discriminant Analysis. We contrast the results of the MCMC task with those of a second discrimination task.

Estimates of the distributions along each of the four dimensions showed that for the outer eyebrow and lip corner variables one of the categories could be discriminated with confidence.

The average examples from both MCMC and discrimination tasks were both plausible. However, the MCMC method allowed for greater sampling from areas of high interest. Finally, we show that a naive Bayes classifier trained on the MCMC data can be used to successfully predict human classification in a discrimination task.

Keywords: MCMC; categorization; representations; facial expressions; emotion.

Introduction

The face provides an important channel for communicating affect. Much emotional information is encoded in people's facial expressions (Darwin, Ekman, & Prodger, 2002). However, affect label mapping from facial expressions is often difficult to define. In this paper we apply a Markov Chain Monte Carlo (MCMC) method (Neal, 1993) to investigate facial expression categorization. Using humans as components in a MCMC process we demonstrate how we can sample from cognitive representations of facial expressions.

MCMC is a sampling method that can be used to estimate probability density functions. A parameter space is searched via Markov Chains. The sampling procedure forms a chain that can be shown to tend to the correct distribution (Neal, 1993). In an environment where the distributions of interest are likely to occupy a small subspace only, MCMC can be an efficient sampling method.

Emotions are controversially defined. However, Ekman and Friesen's (Ekman & Friesen, 1978) set of six basic emotions are an accepted set of simple examples. These six are used as a starting point for our study: anger, disgust, fear, happiness, sadness and surprise.

This paper investigates how people map observed facial expressions to affect labels. Griesser et al. (Griesser, Cunningham, Wallraven, & Bulthoff, 2007) consider a psychophysical investigation of facial expressions. Scene parameters were systematically manipulated in order to investigate the importance of particular facial regions in expression recognition. Padgett (Padgett & Cottrell, 1997; Padgett, 1998) investigates representations of facial images for emotion classification. However, only 97 images are included in the data set. As a result there are a limited number of examples in a high dimensional space from which participants were forced choose one. Both these studies consider a pre-scripted set of stimuli and do not allow efficient exploration of each participant's psychological representations by allowing them to accept and reject samples based on how they fit with the category. Padgett represents human face judgements under multi-dimensional scaling (MDS). Such a method allows for a quantitative measure of similarity in the relationships between facial expressions.

This work considers human labels for expressions rather than the subjects state when displaying the emotion. It is important to consider that a persons evaluation of another affect given their facial expression may not be representative of their actual internal state.

Reasonable facial expressions for a particular emotion label are likely to occupy only a small subspace of the total space of possible expressions. This motivates the use of an MCMC method. MCMC allows regions within a facial action feature space to be populated with labels more efficiently that a discrimination task.

In particular, we investigate the significance of each feature dimension in the categories found. We estimate the density distributions for each category along each dimension. For a simple three category case considered, certain dimensions allow a particular category to be discriminated with confidence.

This is the first work I am aware of that models the relationship between emotional states and facial expressions drawn from continuous values within a multi-dimensional feature space. We allow the participants to navigate to an area of high association with the particular label and sample from this region more frequently (Neal, 1993). Representations are not limited by the number of examples in a data set but only by the ranges placed on the variables.

Related Work

Nosofsky's Generalized Context Model (GCM) of classification proposes that people represent categories by storing exemplars in memory (Nosofsky, 1986). The prototype theory assumes a category's mental representation is based on a prototypic exemplar (Dopkins & Gleason, 1997). In contrast, the exemplar theory assumes a set of exemplars are encoded

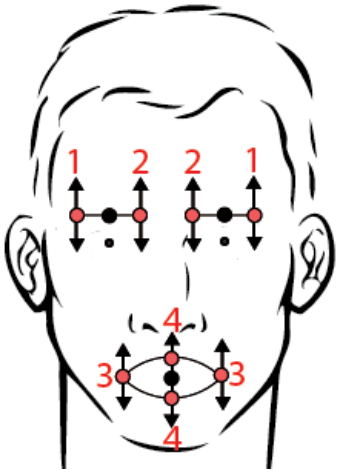


Figure 1: Face representation used in the tests. There are four degrees of freedom. 1. Position of outer eyebrows, 2. Position of inner eyebrows, 3. Position of lip corners and 4. Lip center separation. Center of the eyebrows was fixed (black node). Point about which lip center separation was measured was fixed (black node).

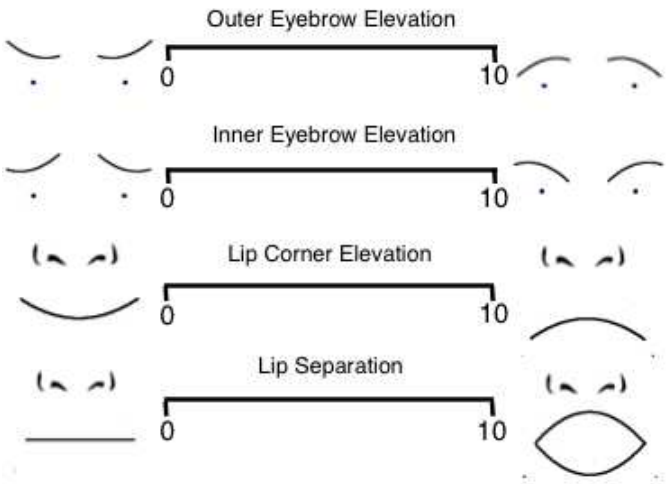


Figure 2: Continuous ranges of four free parameters on the face. Representations of the extreme cases are shown at either end of the scales.

in the category's mental representation (Nosofsky & Palmeri, 1997). A new entity is compared to the exemplars in order to establish whether it belongs to the category.

Sanbourn et al. (Sanborn & Griffiths, 2008; Sanborn, Griffiths, & Shiffrin, 2009) were the first to demonstrate the use of people as components in an MCMC algorithm, in order to explore psychological categories. A method was verified and used to demonstrate that human-MCMC can be used to estimate the structures of real-world animal shape categories.

Padgett (Padgett & Cottrell, 1997) considered representation of facial images for emotional classification. However this study is constrained by the fact that the facial image data set used was limited to a small number of images. The training data relied upon is limited in many cases as the images must be subject to agreement by expert labelers.

Methodology

This is the first investigation, to my knowledge, using cartoon representations of faces in order to investigate categorization of affect by facial expressions. As such it was necessary to begin with a facial representation having a small number of degrees of freedom. A cartoon representation was created with four degrees of freedom that allowed variation of eyebrows, lip corners and lip separation. These are demonstrated in Figure 1.

The limits placed on the displacement of each node are shown in Figure 2. The representation was symmetrical (eyebrows mirrored one another as did the left and right sides of the mouth). A restriction was applied in all tests that prevented the center of the eyebrows being the lowest point. This was the only restriction on the movement other than parameter range limits described. The degrees of freedom loosely correspond to the following action units which are identified in Ekman's (Ekman & Friesen, 1978) Facial Action-unit Coding System (FACS).

Outer Eyebrows - Outer Brow Raiser (AU2).
Inner Eyebrows - Inner Brow Raiser (AU1), Brow Lowerer (AU4).
Lip Corners - Lip Corner Puller (AU12), Lip Corner Depressor (AU15).
Lip Separation - Lips Part (AU25), Jaw Drop (AU26), Mouth Stretch (AU27).

In a set of initial tests two participants performed discrimination tasks with three facial representations. The first presented a mouth, nose and eyebrows where the nodes were joined by straight lines. The second added an outline of the face to the image. The third joined the nodes with smooth curves and also contained the outline of the face, as in Figure 1. The participants more consistently labeled the expressions given the third representation. As a result, this was used for the subsequent tests. This was a male face. Investigation into the effects of gender and ethnicity in this domain are not considered here.

All tests described in this paper were performed on a 15" MacBook Pro. Processing of the data and all GUI interfaces were created in MATLAB. None of the participants in the study were given rewards for completing the tasks. This study was approved by the Massachusetts Institute of Technology Committee On the Use of Humans as Experimental Subjects (COUHES).

Experiments

Three experiments were designed. The preliminary experiment was carried out to identify appropriate categories for the

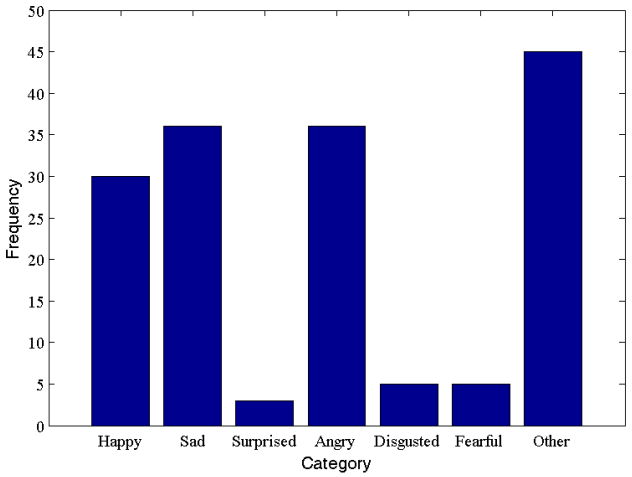


Figure 3: Histogram of results from the preliminary experiment, showing the frequency with which each category was chosen. Four participants labeled 40 different faces each.

human-MCMC tests. The human-MCMC experiment was then conducted to collect samples from these categories. The discrimination experiment was carried out to validate the distributions formed by the MCMC tests.

Preliminary Experiment

In a preliminary experiment four participants were separately shown a series of 40 cartoon faces and were asked to visually categorize them as angy, disgusted, fearful, happy, sad, surprised or other. The visual stimuli were generated from a uniform distribution over the parameter ranges shown in Figure 2. Representations outside these ranges were not considered as they were significantly different from natural movements, as judged by two participants in the initial tests.

Figure 3 shows a histogram of results from the preliminary discrimination experiment. Surprised, disgusted and fearful were each identified as the expression label in less than 5% of cases.

The results demonstrate that the four degree of freedom faces were not versatile enough to clearly represent all of the states. For instance the widening of the eyes that might be expected in a fearful expression was not represented.

There are likely to be many other indicators that influence our judgement of a person's affect that are not captured here. Ekman's facial action coding system (FACS) contains over 60 facial actions and movements many of which have been shown to discriminate between affective state (El Kaliouby & Robinson, 2005). These include skin texture changes, more subtle facial actions and movements. Examples are: nose wrinkles, head nods, shakes and tilts. Contextual information is also absent in our stimuli.

As a result, the affect categories were restricted to happy, sad and angry, which were the 3 most commonly identified categories in the preliminary experiment.

Human-Markov Chain Monte Carlo Experiment

Markov chain Monte Carlo (MCMC) is a sampling technique. At each step of the algorithm a proposed state is compared to the current state and one is rejected. The accepted state becomes the current state for the next step. The desired distribution is approximated using the Markov chain formed by the accepted samples. In this experiment, the MCMC analysis was performed by presenting two representations, one the current state in the chain and the other a proposed representation. The participants were asked: 'Which one is the more happy face?' for chain one, 'Which is the more sad face?' for chain two and 'Which is the more angry face?' for chain three. They selected the appropriate choice using a mouse click on a button below the appropriate picture.

Sanborn et al. identified in their human-MCMC analysis of animal representations that decision rule biases could form towards the current state or proposal (Sanborn et al., 2009). This led to unfavorable effects on the outcomes. In order to reduce the effect of such problems the MCMC chains for happy, sad and angry were interleaved. The decision to sample from a particular chain at any point was random and occurred with equal probability for all chains. As such, over many trials an approximately equal number of samples were taken from each category. The current and proposed states were displayed side by side on the screen during the tests.

Each of the MCMC chains was initialized by drawing a set of values from a uniform distribution over the lower 20% of the ranges in Figure 2. The proposed states were drawn from a multivariate Gaussian distribution with the current state as the mean and a diagonal covariance matrix. The standard deviation of the variables was set to 8% of their total range. In preliminary tests this was found to give a proposal acceptance rate from 30-50%. The ranges of the variables for the MCMC test are shown in Figure 2. If a proposal was outside the range then it was rejected and another set of samples taken.

Many studies fail to carefully consider the the impact of the experimental design on the data collected. To mitigate the effect of biases due to the participants not moving the cursor an unbiased coin flip was used to decide whether the current state would appear on the right or the left hand side of the screen. The select buttons were placed close together in order to minimize the effort required to change between the two.

Four participants performed the task. Participants 1, 2 and 3 evaluated 750 pairs over three chains and participant 4 evaluated 350 pairs over three chains, they all took between 30 and 60 minutes to complete the task. Table 1 shows the statistics from the MCMC experiment. The acceptance rate averaged over the whole participant pool was 36.5%.

In carrying out these tests we must be aware of assumptions made that may affect the results. Firstly, the MCMC method assumes that participants accept proposals by a rule that accepts less likely proposals with a certain probability. Secondly, the Markov assumption is that decisions are based on the current pair of stimuli. In such an experiment where the participants were each asked to evaluate a large number of

	No. of Samples			Acceptance %		
	Happy	Sad	Angry	Happy	Sad	Angry
P1	241	267	242	38	43	34
P2	231	271	248	53	41	37
P3	237	244	274	33	38	41
P4	113	114	123	30	20	30

Table 1: Participant's statistics. Number of samples per chain. Acceptance % per chain.

images they may make judgements based on previous images or may become bored with a particular image.

Discrimination Experiment

In this task the participants were presented with a single representation and asked to categorize it as happy, sad or angry. The representations were drawn from uniform distributions over the ranges shown in Figure 2. 750 different stimuli were categorized. The human-MCMC method allows sampling from the probability from the distribution in the the parameter space associated with each category. Thus even in the same context discrimination and MCMC would produce different information (Sanborn et al., 2009).

Results and Discussion

Human-MCMC is a sampling method. The data collected was in four dimensions (outer eyebrow, inner eyebrow, lip separation and lip corner dimensions). The samples obtained from the MCMC tests were mapped to a two dimensional plane that best discriminated between the expression distributions. This was carried out in order to create a visual structure of the expression categories (Olman & Kersten, 2004). The dimensionality reduction was performed using Generalized Discriminant Analysis (GDA) with a Gaussian kernel. GDA is a method of combining features so as to separate classes within the data. Figure 4 shows the resulting chains for all four participants. Using this visualization a judgement was made on how many samples should be rejected in order that the distributions were stationary. The number of samples burned (samples removed from the start of a chain) per chain was 40, leaving the average chain length 213 samples. The GDA was then performed on the samples in four dimensional space that remained after burn-in. Figure 5 shows the resulting samples for the four participants. The average faces for each participant and each category are shown in Figure 6. A mean face for each category, aggregated across the whole participant pool is shown in Figure 5. These faces appear to be reasonable examples of the three categories. This result in part supports the use of the MCMC method.

In these tasks, with only three categories in a limited dimensional space the categories can be separated effectively. However, if there were a great number of categories a Multi-Dimensional Scaling (MDS) representation could be created. We can calculate the similarity of categories by counting the confusions between pairs of stimuli (Rothkopf, 1957; Nosofsky, 1987). A potential downside of MDS is that it does not find an explicit mapping function from the parameter space. Sanborn et al. (Sanborn et al., 2009) use Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) (Hadsell, Chopra, & LeCun, 2006) that does provide an explicit function. This was not tried here but would be worth considering in future work.

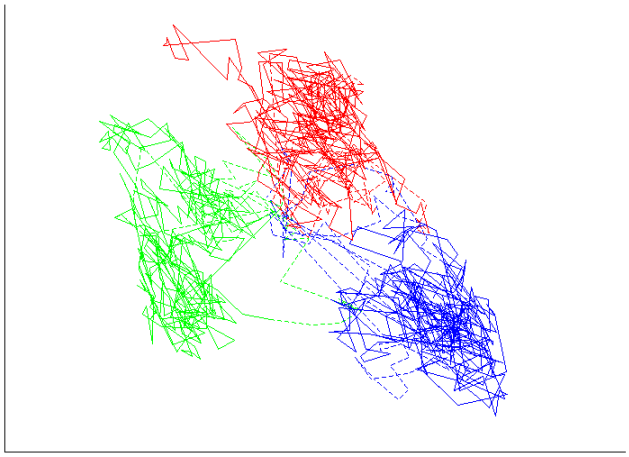


Figure 4: MCMC chains from all participants, before burn-in samples were removed, mapped to the plane that best discriminates between the categories. The dotted lines show the burn-in lengths chosen visually, the first 40 samples from each chain. Chain one - happy (green), chain two - sad (blue), chain three - angry (red).

Within a large parameter space the categories are likely to occupy small subspaces only. As a result a method such as MCMC that allows sampling from the whole parameter space but enables navigation to a particular region is useful compared to a discriminative test that samples from the space randomly.

However, in Figure 6 we compare the mean faces from the MCMC task and the discrimination task for one participant. In both cases the mean representations are reasonable examples. This suggests that the advantage of the MCMC method is not seen in this four dimensional space with the ranges described. As we increase the ranges and the number of degrees of freedom the space will increase greatly in size and it is likely that the benefit of the MCMC method will become apparent.

The discrimination experiment stimuli were categorized using the distributions found from the MCMC results. A naive Bayes classifier with Gaussian kernal was fitted to the four dimensional human-MCMC samples. Using this model the most likely label for each of the discrimination stimuli was chosen. These labels were then compared to the human responses.

The model matched the human identification of the stimuli in 70.1% of cases. This is much better than chance at 33%. The error is likely to be due to the fact that the discrimination

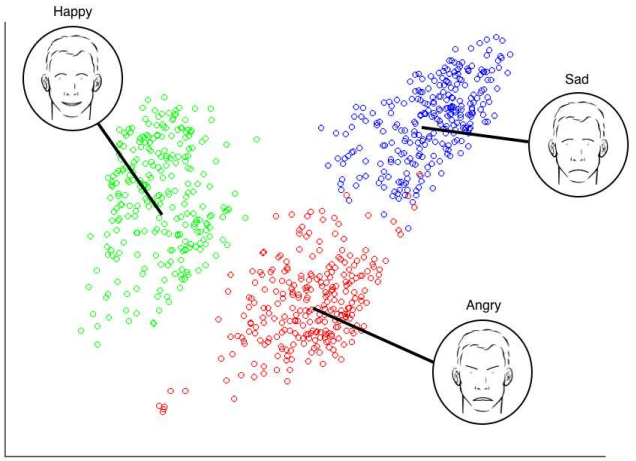


Figure 5: Scatter plot of samples from the four participants, after burn-in, mapped to the plane that best discriminates between the categories. The average face for each category is shown. Samples from: chain one - happy (green), chain two - sad (blue), chain three - angry (red).

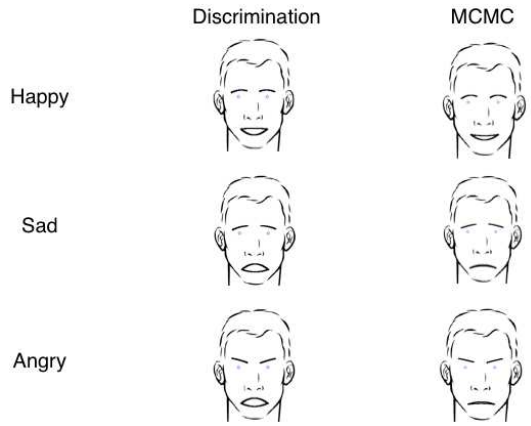


Figure 6: Comparison of mean faces for one participant in the discrimination task and MCMC task.

stimuli were generated from uniform distributions over the ranges. As such, many were far from the samples generated by the MCMC method. It is likely that many of the discrimination stimuli would not have been classified as any of the three categories if there had been other alternatives. Testing on results of a discrimination task with an 'other' option may produce even stronger performance.

For each of the dimensions the probability distributions for each category were estimated from the human-MCMC samples. The samples were separated into 25 equal size bins. Gaussian Process Regression (GPR)[1] was then used to approximate the distributions. A squared exponential (SE) covariance summed with an independent noise function was used. This does not make the assumption of an underlying structure but rather assumes the function is infinitely smooth. The characteristic noise scale and signal variance were set to one and the noise variance also to one. The hyper-parameters could be adjusted further. However, for a qualitative representation of the distributions given by the data these were reasonable choices.

Figure 7 shows the estimated density plots for each dimension after aggregating the data from all participants. It shows that in some dimensions (lip separation, inner eyebrow) none of the categories are significantly distinguished from the other two. However in the cases of the outer eyebrow and lip corner dimensions one of the categories was distinct. For the outer eyebrow dimension the distribution for anger is significantly different from the distributions for happy and sad. For the lip corner it is happy that is more distinguishable. The sad category distributions were not significantly different from both of the other two in any of the cases.

There are certain assumptions and limitations within the experiment that must be noted. As described above, when a proposal was outside the range set it was automatically rejected. In certain cases this rule was enforced and the distribution met one of the boundaries. This is not necessarily a negative point as the ranges restricted the participants to move within a space of reasonably natural expressions. We see from Figure 7 that for the inner eyebrows and lip corners the distributions did push up against the boundaries to a certain extent. This is something to consider in future work.

We should also note some general comments about aspects of the experimental set up. We must consider the impact of participants becoming bored during the experiment and selecting their response arbitrarily. Many samples were required in order to generate stationary distributions. Ways of minimizing the effects of boredom should be considered in future.

Conclusions

This paper demonstrates that human-MCMC methods can be used to gain insight into facial expression categorization using simple cartoon representations. We demonstrated that from 750 samples over three categories the method provides reasonable mean representations for each of the categories and reasonable distributions. By using GDA we were able to map the four dimensional points to a plane and after burn-in reveal three categories. The sad and angry chain samples were not separable in two dimensions. The happy chain samples were separable.

We also show estimates of the distributions for each of the categories along each of the four dimensions. This reveals that for the features tested the lip corner is the best discriminator for happy expressions and the outer eyebrow the strongest for angry expressions. The sad distributions were not distinguishable from both happy and angry distributions in any of the cases.

The mean faces generated by the human-MCMC and discrimination tasks were both reasonable and neither signifi-

[1] Rasmussen and William's GPML toolbox was used for this task.

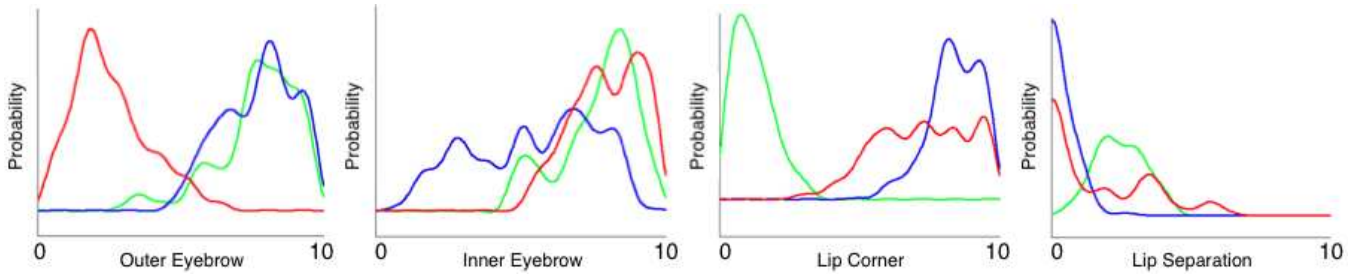


Figure 7: Density estimates for each of the four parameters aggregated over all the participants. The parameter dimensions correspond to the ranges shown in Figure 2. Chain one - happy (green), chain two - sad (blue), chain three - angry (red).

cantly more realistic than the other.

A naive Bayes classifier trained on the aggregated samples generated from the MCMC task performed strongly predicting over 70% of the human labels in the discrimination task correctly.

Further Work

This paper describes the first investigation evaluating human facial expression categorization using a human-MCMC method. It justifies a basis for applying a human-MCMC method for exploring people's representations of facial expressions. Griesser et al. (Griesser et al., 2007) demonstrate the use of detailed computer avatars that can realistically demonstrate skin texture changes as well as facial actions. This type of stimuli could be used in order to seriously investigate a wider range of categories. It would also allow more detailed investigation of the degree to which specific dimensions allow discrimination in terms of affect.

Sanborn et al. (Sanborn et al., 2009) suggest that the human-MCMC method may be used to test models of categorization. Prototype models produce unimodal distributions. Exemplar models are more flexible. As such it is difficult to establish whether a category distribution more closely resembles a prototype or exemplar model in many cases but rather we can test whether a distribution has properties that rule out a prototype model (Sanborn & Griffiths, 2008; Sanborn et al., 2009).

Acknowledgments

This work was funded by the MIT Media Lab Consortium.

References

Darwin, C., Ekman, P., & Prodger, P. (2002). *The expression of the emotions in man and animals*. Oxford University Press, USA.

Dopkins, S., & Gleason, T. (1997). Comparing exemplar and prototype models of categorization. *Canadian Journal of Experimental Psychology*, *51*(3), 212–230.

Ekman, P., & Friesen, W. V. (1978). *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists.

El Kaliouby, R., & Robinson, P. (2005). Generalization of a vision-based computational model of mind-reading. *Proceedings of First International Conference on Affective Computing and Intelligent Interaction*, 582–589.

Griesser, R., Cunningham, D., Wallraven, C., & Bulthoff, H. (2007). Psychophysical investigation of facial expressions using computer animated faces. In *Proceedings of the 4th symposium on applied perception in graphics and visualization* (p. 18).

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proc. computer vision and pattern recognition conference (cvpr06)*.

Neal, R. (1993). *Probabilistic inference using Markov chain Monte Carlo methods*. Citeseer.

Nosofsky, R. (1986). Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, *115*(1), 39–57.

Nosofsky, R. (1987). Attention and learning processes in the identification and categorization of integral stimuli. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *13*(1), 87–108.

Nosofsky, R., & Palmeri, T. (1997). An exemplar-based random walk model of speeded classification. *Psychological Review*, *104*(2), 266–299.

Olman, C., & Kersten, D. (2004). Classification objects, ideal observers & generative models. *Cognitive Science*, *28*(2), 227–239.

Padgett, C. (1998). *A neural network model for facial affect classification* (Tech. Rep.).

Padgett, C., & Cottrell, G. (1997). Representing face images for emotion classification. *Advances in neural information processing systems*, 894–900.

Rothkopf, E. (1957). A measure of stimulus similarity and errors in some paired-associate learning tasks. *Journal of Experimental Psychology*, *53*(2), 94–101.

Sanborn, A., & Griffiths, T. (2008). Markov chain Monte Carlo with people. *Advances in neural information processing systems*, *20*.

Sanborn, A., Griffiths, T., & Shiffrin, R. (2009). Uncovering mental representations with Markov chain Monte Carlo. *Cognitive Psychology*.

Developing a Model of Cognitive Lockup for User Interface Engineering

Tina Mioch (Tina.Mioch@tno.nl)[1]
Rosemarijn Looije (Rosemarijn.Looije@tno.nl)[1]
Mark Neerincx (Mark.Neerincx@tno.nl)[1,2]

[1]TNO Human Factors, P.O. Box 23
3769 ZG Soesterberg, The Netherlands

[2]Delft University of Technology Man–Machine Interaction Group,
Mekelweg 4, 2628 CD, Delft, The Netherlands

Abstract

This paper presents the development of a cognitive model of cognitive lockup: the tendency of humans to deal with disturbances sequentially, possibly overseeing crucial data from unattended resources so that serious task failures can appear—e.g., in a cockpit or control centre. The proposed model should support the design and evaluation of user interfaces that prevent such failures, being used outside the academic community. Based on the practical cognitive task load theory of Neerincx (2003), this model distinguishes time pressure and number of tasks-to-do as two factors that increase task switch costs and the corresponding risk of cognitive lock-up. The CASCaS architecture proved to fit best with the requirements to incorporate these factors and to support the UI engineering process.

Keywords: cognitive lockup; cognitive modeling; cognitive task load model; cognitive architectures; user interface engineering.

Introduction

Aircraft pilots are faced with a complex traffic environment. Cockpit automation and support systems help to reduce this complexity. Currently, a lot of research is done to improve the onboard management of flight trajectories and the negotiation of trajectory changes with Air Traffic Control. During the flight, many factors may induce changes to the original flight plan, e.g. bad weather, traffic conflicts, or runway changes. Safe operation of aircrafts is based on normative flight procedures (standard operating procedures) and rules of good airmanship, which we will refer to as normative activities. We define pilot errors as deviations from normative activities.

In the past, several cognitive explanations and theories have been proposed to understand why pilots deviate from normative activities (e.g. Dekker (2003)). The European project HUMAN, in which the research described in this paper is done, strives to pave a way of making this knowledge readily available to designers of new cockpit systems. We intend to achieve this by means of a valid executable flight crew model which incorporates cognitive error-producing mechanisms leading to deviations from normative activities. The model interacts with models of cockpit systems in a virtual simulation environment to predict deviations and its potential consequences on the safety of flight. The ultimate objective of HUMAN is to apply this model to analyze human errors and support error prediction in ways that are usable and practical for human-centered design of systems operating in complex cockpit environments.

At the initial stage of HUMAN we performed questionnaire interviews with pilots and human factor experts based on a literature survey of error-producing mechanisms. We identified cognitive lockup to be among the most relevant mechanisms for modern and future cockpit human machine interfaces. We take the definition of cognitive lockup from Moray and Rotenberg (1989) who define the term 'cognitive lockup' as the tendency of operators to deal with disturbances sequentially. This has as a result that operators focus on a subpart of a system and ignore the rest of it (Meij, 2004).

In this paper, we discuss factors that can cause cognitive lockup and an architecture of a cognitive model that can be used to help prevent lockup failures during User Interface engineering.

Cognitive Lockup

Previous Research

As the definition from Moray and Rotenberg (1989) shows, cognitive lockup does not occur when people can perform all their tasks consecutively. Therefore they designed a task where this was not possible. Participants were asked to supervise a simulated thermal hydraulic system that consisted of four subsystems. In one scenario they needed only to focus on one fault in one of the subsystems. In another scenario a first fault was followed by a second fault in a different subsystem, which occurred before the participant could have handled the first fault. It was shown that participants shifted attention much later to the second fault then they did to the first fault. Moray and Rotenberg attributed this to limited information processing capacities. In another study that demonstrated cognitive lockup (Kerstholt et al, 1996), participants had to supervise four dynamic subsystems and deal with disturbances. The system included the option to stabilize a subsystem in which additional faults occurred, with which participants acknowledged their understanding of the development of a

disturbance over time. Most participants did not use this option and handled the disturbances sequentially.

Cognitive lockup as a phenomenon is related to the rise of automation, but the tendency to proceed with the current task is not new. Meij (2004) investigated cognitive lockup in relation to planning, task-switching and decision making. He found that both prior investments into a task as the time that is needed to complete the task increases the probability of cognitive lockup. No support was found for refrainment of monitoring (a second fire was detected, but not tended to before the first fire was solved), too optimistic scenarios, and lack of resources (the complexity of the first task did not influence the degree of cognitive lockup).

Cognitive Task Load Model

A model that specifies core aspects of cognitive lockup is the cognitive task load (CTL) model of Neerincx (2003). The development of this model is driven by the need for limited and practical theories and models on human cognition to take validation of the theories and models out the laboratory and into the real world, where the environment is more dynamic.

The CTL-model describes load in terms of three behavioral factors: time pressure, level of information processing and number of task set switches (see Figure 1).

Time Pressure The time pressure is dependent on the scenario and the actions of tasks. The scenario provides information on the number of tasks due to events and the actions that are called upon by the tasks can take a long or a short time to handle. A standard measure for the time pressure is:

$$\textit{Time pressure} = \frac{\textit{time required for tasks}}{\textit{time available for tasks}}$$

Humans reach overload when the time pressure is more than 70-80% (Beevis et al., 1994).

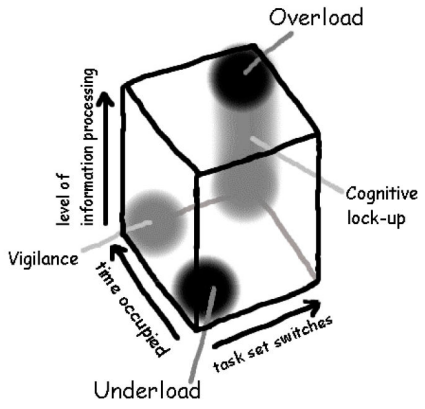


Figure 1: CTL model, with the three dimensions task set switches, level of information processing, and time occupied (time pressure).

Level of Information Processing The level of information processing factor is measured as the percentage of knowledge-based actions using the Skill-Rule-Knowledge framework from Rasmussen (1986). Input information that can be processed at skill level (e.g. when you touch something hot with your hand, you immediately react by removing your hand from the heat source) is not cognitively demanding. When input information triggers a routine consisting of rules (i.e. procedures with rules of the type "if <*event/state*> then <*actions*>") it takes some cognitive capacities to resolve the if/then, but the rest of the procedure is quite automatic. Cognitive demanding are the situations where there is problem analysis needed on the input information and knowledge to reason about it, this can have a large influence on the working memory.

Rasmussen's framework corresponds to the cognitive theory of skill acquisition of Anderson (1982) that distinguishes three memory representations: cognitive, associative and autonomous. These three levels are linked to different memory representations; declarative, procedural and implicit.

Task Set Switches To take into account situations where people have to perform different tasks that appeal to different sources of human knowledge and different objects in the environment, the CTL-model comprises the task set switches factor. A task set contains both the human resources and environmental objects with momentary states, which are involved in the task performance. A switch occurs when the applicable task knowledge on the operating and environment level change. A task set can thus be seen as a goal that is comprised of several (sub-)tasks.

Rubinstein, Meyer and Evans (2001) distinguish two types of task switching: task switching in successive tasks and task switching in concurrent tasks. With successive tasks the first task is responded to and finished before the second task is presented. Concurrent tasks on the other hand are tasks where the second task is presented before the first task has been finished. We are only interested in concurrent tasks, because a pilot usually has multiple concurrent tasks that can be executed, e.g. monitoring different interfaces in the cockpit. Successive task switching studies show that task switching takes time (Jersild, 1927, Rogers & Monsell, 1995). In concurrent task switching studies (De Jong, 1995; Schumacher et al., 1999), it is observed that people are unable to deal with multiple tasks. They postpone the second task until the first task is completed. In these experiments the second task is not of such importance that it should be handled immediately, but in real life situations not handling the second task before finishing the first can cause life threatening situations (e.g. the crash of flight 401 of Eastern Air Lines in 1972 (NTSB, 1973)). Tasks can be interrupted, but with every switch time and effort is needed to do context acquisition to bring the environment information up-to-date (Olsen & Goodrich, 2003).

In the CTL-model, the task set switches can be seen as the number of task set switches possible at a particular moment

in time. This number comes thus forth from the environment and the situation a person is in.

Cognitive Lockup in the CTL Model The three factors of the CTL model are interrelated (Figure 1). Cognitive lockup is independent of information processing level, but does occur when both time pressure and number of task set switches is high. That the information of processing level is not of importance seems counterintuitive, but in an experiment of Meij (2004) (experiment 2) this is supported. In the experiment of Meij, participants were asked to monitor for fires on a ship. When a fire was detected it had to be diagnosed on both priority and treatment. Two fires could exist simultaneously and the participant had to decide which fire to fight. The complexity of this task was varied by making the diagnosis of priority and treatment harder and by varying the moment of introduction of the second fire (e.g. after diagnosis of the first fire or during diagnosis). The data showed that an increasing level of complexity had no influence on when the second fire was detected.

Pilots and Cognitive Lockup

The most famous example of cognitive lockup comes from the aviation domain. In 1972 a plane from Eastern Air Lines, flight 401, crashes. During the landing the pilot is warned about a problem with the landing gear. He cancels the landing and sets the plane in autopilot so that he can solve the problem. Unfortunately, due to his occupancy with the landing gear, the pilot missed the warning signals (alarms and air-traffic control) about decreasing altitude, and the plane crashed (NTSB, 1973).

Modeling of Cognitive Lockup

Cognitive Architecture

Cognitive architectures were established in the early eighties as research tools to unify psychological models of particular cognitive processes (Newell, 1994). These early models only dealt with laboratory tasks in non-dynamic environments (Anderson, 1993; Newell, Rosenbloom, & Laird, 1989). Furthermore, they neglected processes such as multitasking, perception and motor control that are essential for predicting human interaction with complex systems in highly dynamic environments like the air traffic environment addressed in HUMAN with the AFMS target system. Models such as ACT-R and SOAR have been extended in this direction (Anderson et al., 2004; Wray & Jones, 2005) but still have their main focus on processes suitable for static, non-interruptive environments. Below we provide a short overview of the requirements we have for the cognitive model and how these requirements are met by ACT-R 6.1.4, SOAR 9.3.0 and EPIC. Note that we evaluate the requirements only for these versions. ACT-R and SOAR are under constant development and requirements that are not met at the moment might be met in future versions.

The first requirement is that the cognitive model should support multitasking. The three best known cognitive architectures all support a form of multitasking; ACT-R with threading (e.g. Salvucci & Taatgen, 2008), to SOAR (Newell, Rosenbloom, & Laird, 1989) and EPIC (Meyer & Kieras, 1997) it is inherent to the architecture. Secondly, because we want to test interfaces there is a need for perception and motor action abilities. This is inherent to EPIC (Meyer & Kieras, 1997), ACT-R is able to do this since ACT-R/PM (Byrne, 2001), and SOAR cannot do this without coupling with EPIC, although since SOAR 9 there is a vision module (Laird, 2008). All three need interface coupling with a model of the interface (e.g. developed with SegMan (Amant et al., 2005)). Thirdly, the model should be able to learn, SOAR and ACT-R are able to learn, but EPIC is not. Fourthly, we want an explicit Skills-Rules-Knowledge separation (Rasmussen, 1983) to make it easier for users to choose a level on which they want to work and to make it more clear for end users where errors came from. When it is from rules (procedures), adapting procedures can be a solution, when it comes from the knowledge level the solution can be more difficult, because the problems that arise from this level are inherent to people. Finally, it is very important that non-expert users can use the cognitive model in the design and testing process of interfaces. With none of the three discussed cognitive architectures this is possible, because they all require a high level of knowledge of the model, in addition to programming skills, before being able to adapt them to a certain domain or interface.

In the following, we describe shortly the architecture used in the HUMAN project. We choose to describe the architecture to show that our theory of cognitive lockup is embedded in a broader concept. However, this description will only be short and will not go into (implementation) details, as for the theory of cognitive lockup, these details are not necessary.

The cognitive architecture CASCaS (Cognitive Architecture for Safety Critical Task Simulation) is used to model the cognitive process described in the previous section. For a more detailed description of the CASCaS architecture see Lüdtke et al. (2009). CASCaS has multitasking abilities, has a perception and motor module, is able to learn (e.g. production compilation), has a skills, a rules (associative layer) and a knowledge (cognitive layer) based level. Finally, only when you really want to change something of the architecture programming skills are necessary. Otherwise there are editors for the procedures (domain knowledge) and for the interface description. The procedure editor (Frische et al., 2009) can be used by any domain expert, which has been shown by an informal review that was performed by one of the end user partners in the HUMAN project. And UsiXML (Limbourg et al., 2005) which describes the interface in a way that it can be used by the model can automatically transfer HTML pages into the right format, has a graphical editor so that interface designers can use tools that are similar to what they know and XML programming is also possible. UsiXML is developed by human factor experts at the Belgian Laboratory of Computer-Human Interaction (BCHI).

The core of CASCaS is formed by the layered knowledge processing component that contains the associative and the cognitive layer.

A task that is encountered for the first time is processed on the cognitive level with maximal cognitive effort. This processing is goal driven; alternative plans to reach a goal are evaluated usually through mental simulation, and finally one plan is selected to be executed. With some experience, the associative level is used, where solutions are stored that proved to be successful; the pilot has for example learned how to handle the cockpit systems in specific flight scenarios. According to Rasmussen (1983), processing is controlled by a set of rules that have to be retrieved and then executed in the appropriate context. On the autonomous level routine behavior emerges that is applied without conscious thought, e.g. manually maneuvering an aircraft. When solving a task, people tend to apply a solution on the lower levels first, and only revert to solutions on higher levels when lower-level ones are not available (Rasmussen, 1983) or when the situation requires very careful handling due to unusual and safety relevant conditions.

The associative layer selects and executes rules from long-term memory. It is modeled as a production system. Characteristic for such systems is a serial cognitive cycle for processing rules: A goal is selected from the set of active goals (Phase 1), all rules containing the selected goal in their goal-part are collected and a short-term memory retrieval of all state variables in the Boolean conditions of the collected rules is performed (Phase 2). If a variable is absent in memory, a dedicated percept action is fired and sent to the percept component to perceive the value from the environment and to write it into the short-term memory. After all variables have been retrieved, one of the collected rules is selected by evaluating the conditions (Phase 3). Finally the selected rule is fired (Phase 4), which means that the motor and percept actions are sent to the motor and percept component respectively and the sub-goals are added to the set of active goals. This cycle is started when a Boolean condition of a reactive rule is true. In Phase 2 reactive rules may be added to the set of collected rules if new values for the variables contained in the State-Part have been added to the memory component (by the percept component). In Phase 3, reactive rules are always preferred to non-reactive rules. The cognitive cycle is iterated until no more rules are applicable.

The cognitive layer reasons about the current situation and makes decisions based on this reasoning. Consequently, we differentiate between a decision-making module, a module for task execution and a module for interpreting perceived knowledge (sign-symbol translator). In the following, we will describe the decision-making module in more detail, as it is relevant to modeling cognitive lockup. For more information on the cognitive layer see Lüdtke et al. (2009).

The decision-making module determines which goal is executed. Goals have priorities, which depend on several factors: goals have a static priority value that is set by a domain expert. In addition, priorities of goals increase over time if not executed. Implicitly, temporal deadlines are modeled in this way. If, while executing a goal, another goal has a distinctively higher priority than the current one, the execution of the current goal is stopped and the new goal is attended to. This decision depends on the priorities of the goals and is extended by the parameter *Task Switching Costs* (TSC), which determines the difference the priorities need to have to halt the execution of a goal to select a different goal to be executed. TSCs are described extensively in literature (e.g. Jersild, (1927); Rogers & Monsell (1995)). The higher the TSC is, the higher the priority of another goal needs to be to switch to that goal. To determine whether a goal should be interrupted and a different goal should be executed, the TSC is added to the current task priority. Only if a priority of another active goal is above this threshold, this other goal is chosen to be executed. For a visualization of the goals see Figure 2.

Figure 2: Visualization of the goals on the cognitive layer. Dark gray and green goals are active. The framed goal is currently executed. The yellow staff represents the additional task switch costs.

Cognitive Lockup Model

In this section we describe how cognitive lockup is modeled in the cognitive architecture described above. We model cognitive lockup on the cognitive layer. The main reason for this is that, as described above, on the cognitive layer we have an explicit goal decision mechanism in which cognitive lockup can easily be integrated. However, this can be extended to the associative layer, as the principles explained below are generally applicable to the goals of the associative layer as well.

Time Pressure As described in Neerincx (2003), the time pressure for a person plays an important role for cognitive lockup. If a person has a value for the time pressure of more than 0.75 (Neerincx, 2007), the task switch cost increases. In general, this factor depends both on the time pressure of the associative and cognitive layer. However, to simplify matters, we will model this temporarily only related to the cognitive layer, but will extend the concept later to the associative layer. As written above, the formula that we use is the following:

$$\textit{Time pressure} = \frac{\textit{time required for tasks}}{\textit{time available for tasks}}$$

For example, if we have a task that can be done in 25 seconds and we have 100 seconds before it needs to be finished, the predicted time pressure is 0.25.

The time required for a task is the time needed for cognitively processing the task. This knowledge comes both from the analysis of normative behavior, i.e. discussions with experts that give an indication of the time a task takes, in addition to cognitive theories on which the cognitive architecture is based (e.g. (Anderson, 1993; Kieras & Meyer, 1997)).

Modeling the time that is available for a task is quite complex. For some tasks this knowledge is given in the normative behavior. For example, a pilot needs to have set the flaps before reaching the final approach phase. The time that is available for a task can thus be calculated by the knowledge of the current task, and a prediction of when the approach phase begins, which can be gained from the environment. For other tasks, it is not that easy to know the time that is available to execute it. For example, for a monitoring task, there is no standard deadline at which monitoring has to be finished. However, the time pressure will slowly increase, without having a clear deadline of the task, as there is no unlimited time to execute any task.

Thus, for each task, it has to be evaluated whether the time pressure can be based on a calculation of elements of task knowledge and the environmental input, or whether it has to be given a general estimate.

The time pressure is inherent to each goal as it only takes aspects of the individual goal into account, but is dynamic as the time until it needs to be finished is constantly diminishing. We decided that this calculation is done each 50 ms, which is the cycle time of our architecture.

Level of Information Processing As described above, the level of information processing does not play a relevant role for cognitive lockup. This factor is not taken into account in the model of task switching costs.

Task Set Switches As described above, task set switches are defined as possible goal switches at a given moment. The number of task sets is modeled as the number of goals that are active at the moment. Temporarily, we only look at goals in the cognitive layer.

The value of the task set switches is thus the number of active goals in the environment. We assume that the model always has activated all possible tasks that play a role at the moment in the environment and are needed to handle the current situation.

The Model

Above, we have described different aspects that increase the probability of cognitive lockup. In our model, this is simulated by increasing the task switch costs (TSCs) of the goal that at that moment is processed. The TSC determines the difference that the priorities need to have to halt the execution of a goal to select a different goal to be executed. The TSC depends on the number of goals that at that moment is also active and could be selected to be processed, and on the time to spare to execute the current goal. The TSC is higher when there is high time pressure. Furthermore, the higher the number of active goals is (i.e. the possible task set switches) the higher are the costs to switch to another goal. The following formula determines the TSC:

$$TSC = StartTSC * (Time\ pressure + Task\ set\ switches),$$
$$\text{with } Time\ pressure = 0 \text{ if } Time\ pressure < 0.75.$$

This means that the task switch costs depend on a start value, which is a constant, and the sum of the two factors of the time pressure and the task set switches.

As at each moment if there are active goals, at least one goal is selected and executed, the task set switches parameter is always at least 1. If there is only one goal, and the task pressure is not high, the TSC is equal to the constant start value. The moment there are several active goals or the time pressure for the currently selected goal is above the threshold of 0.75, the TSC is increased.

Conclusion

This paper presented the development of a cognitive model of cognitive lockup: the tendency of humans to deal with disturbances sequentially, possibly overseeing crucial data from unattended resources so that serious task failures can appear—e.g., in a cockpit or control centre. The model is based on real life examples of cognitive lockup and the psychological theories that are derived from these examples, and laboratory experiments. It distinguishes time pressure and number of tasks-to-do as two factors that increase task switch costs and the corresponding risk of cognitive lockup. A heightened task switch cost leads to less task switching, even when another task has a higher priority, as the difference between the priorities needs to be higher.

The proposed model should support the design and evaluation of user interfaces that prevent such failures, being used outside the academic community. The CASCaS architecture proved to best fit with the requirements to incorporate these factors and to support the UI engineering process.

At the moment, we calculate the time pressure as a value inherent to the individual goal. The interdependencies between the timing of several goals will be taken into account in the next version of the cognitive model (i.e., several tasks might in themselves not have a high time pressure, but might together be time-critical, as all of them might need to be finished before all of them can be executed).

The values for the parameters we have chosen for our cognitive model are mainly based on literature, and are currently being evaluated in both laboratory experiments and realistic simulator experiments. In this way, we refine and validate the model, improving its plausibility and predictions about the behavior of pilots. Application of the model will provide user interfaces and procedures that reduce the risks for lockup errors. Due to the cognitive plausibility, we predict that the model can also be used in other domains without substantial changes.

Acknowledgments

The work described in this paper is funded by the European Commission in the 7th Framework Programme, Transportation under the number FP7 – 211988.

References

Amant, R.S. and Riedl, M.O. and Ritter, F.E. and Reifers, A. (2005). Image processing in cognitive models with SegMan. *Proceedings of HCI International 2005*.

Anderson, J. (1982). Acquisition of cognitive skill. *Psychological review*, 89(4), 369–406.

Anderson, J. (1993). *Rules of mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4), 1036–1060.

Beevis, D., Bost, R., Döring, B., Nordø, E., Oberman, F., Papin, J., et al. (1994). *Analysis techniques for man-machine system design*. AC/243(Panel 8) TR/7 Vol, 2.

Byrne, M.D. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human Computer Studies*, 55(1), 41-84.

De Jong, R. (1995). The role of preparation in overlapping task performance. *The Quarterly journal of experimental psychology. A, Human experimental psychology*, 48(1), 2.

Dekker, S. (2003). Failure to adapt or adaptions that fail. *Applied Ergonomics*, 34(3), 233–238.

Frische, F. and Mistrzyk, T. and Lüdtke, A. (2009). Detection of Pilot Errors in Data by Combining Task Modeling and Model Checking. *Human-Computer Interaction--INTERACT 2009*, 528-531.

Jersild, A. (1927). Mental set and shift. *Archives of Psychology*. Vol, 14(89), 81.

Kerstholt, J., Passenier, P., Houttuin, K., & Schuffel, H. (1996). The effect of a priori probability and complexity on decision making in a supervisory control task. *Human Factors*, 38(1), 65–78.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the epic architecture for cognition and performance with application to human-computer interaction. *Hum.-Comput. Interact.*, 12(4), 391–438.

Laird, J.E. (2008). Extending the Soar cognitive architecture, Artificial General Intelligence 2008: Proceedings of the First AGI Conference.

Limbourg, Q. and Vanderdonckt, J. and Michotte, B. and Bouillon, L. and López-Jaquero, V., (2005). *Engineering Human Computer Interaction and Interactive Systems*, 200-220.

Lüdtke, A., Osterloh, J.-P., Mioch, T., Rister, F., & Looije, R. (2009, September 23–25). Cognitive modelling of pilot errors and error recovery in flight management tasks. In P. Palanque, J. Vanderdonckt, & M. Winckler (Eds.), *Human error, safety and systems development*, 7th ifip wg 13.5 working conference, hessd 2009 (Vol. 5962). Brussels, Belgium: Springer.

Meij, G. (2004). *Sticking to plans: capacity limitation or decision-making bias?* Doctoral dissertation, Department of Psychology, University of Amsterdam, Amsterdam.

Meyer, D.E. and Kieras, D.E. (1997). A computational theory of executive cognitive processes and multiple-task performance: I. Basic mechanisms. *Psychological Review*, 104 (1), 3-65.

Moray, N., & Rotenberg, I. (1989). Fault management in process control: eye movements and action. *Ergonomics*, 32(11), 1319–1342.

NTSB (1973). Eastern Airlines l-1011, Miami, Florida, December, 29, 1972 (Tech. Rep. No. NTSB-AAR-73-14). Washington, DC: National Transportation Safety Board (NTSB).

Neerincx, M. (2003). Cognitive modelling of pilot errors and error recovery in flight management tasks. In E. Hollnagel (Ed.), *Handbook of cognitive task design* (pp. 283–306). CRC.

Neerincx, M. (2007). Modelling cognitive and affective load for the design of human-machine collaboration. *Lecture Notes in Computer Science*, 4562, 568.

Newell, A. (1994). *Unified theories of cognition*. Harvard Univ Pr.

Newell, A., Rosenbloom, P., & Laird, J. (1989). Symbolic architectures for cognition. In M. Posner (Eds.), *Foundations of cognitive science* (pp. 93–131). Cambridge, MA: MIT Press.

Olsen, D., & Goodrich, M. (2003). Metrics for evaluating human-robot interactions. In *Proceedings of permis* (Vol. 2003).

Rasmussen, J. (1983). Skills, rules, knowledge: Signals, signs and symbols and other distinctions in human performance models. *IEEE Transactions: Systems, Man and Cybernetics*, SMC-13(3), 257–266.

Rasmussen, J. (1986). *Information processing and human machine interaction: An approach to cognitive engineering*. Elsevier Science Inc. New York, NY, USA.

Rogers, R., & Monsell, S. (1995). Costs of a predictable switch between simple cognitive tasks. *Journal of Experimental Psychology-General*, 124(2), 207–230.

Rubinstein, J., Meyer, D., & Evans, J. (2001). Executive control of cognitive processes in task switching. *Journal of Experimental Psychology Human Perception and Performance*, 27(4), 763–797.

Salvucci, D.D. and Taatgen, N.A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review,* 115(1), 101-130.

Schumacher, E., Lauber, E., Glas, J., Zurbriggen, E., Gmeindl, L., Kieras, D., et al. (1999). Concurrent response-selection processes in dual-task performance: Evidence for adaptive executive control of task scheduling. *Journal of Experimental Psychology: Human Perception and Performance*, 25, 791–814.

Wray, R., & Jones, R. (2005). An introduction to soar as an agent architecture. In R. Sun (Ed.), *Cognition and multiagent interaction: From cognitive modeling to social simulation* (pp. 53–78). Cambridge University Press.

Checking the Brain Mapping Hypothesis: Predicting and Validating BOLD Curves for a Complex Task Using ACT-R

Claus Möbus (claus.moebus@uni-oldenburg.de)
Jan Charles Lenk (jan.lenk@uni-oldenburg.de)
Arno Claassen (arno.claassen@ uni-oldenburg.de)
Department of Computing Science, University of Oldenburg
D-26121 Oldenburg, Germany

Jale Özyurt (jale.oezyurt@uni-oldenburg.de)
Christiane M. Thiel (christiane.thiel@uni-oldenburg.de)
Department of Psychology, University of Oldenburg
D-26121 Oldenburg, Germany

Abstract

John R. Anderson proposed a correspondence between ACT-R modules and brain regions (Brain Mapping Hypothesis). Using a paradigm requiring rule-based matching of chemical structures (pseudo formulae) with their respective names, we compared ACT-R-generated blood-oxygen-level dependent (BOLD) signal curves with BOLD curves obtained from functional Magnetic Resonance Imaging (fMRI) scans. We found significant correlations between ACT-R generated and human BOLD curves for sensory and motor modules and regions in particular, whereas a lack of significant results was observed for mappings between internal modules and regions. This result was ascribed to the fact that in contrast to Anderson's studies, our subjects were not urged to follow a single strategy. Instead the task allowed them to construct their personal strategy within a constraint-based strategy space. Accordingly, the mapping hypothesis was tested strategy-specific. As subjects are generally not able to reliably identify their own in a retrospective manner, we used Response-Time (RT) data in combination with a Bayesian Belief Net to identify personal problem solving strategies.

Keywords: ACT-R; BOLD signal prediction, brain-mapping hypothesis

Introduction

The ACT-R architecture (Anderson, 2004) provides a set of modules with sensory, motor, and internal functions. Anderson (2007a; Anderson, et al., 2008b) proposes a neurophysiologic analogy and postulates a mapping between these modules and brain regions (Table 1). For instance, the Procedural module is mapped onto the basal ganglia, while the Declarative module is mapped around the inferior frontal sulcus. The ACT-R 6.0 implementation provides a set of tools which directly predict BOLD signals for these brain regions. Indeed, Anderson has "[..] defined these regions once and for all and use them over and over again in predicting different experiments" (2007b).

Several studies were conducted by Anderson et al. in order to empirically validate the mapping hypothesis. These included experiments from various domains, like algebraic problem solving (Danker & Anderson, 2007; Stocco & Anderson, 2008), associative learning (Anderson et al., 2008a) or insight problems (Anderson et al., 2009). One particular feature in common of all these experiments was the fact that participants had to employ the same problem solving strategy on all tasks.

The empirical validation of the mapping hypothesis is among the research goals of our multidisciplinary research project (see Section Acknowledgements). While also the effects of affective and informative feedback on learning are being studied (Özyurt, Rietze, & Thiel, 2008) an accompanying fMRI study offers us the possibility to compare BOLD signal predictions generated from strategy-specific ACT-R models with BOLD signals obtained from actual fMRI scans.

Table 1: ACT-R module/regions mappings according to Anderson (2007a) with positions in Talairach coordinate and dimensions (D, W, H) in voxels

Module	Region	X	Y	Z	D	W	H
Declarative	Prefrontal	±40	21	21	5	5	4
Imaginal	Parietal	±23	-64	34	5	5	4
Manual	Motor	±41	-20	50	5	5	4
Goal	ACC	±5	10	38	5	3	4
Procedural	Caudate	±15	9	2	4	4	4
Visual	Fusiform	±42	-61	-9	5	5	4
Aural	Auditory	±46	-22	9	5	5	4
Vocal	Motor	±43	-14	33	5	5	4

Results of the present study suggest a further refinement of our modeling methods. In contrast to the experiments described by Anderson et. al. (2008a; Danker & Anderson, 2007; Stocco & Anderson, 2008), the tasks in our experimental setting were far more complex; because in order to solve these tasks, participants were free to choose their *personal* strategies. Because different strategies lead to different predictions of brain region activation, we had to model these different strategies and identify the chosen subject-specific strategy *without* using fMRI data (Möbus & Lenk, 2009). We would work unduly in favor of the mapping hypothesis if we would assign subjects to strategies according to similarity of their BOLD curves with the strategy-specific ACT-R-BOLD curves.

Experiment

All participants were lower-grade schoolchildren with ages ranging from 11 to 13. The exercises which the children had to solve came from the domain of the chemical formula language (Heuer & Parchmann, 2008), which is generally unknown to children of that age. However, instead of real-world chemical elements, pseudo-elements (like *Pekir* or *Nukem*) were used to ensure that the children exclusively applied the rules of the artificial formula language. The children were asked to answer 80 trials in two sessions during fMRI scans. A single trial consisted of the auditive and visual presentation of a chemical compound name and the visual presentation of a pair of structural formulae (Figure 1). The subjects were asked to decide which of the two structural formulae (one on the left, the other on the right matches the compound name. The total presentation of a structural formula lasted for 4.5 seconds. An additional time of 1 second for the answer has been granted, so that the maximum response time amounted to 5.5 seconds.

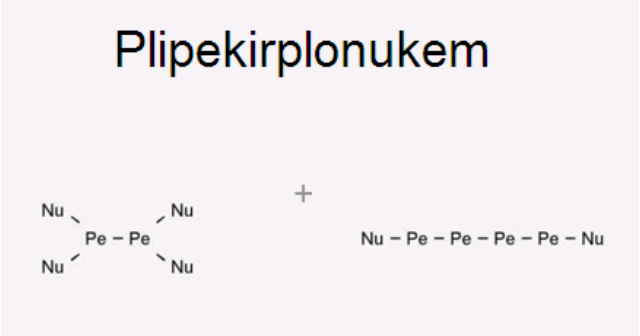


Figure 1: A typical experimental trial: The compound name is at the top, structural formulae left and right below.

If the response had occurred in time, a feedback was given after a jitter time of 2-18 seconds. The feedback consisted of two parts: one part informed about the participant's performance; a second, affective part informed about the performance of a fictional peer group. The total feedback presentation lasted for 2.5 seconds.

In order to find the correct structural formula for a compound name, six rules, which were part of the instruction given to all participants, had to be applied and checked for violations:

1. The abbreviation for an element is defined by two letters
2. The first letter of the abbreviation is the same as the first letter in the name of the element
3. Both letters appear in the element's name
4. An element may have a multiplicity from 1 to 4 in the compound. Distinct numerals are used to denote the multiplicity:
    - -/one
    - pli/two
    - pla/three
    - plo/four
5. The position of a numeral is always in front the element in the compound name
6. The central or inner element of the structural formula is always the first in the compound name

In Figure 1, the left structural formula actually matches the compound, while the right formula's cardinalities mismatch. These rules define the constraints of a strategy space from which correct *personal* strategies can be constructed by the subjects. There is no explicit order in which the rules should be applied. Either the left or the right formula violates at least one of the rules. The trials are thus classified by the position of the faulty formula (left/right) and by the number of the violating rule.

The rules were well known by the children because they went through an extensive instruction phase in multiple sessions. They familiarized themselves with the rules using age-based material and games especially designed for that purpose. They also passed 20 trials on a computer and another 40 in an fMRI simulator prior to entering the actual fMRI experiment.

Overall, 33 participants were included in our study concerning the brain-mapping hypothesis. They were distributed among five experimental groups defined by design matrices, which described the sequential order of trials and jitter times. These 33 participants scored an average 54.64 correct answers from a whole of 80 problems with a standard deviation of 11.9. On the average, they were able to signal the correct solution to the problem in a trial within 3.78 seconds with a standard deviation of 0.8s.

A SONATA MRI system (Siemens, Erlangen, Germany) operating at 1.5T was used with a standard whole-head coil to obtain T2*-weighted echoplanar (EPI) images with BOLD contrast (matrix size: 64x64, pixel size: 3x3 mm^2). Participants completed two experimental runs consisting of 40 trials each. During each functional run 408 volumes of 30 three mm-thick axial slices were acquired sequentially with a 0.6 mm gap (TR = 2 sec, TE = 50 msec). Data were preprocessed with the Statistical Parametric Mapping software SPM5[1]. Following rigid body motion correction, the time series of each voxel was realigned temporally to the middle slice to correct for differences in slice acquisition time. Structural and functional volumes were coregistered and spatially normalised to a standard T1 template based on the Montreal Neurological Institute (MNI) reference brain (resampled to 2x2x2mm^3 voxel). The data were then smoothed with a Gaussian kernel of 8 mm full-width-half-maximum to accommodate intersubject anatomical variability.

Models

Two input channels are available to the problem solver. The visual input channel is mandatory, while the auditory input channel is auxiliary. This fact adds to the complexity of the problem, especially as both channels may be perceived in parallel or consecutively. Either the left or the right formula

[1] http://www.fil.ion.ucl.ac.uk/spm/software/spm5 6/16/2010

or both have to be evaluated visually. This results in a variability of conceivable strategies, which differ in efficiency as well as module activation. A set of *basic tasks* is derived from the rules. These tasks are shared by all strategies, though not necessarily in the order presented here:

1. Visually and/or auditorially perceive and encode the different parts of the compound name (mandatory for any successful strategy)
2. Count the outer elements of a structural formula and compare them with the second numeral in the compound name
3. Count the inner elements of a structural formula and compare them with the first numeral
4. Compare the inner element with the first element of the compound name
5. Compare the outer element with the second element of the compound name
6. Indicate the correct formula

Tasks 2-5 may be applied to both formulae, or, more efficiently, to either the left or the right formula. It should be noted that some concurrency can take place if the compound name is encoded using only auditory input. Tasks 4 and 5 may be split into two different tasks as the abbreviation of an element always consists of two letters. Since the first letter is easier to compare with the name, it may be more appropriate to prioritize the first comparison and leave the second letter for later. A second open question which is not reflected within the above list of tasks is the position of the retrieval for the numerals. It can take place very early when encoding the compound name, but there is also the possibility to retrieve the numeral later on between the counting and comparison stages.

A strategy is defined by the order of task processing and the formulae Tasks 2-5 are applied to. While all the strategies share the same basic set of tasks, they all perform differently on each trial. Some trials may only be solved by counting the elements as in Figure 1, others by name-element comparisons, still others by both. A strategy shows higher performance (shorter response time) if it concentrates on a single structural formula to decide whether it matches or not. Each trial class (the violated rule and location of the violating formula) may have an impact on the performance of the strategy.

Several, though so far not all possible, strategies were modeled, at first on an abstract layer as UML activity diagrams, and subsequently within the ACT-R environment as a set of production rules. As only expert participants were modeled, all modeled strategies find the correct answer but with a large variation in performance. So far, four different strategies, S1 to S4, have been modeled (Table 2). They differ in that they either process the structural formula and the compound name simultaneously using the different input channels, or by processing the compound name first and then proceed to the structural formulae. Thus they either process the trial single- or multithreaded, or single-formula or both formulae.

Table 2: Characteristics of strategies/models

	Multi-Thread	Single-Thread
Single Formula	S1	S3
Both Formulae	S2	S4

Apart from these single- vs. multi-tasking and single vs. both formulae considerations, even more design options are available to the modeler yet. For instance, the exact time when certain tests are carried out may be varied. Thus, the model could compare the element's abbreviations with their respective names before comparing the cardinalities. Also, the costly checking of the second letter of the abbreviation may be postponed by the strategy in order to save time. A heuristic approach could leave the second letter out of consideration completely.

The models perform quite differently on the various trials, which is reflected in the ACT-R module traces. This affects the BOLD prediction. Any realization of Task 1, perceiving and encoding the compound name, would surely engage ACT-R's Visual or Aural module, if not both, and the Imaginal module. Tasks 2 and 3, which encompass encoding and counting the structural formulae, would involve the Imaginal, the Visual and the Declarative module. Tasks 4 and 5 would also require at least the Imaginal module, but it could involve the Visual module if the second letter of the symbol has to be checked for occurrence in the compound name. As Tasks 2-5 can be arranged in any arbitrary order, or even be split into subtasks which could run in parallel, quite different patterns of module activation would emerge. This implies that even models which produce similar behaviors may predict distinct BOLD signals, if the productions involved activate different modules.

Data Analysis

It is doubtful whether the participants are able to remember their problem solving strategy for each trial. It is also possible that they applied varying strategies to trials. The choice of strategy may be related to the trial class. However, we assume that the participants already settled for a *single* strategy after the extensive instruction and training phases. In order to determine which of our models is suitable to explain the performance of the actual strategy used by the participant, we devised a Bayesian Classifier with a Bayesian Belief Network (BBN) (Jensen, 2007) as diagnostic tool. The BBN (Figure 2) is trained with data from ACT-R model runs. Subsequently, behavioral data from the actual experiment is entered as evidence for identifying the personal trial-independent strategy of the subject. Strategies are thus classified by response times (RT).

The main idea is that all models produce distinct response times for each trial. We assume that response times for a strategy are dependent on the trial. This is reflected in the BBN in Figure 2. The probability tables of the BBN are

being learned by running all of the strategy-specific ACT-R models to generate cases. This results in a data matrix whose columns correspond to the nodes from the BBN and whose rows correspond to trials. During model runs, the default values of ACT-R's parameters were used.

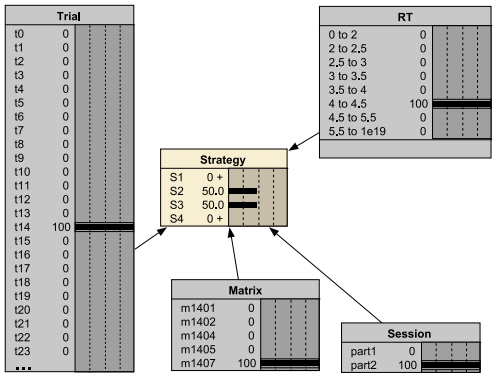


Figure 2: BBN for strategy classification

The trial is entered as evidence into the "Trial", "Matrix", and "Session" nodes. The response time of the participant is entered as evidence into the "RT" node. It is then possible to infer on the strategy most likely used by the participant in the "Strategy" node. In Figure 2, the trial in question is the 14th trial from the second session of the experimental group defined by design matrix 1407. In this particular case, for participant with a response time between 4 and 4.5 seconds, S2 and S3 are equally probable.

The collected fMRI data is analyzed by using the Regions of Interest (ROI) approach (Jäncke, 2005). The regions are specified by the module positions and dimensions given by Anderson's Brain Mapping Hypothesis in Table 1. The Talairach coordinates were transformed into MNI coordinates. The raw values of each voxel lying in the ROI are extracted from the images and averaged per region, resulting in an activation timeline for each person and region (Figure 3).

An averaged BOLD curve for each region is obtained by applying *a strategy-specific weighted means function* to and subsequent aggregation of the individual BOLD curves. For each trial t of the 80 trials, a probability $p_{s,t}$ for a particular strategy s is inferred with the BBN from Figure 2. In order to neutralize the effects of varying base levels of individual BOLD signals, we employed *ipsative measures*: the deviations from the individual's BOLD curve averages are aggregated as weighted averages using trial- and strategy-specific weights and compared with the deviations from the predictions.

For each ROI/Module pair, the averaged BOLD curve deviations are tested for correlation with the respective BOLD prediction computed from the ACT-R module activation (Anderson et al., 2008). The default parameters of the ACT-R BOLD module were used for this computation. Each time series consists of 400 data points.

As the Pearson's correlation coefficients were calculated independently for each experimental group, the resulting values were averaged among the experimental groups by using the Fisher-z transformation. Table 3 shows the final correlation results for each strategy separately for left and right brain hemispheres. If the correlation coefficient is higher than 0.098, the null hypothesis is rejected with $\alpha = 0.05$. In this case, nearly all correlations between the BOLD signal in the ROI and the ACT-R module's prediction can be considered statistically significant. This is due to the large N. The practical significance depends on the percentage of explained variance $r^2 \cdot 100$. This is the basis of our discussions.

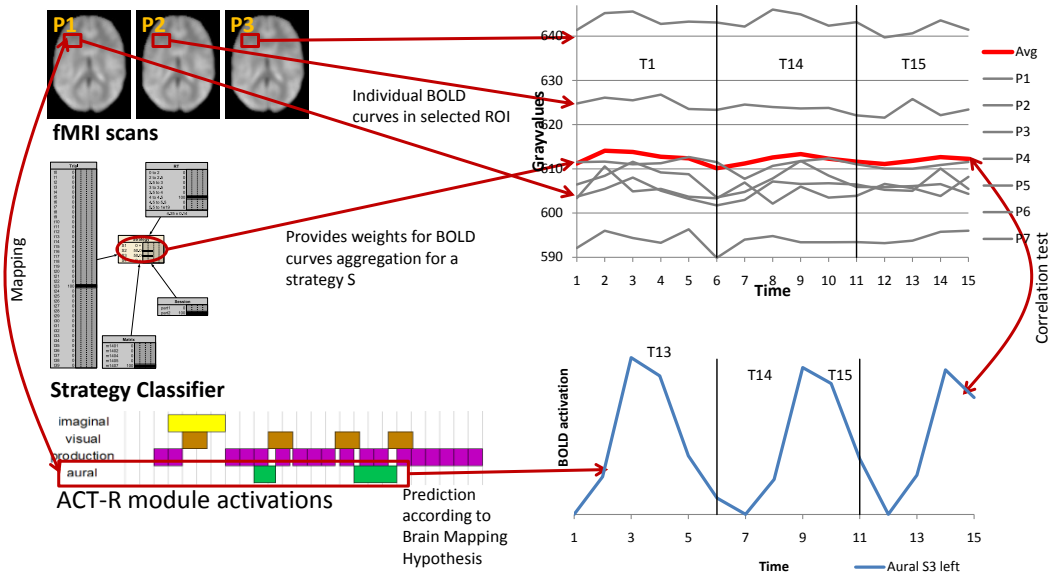


Figure 3: Aggregation of BOLD curve per ROI and correlation test with ACT-R prediction

Table 3: Correlations between ACT-R predictions and ROI activities. Each module's prediction has been tested for correlation with any of the regions from Table 1. Correlations marked with an asterisk are highest for the postulated mapping

Hemisphere	Strategy	Production	Declarative	Imaginal	Visual	Goal	Manual	Aural
Left	S1	0.458	0.365	0.258	0.525	-0.262	0.389	*0.691
	S2	0.489	0.402	0.259	*0.647	-0.267	0.403	*0.691
	S3	0.495	0.408	0.258	*0.617	-0.264	0.414	*0.692
	S4	0.489	0.414	0.246	*0.367	-0.265	0.194	*0.693
Right	S1	0.428	0.191	0.389	0.556	-0.218	-0.052	*0.659
	S2	0.438	0.220	0.397	*0.606	-0.218	-0.049	*0.660
	S3	0.450	0.216	0.389	*0.596	-0.218	-0.044	*0.659
	S4	0.432	0.231	0.397	0.295	-0.218	-0.065	*0.660

Discussion

Correlations between the Aural Module's predictions and left and right ROIs alike are high for every strategy. This might be expected, as the aural input is only available to each model for a short time, and thus the productions which perceive and encode that information fire at approximately the same time for all models.

The same applies to the Visual Module. The visual presentation lasts 4.5 seconds. During this time span, any model will perceive and encode visual information. Models S2 (multi-threaded, both formulae) and S3 (single threaded, single formula) perform with the highest correlation here. Both models show the same behavior regarding response times. However, the visual module is more engaged in the S2 model, which examines both formulae. Correlation is also the highest for this model.

The Manual Module's predictions are higher for the left than for the right hemisphere. This was expected as all subjects responded with their right hand. All strategies except S4 (single-threaded, both formulae) have a moderate correlation coefficient. The moderate correlation is surprising, as models were matched to the participants' BOLD signals according to their response time, which would suggest a higher correlation coefficient.

The Procedural Module offers fair correlations for both hemispheres and all strategies, even if the correlations for S1 are somewhat lower than those for the other strategies. The correlations of the Declarative Module's predictions are moderate for the left hemisphere and low for the right hemisphere. The higher prediction for the left Retrieval Module is in line with previous research showing a left hemispheric dominance for the retrieval of verbal information (Petrides Alivisatos, & Evans, 1995; McDermott, Buckner, Petersen, Kelley, and Sanders, 1999).

The opposite is the case for the Imaginal Module's prediction: These correlate better with the right than with the left hemisphere. The Goal Module's correlation is negative in all cases.

In general, the correlations are higher for the sensor modules, the Visual and Aural Modules. The internal modules, Procedural, Declarative, and Imaginal, show lower correlations alike. However, this cannot be ascribed to faulty assumptions in the modeling process, as they are still high when tested for significance. Rather, they suggest that participants may be occupied with other processes which the models do not account for. This could especially be the case as the experimental design provided large jitter times or delays, during which the participant remained inactive. This has also been observed by Danker and Anderson (2007).

All of our models assume a single goal which is created at the beginning of a trial. The negative correlation coefficients suggest that this assumption is wrong. Thus, the creation of sub-goals for individual tasks should be considered an alternative. A model using sub-goals would have a decreased performance and higher response times due to goal chunk creation costs. Using the Competing Strategies paradigm (Taatgen, Lebiere, & Anderson, 2006), the model would optimize performance by production rule learning.

The models' deficiencies are also evident from the scatter plots in Figure 4. These show predictions versus experimental evidence. Ideally, experimental evidence would increase with model predictions with little variance to the regression line, which would indicate similar peaks and depressions for both curves. This is clearly not the case for the Goal module on the right. Instead, both scatter plots show clustering on the prediction axis. This indicates monotonous activity patterns in the respective modules, which is due to the chunk loading and manipulation actions as implemented by the model.

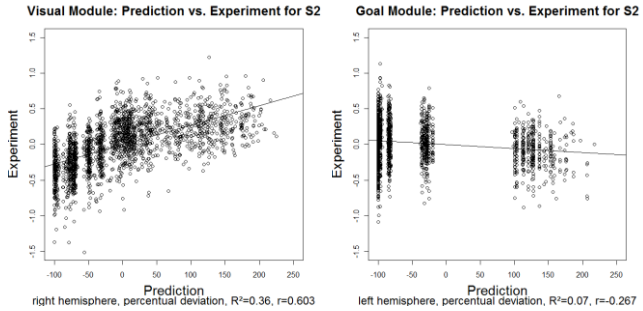


Figure 4: Scatter plots of predictions vs. evidence for S2

Conclusion

The correlations presented here are generally lower than in previous studies (Danker and Anderson, 2007). However, the experimental design, which does not account for functional separation, might contribute to that fact. For a

complex task which allows for a multitude of strategies to be pursued, many models may reproduce similar human behavior but do not predict the same BOLD curves.

The ACT-R architecture features many free parameters which may be altered in order to fit the model to experimental data, even if this may seem contrary to the intention of a cognitive architecture (Taatgen & Anderson, 2008). Also, many different modeling paradigms exist which may be more or less appropriate to the task.

Thus, three options arise for the continuation of our research. First, we could redesign our experiment in order to separate functionalities, which is the approach currently done by other research groups. Second, we could refine our models by using a modified internal representation such as sub-goal chunks. Third, we could define other ROIs and look for correlations there.

So far, the second and third choices are being pursued by us. The second choice would also include the calibration of the modified model to the individual participant's behavior by adjusting ACT-R's parameters. This should have positive effect on BOLD prediction and signal correlations.

Especially the third choice of defining alternative ROIs is of great importance. As can be seen in Table 1, Anderson's brain mapping hypothesis covers only a very small portion of the brain. However, a review of imaging research attributes the functions of ACT-R's modules to a much wider range of areas (Kaspera, 2010). Also, many of these regions seem to interact when performing a certain function, a phenomenon which the one-to-one mapping presented by Anderson cannot account for.

Acknowledgments

Cognitive Modeling and Bayesian Identification Analysis (CoMBIAn), work package within project *Impact of affective and informative feedback on learning in children before and after a reattribution training: An integrated approach using neuroimaging, educational research and modeling*, Möbus, Moschner, Parchmann & Thiel (main applicant), BMBF-Program for the Promotion of Scientific Collaboration between the Neurosciences and Research on Learning and Instruction, 03/01/2008 - 02/28/2011.

We wish to thank Mrs. P. Arndt, Mrs. B. Moschner, Mrs. I. Parchmann, Mrs. A. Anschütz, and Mr. S. Bernholt for the experimental design and support by discussing our results.

References

Anderson, J. R. (2007a). *How can the human mind occur in the physical universe?* Oxford University Press

Anderson, J. R. (2007b). Using Brain Imaging to Guide the Development of a Cognitive Architecture, in: Gray, W. D. (ed.), *Integrated Models of Cognitive Systems*, Oxford University Press

Anderson, J. R., Anderson, J. F., Ferris, J. L., Fincham, J., & Jung, K. J. (2009). The lateral inferior prefrontal cortex and anterior cingulate cortex are engaged at different stages in the solution of insight problems, *Proceedings of the National Academy of Sciences*, 106, 10799-10804

Anderson, J. R., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind, *Psychological Review*, 111, 1036-1060

Anderson, J. R., Byrne, D., Fincham, J., & Gunn, P. (2008a). Role of prefrontal cortices in associative learning, *Cerebral Cortex*, 18, 904-914

Anderson, J. R., Fincham, J., Qin, Y., & Stocco, A. (2008b). A central circuit of the mind, *Trends in Cognitive Science,* 12, 136-143

Danker, J. & Anderson, J. R. (2007). The roles of prefrontal and posterior parietal cortex in algebra problem solving, *Neuroimage*, 35, 1365-1377

Heuer, S. & Parchmann, I. (2008). Son_2e oder Fus_2bal_2 – wie Sechstklässler die chemische Formelsprache interpretieren, *Naturwissenschaften im Unterricht*, 106/107, 20-24

Jänke, L. (2005). *Methoden der Bildgebung in der Psychologie und den kognitiven Neurowissenschaften*, Kohlhammer

Jensen, F. V. & Nielsen, T. (2007). *Bayesian Networks and Decision Graphs*, Springer

Stocco, A., & Anderson, J.R. (2008), Endogenous control and task representation: An fMRI study in algebraic problem solving, *Journal of Cognitive Neuroscience*, 20, 1300-1314

Taatgen, N., & Anderson, J. R. (2008). Constraints in Cognitive Architectures, in: Sun, R. (ed.) *Handbook of Computational Psychology*, Cambridge University Press

Taatgen, N., Lebiere, C., & Anderson, J. (2006). Modeling Paradigms in ACT-R, in: Sun, R. (ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, Cambridge University Press

Kaspera, R. (2010). *ACT-R: Zur Überprüfung der Brain Mapping Hypothese (On Validation of the Brain Mapping Hypothesis)*, Technical Report, University of Oldenburg

McDermott, K. B., Buckner, R. L., Petersen, S. E., Kelley, W. M., & Sanders, A. L. (1999). Set- and code-specific activation in frontal cortex: an fMRI study of encoding and retrieval of faces and words, *Journal of Cognitive Neuroscience*, 11, 631-640.

Möbus, C. & Lenk, J. C. (2009). Bayesian Identification of Problem-Solving Strategies for Checking the ACT-R/Brain-Mapping Hypothesis, in: Schmid, U., Ragni, M., & Knauff, M. (eds.), Proceedings of the KI 2009 Workshop Complex Cognition, Paderborn, Germany, *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik*, 82, 37-47

Özyurt, J., Rietze, M. & Thiel, C. (2008). fMRI of feedback processing in children and adults, *Frontiers in Human Neuroscience*

Petrides, M.,Alivisatos, B., and Evans, A. C. (1995). Functional activation of the human ventrolateral frontal cortex during mnemonic retrieval of verbal information, *Proceedings of the National Academy of Sciences*, 92, 5803-5807.

Modeling Statistical Learning and Response Inhibition with the Change Signal Task

L. Richard Moore Jr. (larry.moore@mesa.afmc.af.mil)
Lockheed Martin
Air Force Research Laboratory
Mesa, AZ 85212 USA

Glenn Gunzelmann (glenn.gunzelmann@mesa.afmc.af.mil)
Air Force Research Laboratory
Mesa, AZ 85212 USA

Joshua W. Brown (jwmbrown@indiana.edu)
Dept. of Psychological & Brain Sciences, Indiana University
Bloomington, IN 47405 USA

Abstract

The change signal task is a two alternative forced choice task with the addition of a *change signal* presented on 1/3 of the trials at some delay relative to the initial stimulus. The change signal indicates to participants that they should inhibit their initial choice and respond with the other alternative. It provides an opportunity to examine the cognitive mechanisms involved in statistical learning and response inhibition. Within the task, change signal delays are associated with stimulus color, and are adjusted independently with a step function to produce high and low error conditions. Observed data show a significant difference in reaction times between these two conditions. In this paper we propose a model for the change signal task that leverages existing declarative memory mechanisms in ACT-R as a surrogate for the implicit contextual learning observed in human trials. We compare the mechanisms in this model briefly to an existing neural account, and use the model to predict the consequences of cue-conditional reversal.

Keywords: response inhibition; statistical learning; declarative memory; ACT-R.

Introduction

Executive control of behavior is a defining component of high-level cognition. One aspect of executive control, response inhibition, has been explored extensively using the stop signal paradigm. The classic task from Logan and Cowan (1984) visually presented subjects with one of four letters, which the subjects then classified into groups by pressing one of two buttons. On 25% of the trials an audible tone signaled that they should inhibit their response. The probability of responding was related to the timing of the *stop signal* (with a greater chance of inhibition with shorter delays) and so the authors proposed a "horse race" model for resolving executive conflict.

Brown and Braver (2005) extended the stop signal paradigm to assess error-likelihood effects. In their *change signal* task, a two alternative forced choice task is presented. On one third of the trials, however, a second stimulus is presented at some delay following the original stimulus. The second stimulus – the change signal – indicates to subjects that they should inhibit their response to the original stimulus and respond with the other alternative instead. To ensure a particular error rate in the task, the delay between the initial stimulus and the change signal is manipulated.

In Brown and Braver (2005), two colors were used for the stimuli, each of which was associated with a different target error rate. They collected fMRI data from participants across the four stimulus conditions (i.e., Change versus No Change trials crossed with High versus Low error probability) to evaluate two alternative models of anterior cingulate cortex (ACC) function. The successful model, known as the error-likelihood model, correctly predicted a learned response in the ACC that was sensitive to the stimulus color (error rate condition), for both the "go" and "change" trials.

The model presented in Brown and Braver (2005) was focused on understanding the role of the ACC in learning to recognize situations in which the risks of errors are high. Previous work suggested that the ACC detected actual errors (Dehaene et al., 1994) as well as conditions of response conflict (Botvinick et al., 2001). The error likelihood model further suggested that the ACC activity warns of an impending error as a basis for implementing proactive control.

There are other interesting aspects to the empirical data that are not addressed directly by Brown and Braver (2005). For instance, the model does not address the sequential behavior of participants in terms of their reaction times. In addition, the model does not explicitly account for differences in reaction times for the two different error conditions. These effects in the empirical data provide further evidence regarding the cognitive mechanisms involved in human performance on this task that will be explored in the current research.

To better understand the mechanisms influencing human performance on the change signal task, we have created a complementary model that focuses on the detailed behavior

of participants. For instance, the data illustrate that the conditional learning predicted by the error-likelihood model (i.e., differences in ACC activation for High versus Low error conditions) has an impact on reactions times that unfolds over the course of many trials. We used the ACT-R (Anderson, 2007) computational cognitive architecture to model these results from Brown and Braver (2005) study. After we describe the model and results in detail, we discuss the distinct and complementary insights afforded by the modeling approach used here versus Brown and Braver (2005).

The Task

We reimplemented the original Brown and Braver change signal task in Lisp to accommodate integration with ACT-R. The only known differences include color choices, symbols presented, and response keys. Although these items were altered for implementation convenience, they have no bearing on model behavior or performance. The remaining description will focus on the task as presented to human subjects. Additional details regarding the task and instructions can be found in the supplementary materials from Brown and Braver (2005).

A schematic of the change signal task is shown in Figure 1. After a .5s blank inter-trial delay, subjects were presented with a cue stimulus in one of two colors. Unbeknownst to the subjects, the cue color represented either a high error rate condition or a low error rate condition. After one second, the cue was replaced with a similarly colored *go signal*—an arrow pointing either right or left. The subjects were instructed to respond to the go signal by pressing the corresponding right or left arrow key on the keyboard.

On one third of the trials, a larger arrow pointing in the opposite direction of the go signal appeared after a change stimulus delay (CSD). (Again, the coloring was consistent with the error rate condition.) In this case, subjects were instructed to inhibit their initial response to the go signal, and instead respond to the "change signal." A response ended the trial, which progressed directly to a blank screen and the inter-trial delay. No feedback was presented. If the subject failed to respond within one second after the go signal appeared, the trial timed out.

The high and low error rate conditions were bound to unique CSDs, which were adjusted independently using a step function to maintain a consistent error rate defined for each condition. In both error rate conditions, CSDs were constrained to a range of 20 to 800ms and incorrect responses reduced the CSD by 50ms. In the low error rate condition, correct responses led to a 2ms increase in the CSD, while in the high error rate condition the CSD increased by 50ms when a correct response was made. These adjustments were made to motivate a 4% error rate, and a 50% error rate, respectively. Responses made prior to the presentation of the change stimulus were considered errors.

The original experiment used five blocks with approximately 107 trials each, although the trial count varied slightly across subjects. Our task fixed this number to 107, and the direction of the go signals and error rate conditions was randomized and counterbalanced within each block as best as possible. Stimulus colors were consistent with the error rate condition in all blocks except the last. For the final block, the relationship between stimuli colors and error rate conditions was reversed.

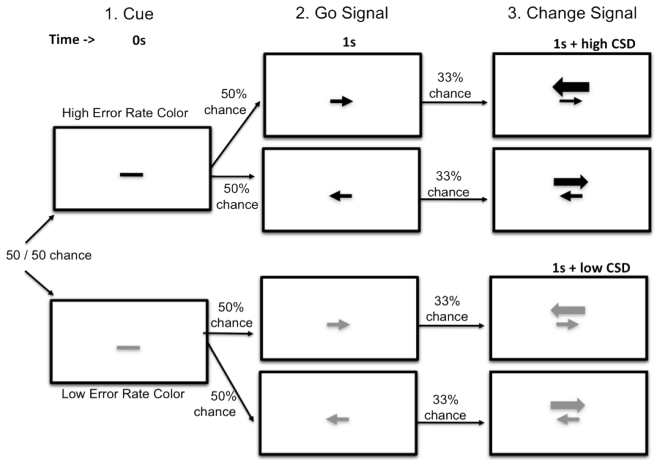


Figure 1: Task schematic. A cue signal is presented in one of two colors, followed by a go signal 1 second later. There is a 33% chance that a subsequent change signal will be presented, the timing of which is determined by a change stimulus delay bound to the signal color.

Human Performance

Figure 2 shows aggregate reaction times across trials collapsed across subjects and conditions. The solid line represents the central tendency as predicted by a simple linear regression of a logarithmic model, although the regression is intentionally discontinuous at the start of the reversal block, indicated by the grey area. The subjects performed more slowly across trials until they reach an asymptote. The regression model coefficient affecting the rise and asymptote of the curve is significantly greater than zero for the normal trials ($p < .001$), and not significant for the reversal block. This suggests that there are not enough reversal block trials to reveal an effect, if there is one.

Time on task effects may account for some of the performance decline (e.g., Gunzelmann, G., Moore, L. R., Gluck, K. A., Van Dongen, H. P. A., & Dinges, D. F., 2010), but we believe that the more influential factor is that subjects were strategically hedging their responses to improve their odds of successfully responding to change signals. (Of course, such a strategy is futile in this experiment because the CSDs were adjusted to encourage a consistent error rate.) Evidence for strategic hedging becomes apparent when we examine reaction times for each condition, also shown in Figure 2. The dashed line shows the central tendency for the high error rate condition, and the dotted line shows the central tendency for the low error

rate condition. Again, the regression is intentionally discontinuous at the start of the reversal block.

Not surprisingly, the statistics for the two error rate conditions match those of the collapsed data, with highly significant coefficients for the normal blocks ($p < .001$) and insignificant coefficients for the reversal block. The confidence intervals for the normal block coefficients, however, are more interesting because they do not overlap. ($17.8 < A_{high} < 27.7$, and $3.0 < A_{low} < 11.2$) The significant difference between these coefficients suggests a relationship between stimulus color and reaction time. In other words, over the duration of the experiment, subjects learn to delay their response more for the high error rate condition than for the low error rate condition. A simple time on task effect would not produce a disparate hedge times across error rate conditions.

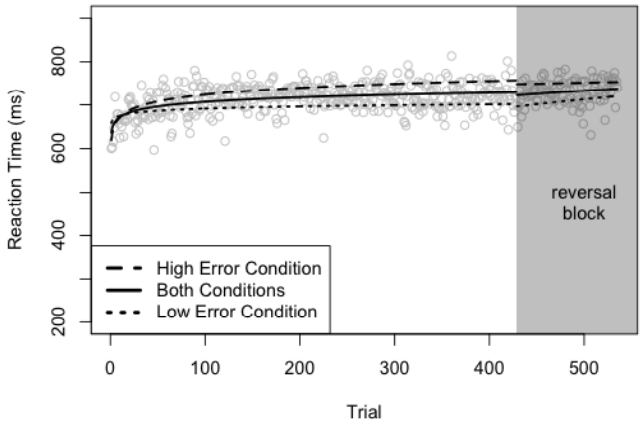


Figure 2: Reaction times collapsed across conditions are shown in the grey scatterplot, with the central tendency shown as a solid black line. Central tendencies for the high and low error conditions are shown as dashed and dotted lines, respectively. The central tendencies, generated through regressions, are discontinuous at the start of the reversal block, shown in grey.

The Model

The ACT-R 6 (Anderson, 2007) cognitive architecture provides the computational framework for our model. It integrates perceptual, cognitive, and motor processing mechanisms from the psychological literature. At its core, it is a symbolic production system with a semantic network memory and simulated subsymbolic effects. Specifically, our model leverages the procedural and declarative capabilities, the intentional module, and a timing capability derived from a temporal module (Taatgen, Van Rijn, & Anderson, 2007).

The empirical data from Brown and Braver (2005) demonstrate that subjects implicitly learned the association of stimulus color to error rate condition. In this paper, we show that this learning measurably influenced subject performance—their response times were strategically mediated by stimulus color. Out of several possible approaches to model this in ACT-R, we chose to use the declarative module to emulate the statistical learning attributed to the ACC.

From the perspective of the ACT-R theory, the declarative module is not intended to represent the functional properties of the ACC (see Anderson, 2007), but it does provide the appropriate Bayesian dynamics to represent the learning we hypothesize may be involved. Thus, we treat the declarative module as a surrogate for the ACC functionality that is not represented by existing mechanisms in ACT-R. This absent functionality would appear to appropriately reside within ACT-R's intentional module, which is associated, in part, with ACC function (Anderson, 2007).

The model employs a simple hedging strategy to accomplish the task. Upon attending to a cue, it attempts to retrieve a similar trial from declarative memory based on the cue color. When the subsequent go signal is attended, the model does not respond immediately. Instead, it waits according to a remembered "hedge time" from the trial that was retrieved from declarative memory. If no similar trial exists (i.e., nothing was retrieved), a default initial hedge time is used, which is a free parameter discussed below. If a change signal is seen prior to the expiration of the hedge time, a response is made accordingly. If no change signal is seen and the hedge time expires, the model responds to the go signal.

Even when the model responds to the go signal, the key press does not occur immediately. Instead, the ACT-R motor module initiates a 3-phase motor movement that can take well over 100 milliseconds before the actual key press is registered by the task (Byrne & Anderson, 2001). During this time, the model can detect a change signal, although it is too late to cancel the requested motor action. The model learns from its failure by associating the CSD with the color for that trial in its goal buffer of the intentional module. This timing information is based upon estimates from the temporal module (Taatgen et al., 2007).

At the start of the next trial, the contents of the goal buffer, which includes the association between the stimulus color and hedge time, is stored in declarative memory to serve as an exemplar for future trials. Because detected errors typically associated a longer hedge time than what was originally retrieved, they have the effect of increasing future hedge times (Rabbitt, 1966). As currently written, the model has no specific mechanism to reduce hedge times.

Without a mechanism to reduce hedge times, it might seem that model response times would always increase and never asymptote. Indeed, sharp increases in hedge times do occur in early trials. However, because each stimulus color / hedge time pairing is stored as an independent chunk (i.e. there is no merging) the likelihood of retrieving a particular hedge time increases the more often it is used, in part due to the influence of stochasticity in declarative memory. After a large number of trials, the declarative memory becomes so saturated with hedge times associated with each stimulus

color, that the model's hedging essentially reaches a steady state.

Three parameters were involved with fitting the model to observed data. The first is the initial hedge time, which we believe was established either through practice trials or as a side effect of instructions that informed subjects of delayed change signals. This has the simple effect of moving the y-intercept in Figure 3.

The second free parameter was activation noise, which reflects the effect of subsymbolic processes in the declarative memory system. Noise influences the likelihood that recent and correct declarative information will be retrieved. In terms of the curve in Figure 2, noise affects the overall shape—higher noise flattens it out. In ACT-R, activation noise is set with the ans parameter, for which we settled on a value of .53. This produces a standard deviation of .96 in the distribution of noise that is sampled to add stochasticity to the activation of declarative memories.

Lastly, the ACT-R declarative memory system allows for errors of commition through a mechanism called partial matching. We used this mechanism so that the model would be indifferent to stimulus colors in early trials and develop a differentiation in later trials. The mechanism requires us to specify a degree of similarity between stimulus colors, which we set to 50%. We did not use this as a free parameter in the fitting processes because the other parameters provided the necessary degrees of freedom.

Results

Using the parameter values described above, we aggregated the results from 100 model runs to obtain reliable measures of central tendency. A comparison of reaction times between model and human data are shown in Figure 3. Because a large amount of stochasticity still remains even after aggregation, the model results are represented using linear regressions of a logarithmic model in the same way the human data is shown. (The standard deviation is considered as a separate measure of fitness below.)

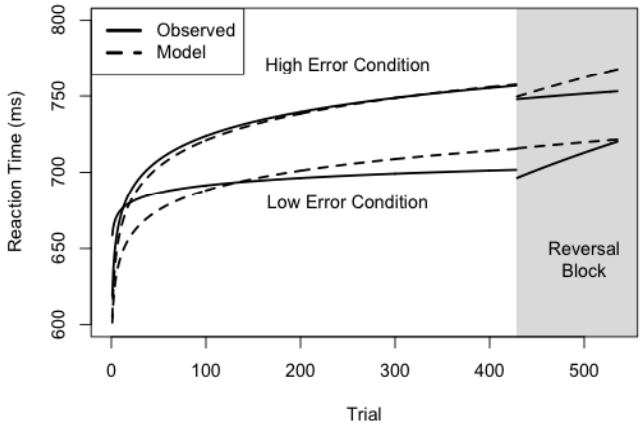


Figure 3: ACT-R model results are shown as dashed lines on top of the human data shown as black lines.

The RMSD values calculated from the model and human reaction time data are shown in Table 1. The overall mean RMSD was 58.5ms, which seems reasonable given that some of the deviation is a result of remaining stochasticity in the model and human data.

Table 1: RMSD values between model and human data.

Condition / Block	RMSD (ms)
High Error / Normal	51.6
High Error / Reverse	48.8
Low Error / Normal	74.3
Low Error /Reverse	59.1

The high stochasticity suggests that the standard deviation of the reaction time is another important measure of fitness (non-responses were removed for this analysis). Figure 4 overlays model performance on top of a box plot of the subject data. The model's standard deviation was in the middle of the 1st quartile for the subject data. This could be improved by increasing noise in other areas of ACT-R, but we opted against doing so in the interest of parsimony.

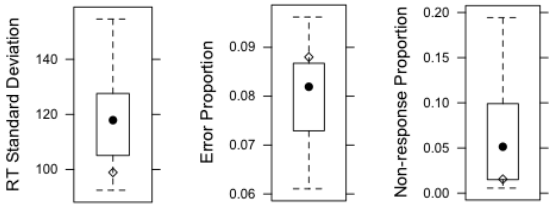


Figure 4: ACT-R model standard deviation, error proportion, and non-responses overlaid on subject data. The hollow diamonds indicate ACT-R values.

The proportion of incorrect responses made was also a consideration. For purposes of this analysis, an incorrect response occurs when the subject presses the wrong arrow key, regardless of condition. Since a response is actually made, this does not included non-responses, which are analyzed separately below. Also shown in Figure 4, the results were within the range of humans, although on the high side.

The remaining measure of fitness is the proportion of non-responses. A non-response occurs when the model fails to respond to a go signal within 1 second. The temporal module in ACT-R adds some stochasticity to the timing so this can occur even if the intended hedge time is within the trial period. Again, the non-responses were well within the human range (see Figure 4), but on the low side of the second quartile. As was the case with standard deviation, this could be improved if we allowed the model another degree of freedom.

Finally, fMRI studies, including the Brown and Braver (2005) work, often use a blood oxygenation level-dependent (BOLD) contrast mechanism. With this technique, regions

of the brain with higher blood oxygenation appear more intensely on images, which indicates greater neural activity. ACT-R uses buffer activity to make BOLD predictions (Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004), as shown in Figure 5. In this figure, ACT-R makes BOLD predictions for the ACC region based on activity in the goal buffer of the intentional module. To produce this graph, the inter-trial delay was extended to 10 seconds to isolate responses. Data was aggregated from 12 normal blocks of 107 trials.

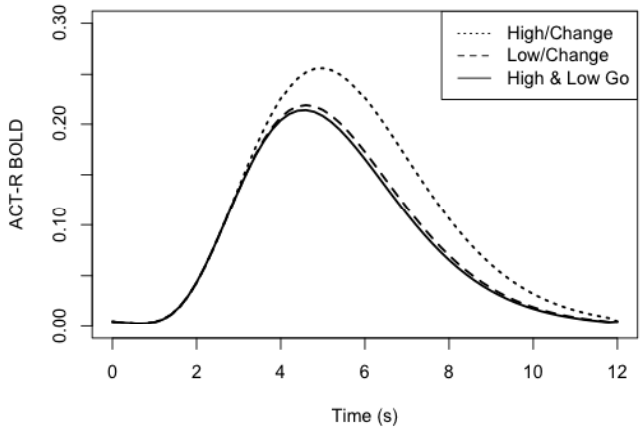


Figure 5: ACT-R BOLD predictions for the ACC region in each of the four conditions.

Discussion

As modelers, we often confront (and perhaps carry our own) biases related to specific modeling approaches, whether it be production level architectures like ACT-R, connectionist approaches like the error likelihood model, diffusion models, dynamic systems, or others (Anderson & Lebiere, 2003). This is unfortunate, because as this research demonstrates, each methodology maintains distinct advantages as well as disadvantages that may be overcome using a variety of techniques. Specifically, the error likelihood model makes detailed predictions about neurological processes in the ACC beyond the current scope of ACT-R. However, ACT-R brings to the table a generalized account of end-to-end perceptual-cognitive activity, which can reproduce observed behavior.

If we accept that both models contain elements of truth, there must be some functional overlap despite the differing levels of abstraction. Recent work on the theory of ACT-R has focused on mapping functionality to specific brain regions (e.g., Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004). Specifically, the ACC is attributed to the ACT-R intentional module, which includes the goal buffer (Anderson, 2007). The goal buffer typically maintains the internal and relevant external information required to make decisions. This is intended to include the conflict resolution typically attributed to the ACC, but it is a functionally broader interpretation.

In our change signal model, the goal buffer contains the stimulus color and hedge time, among other state information. The current implementation of ACT-R provides no functional computation in the intentional module, so the statistical learning demonstrated by the error likelihood model involves knowledge maintained in the declarative module, which acts as a surrogate. Our position that the declarative memory acts as a surrogate is largely based on that fact that many subjects were unable to explicitly distinguish the difference between stimulus colors in terms of their pairing with error likelihood even after the experiment.

This is not a firm position, and we are planning a follow-up study to guide our modeling direction. A more detailed participant debriefing will help determine the degree of declarative learning and influence on behavior. The results may suggest that the declarative component is more than just a surrogate—perhaps the ACC activity is epiphenomenal to declarative function. On the other hand, it may be confirmed that there is little relation between declarative knowledge and subject behavior with respect to high and low error conditions. In this case, the model may evolve towards a bottom-up learning approach, perhaps though augmenting the intentional module in ACT-R or focusing on a procedural learning approach.

In the mean time, the declarative module provides a reasonable proxy for ACC function because it employs a similar statistical learning process. Because the information managed in declarative memory relates stimulus color and hedge times, greater activity occurs when change signal errors are detected. This is reflected in the goal buffer, which results in higher predicted BOLD responses in ACT-R. Furthermore, because errors are detected 3x more often in the high error rate change condition, its mean BOLD response will rise above all other conditions. This is supported in Figure 5.

The ACC BOLD responses recorded in the Brown and Braver (2005) study aligns with some, but not all, of the ACT-R predictions. Specifically, the high error change condition shows the highest activation, followed by low error change and high error go conditions which are essentially tied.

The low error rate go condition is a significant divergence, as the BOLD response show that the activation is clearly lower than the other conditions in that region. Unfortunately this was one of the key findings that distinguished the error likelihood model from the alternative "conflict" model. The current ACT-R model does not produce a similar prediction because extra goal manipulation only occurs when errors are detected in change conditions. One could argue that this is a response to the statistical learning that was delegated to the declarative memory system in our model. In this regard, the ACT-R model stands in contrast with the Brown and Braver (2005) model, which predicted greater fMRI activity in ACC for

high vs. low error likelihood trials, even when restricted to correct trials with no change signal. Nevertheless, if the hedge time in the declarative memory were to increase the simulated fMRI activity, then our model might be able to simulate an error likelihood effect in ACC activity.

Finally, with an ACT-R model of the change signal task performing reasonably well, we have an opportunity to make a prediction. The reversal block in the observed human data had surprisingly little effect, and the ACT-R model produced similar results. By extending the number of reversal blocks, we can predict how many trials will be required to see an effect, and what that effect might be.

The predicted results of 24 reversal blocks are shown in Figure 6. As mentioned previously, the model does not currently have a mechanism to reduce hedge times. Both conditions achieve a steady state at their asymptotes through a combination of accumulated statistical evidence and retrieval noise. Even when failures to respond to change signals are detected and increased hedge times remembered, noise in the declarative retrieval process makes it unlikely that the latest trial information will be retrieved over the large number of older, lower trial hedge times available.

Without this statistical influence, the low error rate condition would never achieve an asymptote below the high error condition without a mechanism to hedge downward. This also provides an explanation for the predictions in Figure 6, which continue on the same trajectory as the normal block. In contrast, the error likelihood model of Brown and Braver (2005) would predict that over time, the ACC will learn the reversed error likelihood pairings, leading to a reversal of error likelihood effects on reaction time. Although our current data is insufficient to make concrete statements about which prediction is correct, our follow-up study will extend the number of reversal blocks with hopes to allow such a test. Once again, this will help inform future model development.

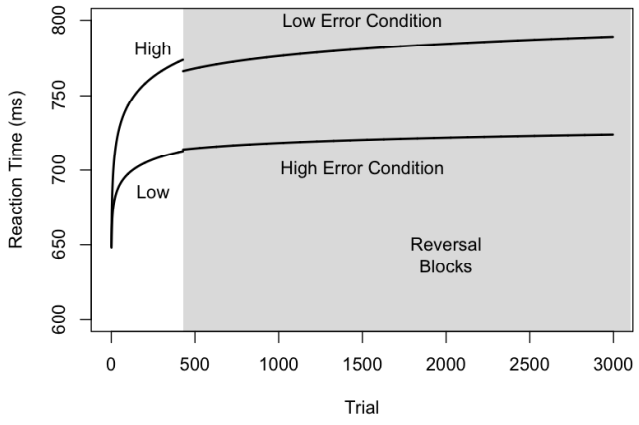


Figure 6: ACT-R model prediction of color reversal over 24 blocks, shown in the grey region.

Acknowledgments

The views expressed in this paper are those of the author and do not reflect the official policy or position of the Department of Defense, the U.S. Government, or Lockheed Martin Corporation. This research was sponsored by grants 07HE01COR and 10RH04COR from the Air Force Office of Scientific Research. We would like to thank Tim Halverson for his contributions to this research.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford, UK: Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y . (2004). An integrated theory of the mind. *Psychological Review* 111, (4). 1036-1060.

Anderson, J. R. & Lebiere, C. L. (2003). The Newell test for a theory of cognition. *Behavioral & Brain Science* 26, 587-637.

Botvinick, M. M., Braver, T. S. , Barch, D. M., Carter, C. S., Cohen, J. C. (2001). Conflict monitoring and cognitive control. *Psychological Review,* 108, 624-652.

Brown J. W, & Braver T. S. (2005). Learned predictions of error likelihood in the anterior cingulate cortex. *Science* 307, 1118–1121 doi: 10.1126/science.1105783.

Byrne, M. D., & Anderson, J. R. (2001). Serial modules in parallel: The psychological refractory period and perfect time-sharing. *Psychological Review*, 108, 847-869.

Dehaene, S., Posner, M.I., & Tucker, D. M. (1994). Localization of a neural system for error detection and compensation. *Psychological Science,* 5, 303-306.

Gunzelmann, G., Moore, L. R., Gluck, K. A., Van Dongen, H. P. A., & Dinges, D. F. (in press - 2010). Fatigue in sustained attention: Generalizing mechanisms for time awake to time on task. In P. L. Ackerman (Ed.), *Cognitive Fatigue: Multidisciplinary Perspectives on Current Research and Future Application.* Washington, DC: American Psychological Association.

Logan, G. D. & Cowan, W. B. (1984). On the ability to inhibit thought and action: A theory of an act of control. *Psychological Review*, 91, 295-327.

Rabbitt, P. M. A. (1966). Errors and Error Correction in Choice-Response Tasks. *Journal of Experimental Psychology*, 71, 2, 264-272.

Taatgen, N., Van Rijn, H., Anderson, J. R. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention, and learning. *Psychological Review*, 114(3), 577-598.

Rewards and Punishments in Iterated Decision Making: An Explanation for the Frequency of the Contingent Event Effect

Antonio Napoli (antonio.napoli@phd.units.it)
Danilo Fum (fum@units.it)
Dipartimento di Psicologia, Università degli Studi di Trieste
via S. Anastasio, 12, I-34134 Trieste, Italy

Abstract

Iterated decision making can be studied in laboratory using situations, like the Iowa Gambling Task (IGT), in which participants face repeatedly the same decision problem getting feedback after each choice. In the paper we focus on a recurring finding in experiments carried out with the IGT, the frequency of the contingent event effect—i.e., the fact that people consistently prefer options associated with rare losses, independently of their attractiveness, expected value and loss magnitude—that has not yet received a satisfactory explanation. An experiment reveals that the effect relies on simply experiencing rewards and punishments, not being influenced by the net outcome (loss or win) to which they are associated, and a computational model, implemented in the ACT-R cognitive architecture, corroborates the idea that punishments and losses on one hand, and rewards and wins on the other, play the same functional role in determining the participants' behavior in IGT.

Keywords: Iterated decision making; Reinforcement learning; Iowa Gambling Task; ACT-R; Feedback.

Introduction

Iterated decision making relies on the regulation of behavior according to its consequences. This process is characterized by three steps (Ahn, Busemeyer, Wagenmakers, & Stout, 2008): (1) the choice of a possible option and the execution of the associated action, (2) the encoding of the action consequences, (3) the integration of the consequences in a format that allows options comparison. Iterated decision making can be simulated in laboratory using the so-called *multi-armed bandit* tasks (Sutton & Barto, 1998) in which participants face repeatedly the same decision problem and get a numerical reward after each choice. Behavior in multi-armed bandit tasks is usually modeled by Reinforcement Learning models in which agents, requested to maximize their expected total reward over a given number of trials, learn about the structure of the environment by taking into account the reward associated with each choice. In the paper we will adopt Reinforcement Learning to explain the results obtained in a particular multi-armed bandit task, the Iowa Gambling Task—henceforth, IGT (Bechara, Damasio, Damasio, & Anderson, 1994). Our models will be based on the ACT-R cognitive architecture (Anderson, 2007) which provides the resources for the steps (1) and (3) of the decision making process described above, and we try to figure out how step (2) is carried out.

The IGT has been proposed as a simulation of real life decision making in the way it factors reward, punishment and outcome uncertainty (Bechara et al., 1994). The IGT involves four decks of cards. Participants repeatedly choose a card at a time from one of the decks. Each time a card is turned, it allows participants to gain a given amount of money, but sometimes the card forces them to give up some money, too; therefore, while all cards contain a reward, only some cards contain a punishment. Two card decks (let's call them A and B) feature high wins per card ($100) but they yield also higher losses so that, by choosing them, participants lose more money than they win. These decks are referred to as "bad decks". The remaining decks (C and D) give rise to small gains ($50) but even smaller losses, so that it is profitable to choose cards from them. These decks are referred to as "good decks". Generally participants, after being initially attracted by the dangerous bad decks featuring high wins and higher losses, gradually shift their preferences toward the good ones, a result which has been replicated by most IGT studies (Dunn, Dalgleish, & Lawrence, 2006). So, according to the standard interpretation, participants' behavior can be explained by a conflict between two deck features: their *attractiveness*, i. e., the amount of money each cards allows immediately to win—which drives the participants choices in the first trials—and the long term *expected value*, i. e., the net amount of money gained or lost— which drives them in the subsequent trials.

In recent years a growing number of researchers have been suggesting that this interpretation of the IGT is unsatisfactory (see Dunn et al. (2006) for a critical review of the literature). In the present paper we will focus on a recurring finding in the experiments carried out with the IGT which has not yet received a satisfactory explanation. This finding has been termed the "frequency of the contingent event effect" by Fum, Napoli, and Stocco (2008) and the "prominent deck B phenomenon" by Chiu et al. (2008) and refers to the fact that people consistently prefer the decks associated with rare losses—to the point that the bad-but-rare-loss deck B which gives raise to a small number of losses is consistently preferred to the good-but-frequent-loss deck C—independently of their attractiveness, expected value and loss magnitude. Even if the theoretical interpretations of the phenomenon put forward by the two research groups are similar, they differ in some important details.

Frequency of the contingent event

Traditionally, the performance in the IGT has been recorded by subtracting the number of bad deck selections from the good ones (the so-called Good−Bad index). In the original version of the IGT (see Table 1), for every block of ten cards, decks A and C originate five money losses while decks B and D give rise to only one.

Table 1: Deck matrices of the original Iowa Gambling Task

Card	A		B		C		D	
#	**Rew**	**Pun**	**Rew**	**Pun**	**Rew**	**Pun**	**Rew**	**Pun**
1	+100	0	+100	0	+50	0	+50	0
2	+100	0	+100	0	+50	0	+50	0
3	+100	-150	+100	0	+50	-25	+50	0
4	+100	0	+100	0	+50	0	+50	0
5	+100	-300	+100	0	+50	-75	+50	0
6	+100	0	+100	0	+50	0	+50	0
7	+100	-200	+100	0	+50	-25	+50	0
8	+100	0	+100	0	+50	0	+50	0
9	+100	-250	+100	-1250	+50	-75	+50	-250
10	+100	-350	+100	0	+50	-50	+50	0
EV	Bad		Bad		Good		Good	

Rew: Reward. Pun: Punishment. EV: Expected Value. Punishments which do not result in a net loss are evidenced in gray.

Because A and B are the bad decks and C and D are the good ones, any possible effect of the number of losses is confounded with that of the deck quality, as expressed by their expected value. In recent years researchers have started to present the analytical data for each deck and evidence has been growing about the "frequency effect", i.e. the functional role that the frequency of money losses could play in addiction (or in opposition) to the effects of decks' attractiveness and expected value.

To understand which deck features exert the most important effect on IGT, Fum et al. (2008) manipulated the decks pay-off matrices in three different experimental conditions. In all the conditions the decks attractiveness and the loss frequency were kept the same as in the original IGT, while their expected values were manipulated. The first condition replicated the setting of the original IGT. In the second condition the expected value of the decks was zeroed, so that the amount of money participants were expected to win in the long run for each deck was identical to that they were expected to lose. In the third condition the two decks with frequent punishments (A and C) were good while the decks with less frequent punishments (B and D) were the bad ones; in this case loss frequency and expected value were put in opposition for each deck.

Two findings were particularly significant: (1) the number of selections from each deck was almost the same in all the conditions, and (2) participants showed a strong preference for the low frequency loss decks, even in the condition in which these decks were bad. In the same study, the IGT task was carried out in a scenario in which participants always lost money when they turned a card while the contingent event was represented by a win, a variant originally developed by Bechara, Tranel, and Damasio (2000). Similar results were obtained with the same pattern of choices in all the conditions and a strong preference for the decks originating a higher number of wins. The fact that participants chose the same number of cards from all decks despite the change in their expected value means that this feature plays a small or no functional role in determining their choices. The fact that participants preferred the decks with a small number of losses (or those with a high number of wins) means that the frequency effect is both independent from and much stronger than the effect of the other two features. This effect was termed "the contingent event effect".

An important empirical finding remains, however, unexplained by the contingent event effect and it is constituted by the fact that, when this effect is confounded with that of the expected value, a preference for the economically advantageous decks (a "goodness" effect) is normally found which indicates that the frequency of the contingent event cannot cover the whole story in the IGT. Stocco, Fum, and Napoli (2009) hold the idea that participants' behavior in this task is guided by a dual process. The first one is a low-level emotion-based mechanism which is sensitive to punishment (or reward) frequency, while the second one, high-level and based on the analysis of the monetary outcomes, is sensitive to the decks' expected value. Even if the former is normally the most important factor in guiding participants' choices, the latter may sometimes enter into play being responsible for the goodness effect.

A different explanation for the goodness effect which devaluates the deck's expected value has been put forward by Chiu et al. (2008). In order to understand their proposal it is necessary, however, to introduce some terminological distinctions.

From now on, we will discriminate between a punishment and a loss, on one hand, and between a reward and a gain, on the other. A *punishment* is an event that happens every time participants turn a card that makes them give away money. So, for example (see Table 1), in card #3 of deck C, after having earned \$50 you are forced to give \$25 back, and this is a punishment. A *loss* is a particular kind of punishment in which the amount of money lost is higher than that won; so, in card #3 of deck A, you win \$100 but you are forced to refund \$150, and this constitues a loss. All losses are therefore punishments, but not vice versa. In the same vein, in the variant IGT in which every card turn makes you lose some money, a *reward* is a contingent event in which you earn some money while a *gain* is a reward in which the amount of money gained is higher than that lost.

Chiu et al. (2008) argue that the process driving participants' behavior in the IGT is sensitive to loss (in the sense we have just defined) frequency. Some cards in deck C (evidenced in gray in Table 1) present a punishment which is not a loss, as for example the card (+\$50, -\$25), whose outcome is a net gain of \$25. Every block of 10 cards, deck C contains on average 6.25 gains, 2.5 standoffs and 1.25 losses, deck D contains 9 gains and 1 loss, deck A contains 5 gains and 5 losses, and deck B contains 9 gains and 1 loss. Therefore, taken together, the good decks (C and D) present a total of 15.25 gains, 2.5 standoffs and 2.25 losses, whereas the bad decks

(A and B) present 14 gains and 6 losses. According to Chiu et al. (2008), the lower number of losses in the good decks explains the participants' preference for them. These authors also propose their own version of the IGT, the Soochow Gambling Task (henceforth, SGT), in which every punishment is always a loss, thus eliminating the "ambiguous" Deck C. In SGT the bad decks have a high number of wins, while the good decks have a high number of losses. Results show that participants choose more cards from the former than from the latter type of decks, and this corroborates the idea that their behavior is more sensitive to losses than to expected value.

The proposals of the two research groups differ in two respects: the first one is that Fum et al. (2008) assume that participants avoid all kind of punishments, while according to Chiu et al. (2008) they avoid only punishments which result in a net loss. The second, which is strictly tied to the first, is that according to Stocco et al. (2009), the goodness effect is due to an understanding of the decks' expected value, while according to Chiu et al. (2008) the goodness effect is due to the lower number of losses in the deck C. In this paper we present an experiment which tries to distinguish between the two proposals by addressing the (possible) different effects of punishments and losses.

The Experiment

A first idea for discriminating between the above mentioned positions is to compare the choices made from two different kind of decks that, while sharing the same expected value, provide the same number of punishments but a different number of losses. So, the first deck should give rise to a given number of losses (which are all punishments) while the second should originate the same number of punishments of which, however, only some are losses. According to Chiu et al. (2008), participants should prefer the latter kind of deck while, according to Fum et al. (2008), participants should choose the same number of cards from the two decks.

A second way of discriminating between the hypotheses would take into account the specific format of the information provided during the experiment, i.e., the feedback received after each choice. In the original IGT, participants received a "double feedback" stating separately the amount of money provided by the default and the contingent event (which could be possibly null). In a "single feedback" task (such as the SGT) each card turn informs only about the net amount of money lost or gained. According to Chiu et al. (2008), participants should exhibit the same pattern of choices both in a Single and in a Double feedback task, while, according to Fum et al. (2008), participants should modify their behavior whenever the manipulation changes the number of punishments in one or more decks.

In the experiment we contrasted the participants' behavior in a variant of the IGT featuring both a Double feedback and a Single feedback condition. In the Double condition all the decks (A, B, C and D) provided the same punishment frequency (5 every 10 cards), but for two of the decks (A and C) all the punishments were losses (giving thus 5 losses every 5 punishments) while the remaining decks (B and D) provided only 1 loss every 5 punishments (see Table 2).

Table 2: Deck matrices of the Double Feedback - Standard Frame condition.

Card	A		B		C		D	
#	**Rew**	**Pun**	**Rew**	**Pun**	**Rew**	**Pun**	**Rew**	**Pun**
1	+90	0	+90	0	+90	0	+90	0
2	+110	-300	+110	-25	+110	-125	+110	-25
3	+120	-250	+120	-1050	+120	-175	+120	-550
4	+90	0	+90	0	+90	0	+90	0
5	+100	-250	+100	-50	+100	-150	+100	-50
6	+110	0	+110	0	+110	0	+110	0
7	+120	-150	+120	-50	+120	-150	+120	-50
8	+100	0	+100	0	+100	0	+100	0
9	+80	0	+80	0	+80	0	+80	0
10	+80	-300	+80	-75	+80	-150	+80	-75
EV	Bad		Bad		Good		Good	

Rew: Reward. Pun: Punishment. EV: Expected Value. Punishments which do not result in a loss are evidenced in gray.

In the Single condition we used the same pay-off matrices of the Double condition but we presented participants only the net amount of money won or lost. This resulted in a different effect for the punishment cards which were losses and those which were not. In fact, a card such as (+$100, -$75) in the Double condition would become a (+$25) card in the Single one, thus resulting in a non-loss card. On the other hand, a card such as (+$100, -$300) would become a (-$200) card in the Single condition, giving thus rise to a net loss. As a result, decks B and D, which presented 1 loss every 5 punishments in the Double condition, had 1 loss every 10 cards in the Single condition, while the decks C and D, which had 5 losses every 5 punishments in the Double condition, presented 5 losses every 10 cards in the Single condition (see Table 3).

To control for the other features, all decks had the same attractiveness, so participants gained on average $100 every time they turned a card. The expected value was balanced instead: there was one good deck and one bad deck among the ones with high loss frequency, and one good deck and one bad deck among the ones with low loss frequency.

We ran both feedback conditions in two different frames: a Standard condition, which we just described and in which each card turn originated as default event a win and the contingent event was represented by punishments as in the original IGT scenario presented in Bechara et al. (1994), and a Reversed condition, in which participants always got a punishment when they turned a card and the contingent event was represented by rewards, as in Bechara et al. (2000). In the Reversed condition all the decks had the same reward frequency but differed in the number of gains; the effect of attractiveness and expected value was controlled in the same way as in the Standard condition.

Table 3: Deck matrices of the Single Feedback - Standard Frame condition.

Card #	A Payoff	B Payoff	C Payoff	D Payoff
1	+90	+90	+90	+90
2	-190	+85	-15	+85
3	-130	-930	-55	-430
4	+90	+90	+90	+90
5	-150	+50	-50	+50
6	+110	+110	+110	+110
7	-30	+70	-30	+70
8	+100	+100	+100	+100
9	+80	+80	+80	+80
10	-220	+5	-70	+5
EV	Bad	Bad	Good	Good

Please note that the "Payoff" column results from the sum of "Reward" and "Punishment" columns of Table 2.

Method

Participants. Eighty-eight participants (40 males) were recruited from students enrolled at the University of Trieste, in Italy. They were aged between 19 and 28 years (M= 19.9, SD= 3.7). The participants were randomly assigned to the experimental conditions. We excluded from the analyses those participants who, in some condition, turned a number of cards from a deck that differed by 3 SDs, or more, from the average number of choices made for that deck. Eight participants satisfied this criterion and were discarded.

Experimental Design. The experiment followed a 2x2 between subjects design with Feedback (Single vs. Double) and Frame (Standard vs. Reversed) as main factors.

Materials. Deck features are summarized in Table 2 and Table 3. Note that in all the conditions A and B were the bad decks while C and D were the good ones, and that B and D were those decks in which a possible frequency effect should show up since they provided low-frequency losses in the Standard condition and high-frequency gains in the Reversed condition.

Procedure. Experimental sessions were held individually. Participants played a computer-based implementation of the IGT. Decks were visually presented in the lower part of a 15 in LCD screen, and participants used a mouse to point and select the deck they had chosen. Immediately after each card selection, the amount of money obtained through the default event (and possibly through the contingent one) was displayed in the upper half of the screen. The running total of money was coarsely indicated by a colored bar in the uppermost part of the screen that was updated after each selection. In each experimental condition participants had to perform 100 card selections.

Results and Discussion

Table 4 reports the average number of choices made from each deck in the different experimental conditions.

Table 4: Means (and Standard Deviations) of deck choices in the four experimental conditions.

	Deck			
Condition	**A**	**B**	**C**	**D**
Double-Reversed	**21.06** (7.99)	**22.94** (5.03)	**26.71** (9.3)	**29.29** (9.48)
Double-Standard	**22.45** (8.18)	**23.65** (9.33)	**23.35** (9.68)	**28.55** (12.56)
Single-Reversed	**19.59** (5.82)	**25.86** (8.08)	**24.18** (10.03)	**30.36** 10.57)
Single-Standard	**17.62** (6.4)	**28.95** (12.24)	**19.57** (6.87)	**33.86** (13.24)

We analyzed the participant's performance on two synthetic indices: P, which measures the tendency to choose according to the expected value and is calculated by (C+D)−(A+B), and Q, which measures the tendency to choose according to the frequency of the contingent event. Q is calculated by (B+D)−(A+C) and it measures the preference for decks with low loss frequency in the Standard condition and decks with high gain frequency in the Reversed condition (see: Stocco et al. (2009)). We monitored the participants' behavior throughout the experiment by analyzing the two indices in successive blocks of 20 choices each. We ran a mixed design ANOVA both on P and Q, using Feedback (Single vs. Double) and Frame (Standard vs. Reversed) as between factors, and Blocks (from 1-20 to 81-100) as within factors.

As for P, the ANOVA didn't reveal any significant difference for the two factors nor for the blocks. The interaction between Blocks and Feedback resulted marginally significant ($F(4,304)=2.39$, $p=0.51$) and was caused by the low number of selections from good decks in the first block made by participants in the Single condition in comparison to those in the Double one. Since there was no main effect of any factor, we collapsed the value of P at the end of the experiment across all conditions. A t-test on this value revealed that participants chose more cards from the good decks than from the bad ones ($M=8.8$, $t(79)=3.44$, $p<.001$).

As for Q, the effect of Feedback ($F(1,76)=8.15$, $p<.01$), of Blocks ($F(4,304)=4.72$, $p<.01$) and the Blocks x Frame interaction ($F(4,304)=3.6$, $p<.01$) resulted statistically significant, while the Blocks x Feedback interaction was only marginally significant ($F(4,304)=2.1$, $p=.081$). We also performed two t-tests on the value of Q at the end of the experiment separately for the Single and Double Feedback conditions collapsing the Standard and Reversed Frame. The results were significant for the Single condition ($M=18.89$, $t(42)=5.32$, $p<.0001$) but

not for the Double condition (M=4.43, $t(36)$=1.19, p=.24).

The analyses show thus that there was a frequency effect only in the Single condition but not in the Double one. As explained in the previous section, according to Chiu et al. (2008), participants were expected to be influenced by the frequency of the contingent event in both cases, while according to Fum et al. (2008) the effect should only be present in the Single feedback. The results support our hypothesis that participants try to avoid all kind of punishments and not just the ones which result in a net loss (and are sensible to any reward and not only to wins). Because the matrices of the decks in the Single feedback condition were obtained directly from those used in the Double one, this result cannot be attributed to possible different values employed in the two conditions. On the other hand, because the SGT did not directly contrast Single vs. Double feedback, the results obtained by Chiu et al. (2008) could depend critically on the specific values used in their matrices. This experiment also suggests that participants, being sensible to the difference between Single and Double feedback, take separately into account the value of both the default and contingent event and do not rely only on the net value of each trial.

The analyses, by highlighting a goodness effect in all the conditions, show that participants are somehow sensible to the expected value of the decks, too. However, if they had really understood which decks were the good ones, they would have consistently chosen them. This did not happen because in no condition the (good) deck C was chosen more frequently than the (bad) deck B, a result that is compatible with the "prominent deck B phenomenon" normally found in traditional IGT.

The difference between the results of our experiment and those obtained with the SGT by Chiu et al. (2008) demonstrate that participants' behavior cannot be easily ascribed to the effect of a single feature. Participants could behave differently when dealing with decks which have similar qualitative features but that vary in their numerical values. Therefore, an understanding of their performance would require the use of cognitive models capable of making any feature effect an emergent property of their parameters providing thus an explanation for the influence of the qualitative features.

Modeling the results

In discussing the models of the IGT used by previous researchers, Ahn et al. (2008) identified three general assumptions: "First, an individual's evaluation of the positive and negative payoffs can be represented by a unidimensional utility function. Second, expectations about payoffs for each deck are learned on the basis of the experienced utilities on each trial. Third, these expectancies determine the choice probabilities for selecting each deck on each trial" (p. 1393). As a consequence, any model for this task, and similar iterated decision making problems, will employ at least three different functions: (1) an evaluation function to assess the payoff associated with each choice, (2) a learning function to upgrade the expectancies concerning the expected payoff of each option, (3) a selection function to choose on each trial a particular option on the basis of its expected payoff.

By adopting an architectural approach to modeling, the problem of identifying the functions necessary to replicate human performance in the task of interest is facilitated because some of these are considered as resources provided directly by the architecture. In particular, ACT-R (Anderson, 2007) makes available, by default, both a learning and a selection function. The former is given by the linear equation proposed by Bush and Mosteller (1955):

$$U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)] \tag{1}$$

where:
U_i is the utility associated with option i
n is the current time step, with $n-1$ indicating the previous one
R_i is the reward associated with option i,
and α is a parameter regulating the learning rate.

The second equation is given by:

$$P_i = \frac{e^{U}_{j}/s}{\sum_i e^{U}_{j}/s} \tag{2}$$

and determines the probability P that a given option i will be selected among the j possible options. This probability is a function of the value U (the utility, in ACT-R parlance) of the particular option compared to the sum of all the possible option values, while s is a noise parameter, analogous to the temperature of Boltzmann machines, that introduces some kind of nondeterminism in the selection process.

By having two of the three main modeling problems solved by the architecture, we concentrated on the evaluation function used to assess the outcome of each card choice. Traditionally (Ahn et al., 2008; Yechiam & Busemeyer, 2005) two different kind of functions have been employed.

The first one, called the *expectancy function* by Ahn et al. (2008), computes a weighted average of the rewards and punishment associated with the chosen option in each trial. This function can be expressed as following:

$$v(t) = (1-W) \cdot rew(t)^{\gamma} - W \cdot pun(t)^{\gamma} \tag{3}$$

with $rew(t)$ and $pun(t)$ indicating the value of the reward and punishment at time t, respectively, while γ is a parameter that determines the curvature of the evaluation function, and W denotes the differential weight participants place on losses over gains.

An alternative evaluation rule is provided by the so called *prospect function* (Ahn et al., 2008) expressed by:

$$v(t) = \begin{cases} net(t)^{\gamma} & : \text{ if } net(t) \geq 0 \\ -\lambda |net(t)|^{\gamma} & : \text{ if } net(t) < 0 \end{cases} \tag{4}$$

with $net(t)$ indicating the net outcome, i.e. the difference between the default and contingent event, and λ representing a loss aversion parameter.

The two functions are similar according to several features: they both assume a nonlinear evaluation of the monetary outcome and both weight losses differently from gains. The most important difference between them is constituted by how they take into account the default and contingent event. The expectancy function assess them separately before combining them into a scalar value; the prospect function, on the other hand, assumes that decision makers process directly the net outcome. The two functions can thus be considered as implementing the different assumptions held by Fum et al. (2008) and Chiu et al. (2008), respectively, and we used them to implement two different computational models through which we tried to replicate the empirical results. We ran a series of 500-run simulation trials with a large range of parameters and the results we obtained were quite straightforward.

Both functions are able to capture the frequency of the contingent event effect as revealed in the Single feedback condition but the prospect function, taking into account only gains and losses, is not sensitive to the effect of rewards and punishments, which also play a critical role in determining the participants' behavior in IGT, and therefore gives raise in the Double feedback condition to an effect that is absent in the experimental data. Table 5 reports the best performing models employing the expectancy (with parameters W=0.05 and $\gamma = 0.15$) and the prospect functions (with parameters $\lambda = 0.1$ and $\gamma = 0.1$) respectively. While these models have grossly similar synthetic measures of fit (for instance, RMSE= 2.35 for the expectancy and RMSE= 3.23 for prospect; chi-squared= 3.56 (p = .99) for the expectancy and chi-squared= 6.87 (p = .96) for the prospect) the prospect model fails to replicate the participants' performance by providing predictions that fall out of the 95% confidence intervals in four data points.

Table 5: Means of deck choices by the two models. The predictions which fall out of the confidence intervals are evidenced in grey.

		Deck			
Condition	**Model**	**A**	**B**	**C**	**D**
DR	Expectancy Function	25.01	24.63	24.79	25.58
	Prospect function	20.86	28.4	21.00	29.74
DS	Expectancy Function	24.55	25.03	24.93	25.5
	Prospect function	21.23	29.21	20.31	28.76
SR	Expectancy Function	20.35	28.1	20.79	30.77
	Prospect function	20.82	28.49	20.99	29.7
SS	Expectancy Function	20.22	29.59	19.84	30.36
	Prospect function	20.31	29.43	20.71	29.56

DR: Double-Reversed. DS: Double-Standard. SR: Single-Reversed. SS: Single-Standard.

Conclusions

In the paper we proposed an explanation for the frequency of the contingent event phenomenon which lies beneath the fact that people are attracted by options that are associated with the most frequent positive, and the less frequent negative, outcomes. A fundamental problem, deriving from the fact that the IGT is grounded on a conflict between the value of the default event (which codes the immediate attractiveness of an option) and the contingent one (which represents the options' long term expected value) is to establish whether this phenomenon is caused by any positive or negative outcome independently of its magnitude or, on the contrary, it is triggered by the net result deriving from the two events. The findings of our experiment corroborate the former hypothesis and the simulation results indicate that only a model sensible to rewards and punishments, and capable of analyzing them separately, can replicate the empirical data.

References

Ahn, W.-Y., Busemeyer, J. R., Wagenmakers, E.-J., & Stout, J. (2008). Comparison of decision learning models using the generalization criterion method. *Cognitive Science*, *32*(8), 1376–1402.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Bechara, A., Damasio, A., Damasio, H., & Anderson, S. (1994). Insensitivity to future consequences following damage to human prefrontal cortex. *Cognition*, *50*, 7–15.

Bechara, A., Tranel, D., & Damasio, H. (2000). Characterization of the decision-making deficit of patients with ventromedial prefrontal cortex lesions. *Brain*, *123*, 2189–2202.

Bush, R. R., & Mosteller, F. (1955). *Stochastic models for learning*. New York: Wiley.

Chiu, Y.-C., Lin, C.-H., Huang, J.-T., Lin, S., Lee, P.-L., & Hsieh, J.-C. (2008). Immediate gain is long-term loss: Are there foresighted decision makers in the Iowa Gambling Task? *Behavioral and Brain Functions*, *4*, 13.

Dunn, B. D., Dalgleish, T., & Lawrence, A. D. (2006). The somatic marker hypothesis: a critical evaluation. *Neuroscience and Biobehavioral Reviews*, *30*(2), 239–71.

Fum, D., Napoli, A., & Stocco, A. (2008). Somatic markers and frequency effects: Does emotion really play a role on decision making in the Iowa Gambling Task? In *Proceedings of 30th Annual Conference of the Cognitive Science Society* (pp. 1203–1208).

Stocco, A., Fum, D., & Napoli, A. (2009). Dissociable processes underlying decisions in the Iowa Gambling Task: a new integrative framework. *Behavioral and Brain Functions*, *5*, 1.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: Oxford University Press.

Yechiam, E., & Busemeyer, J. R. (2005). Comparison of basic assumptions embedded in learning models for experience-based decision making. *Psychonomic Bulletin & Review*, *12*, 387–402.

Cognitive Modeling of the Acquisition of a Highly Inflected Verbal System

Jesús Oliva (joliva@iai.csic.es)
José Ignacio Serrano (nachosm@iai.csic.es)
María Dolores del Castillo (lola@iai.csic.es)
Ángel Iglesias (iglesias@iai.csic.es)
Bioengineering Group. Consejo Superior de Investigaciones Científicas - CSIC.
Carretera de Campo Real, km. 0,200. La Poveda, Arganda del Rey. CP: 28500. Madrid, Spain.

Abstract

How do children cope with the general regularities that govern language while keeping track of the exceptions to them? This question has been the subject of debate for many years and it is still an open question. In particular, learning the English past tense has been studied in depth given that it is a simple problem that combines a rulelike process with many irregularities. In this paper we try to extend these studies to a quite more complex problem: the Spanish verb inflectional system. This paper presents an ACT-R model that shows the well-known U-shaped learning and mimics in many aspects the process of learning exhibited by children. Thus, our approach shows how a highly inflected morphology system can be acquired in terms of dual-mechanism theories and sheds light on the posible structures involved in general language acquisition.

Keywords: Cognitive Modelling; Cognitive Linguistics; Language Acquisition; Spanish Morphology; ACT-R

Introduction

Language acquisition has been one of the central topics in Cognitive Science. However, it is still an open question how children manage to discover the general patterns present in language while maintaining knowledge of the exceptions to them. Verb inflection has been studied not only because it is an inherently interesting task but also because is an isolable subsystem in which grammatical mechanisms can be studied in detail, without complex interactions with the rest of language. Verb inflection is independent of syntax, semantics or phonology given that no aspect of these three other subsystems works differently with regular and irregular verbs. Furthermore, the particular phenomenon of U-shaped learning that presents the irregular inflection acquisition proccess lead us to the interesting question of what are the causes for that U-shaped learning and, going beyond, how we humans deal with the general regularities that govern language while keeping track of the exceptions to them. There are two main accounts to these questions. On the one hand, the so-called dual-mechanism theories posit that knowledge is somehow dissociated. Irregular forms are stored in memory as entries in the mental lexicon while regular forms are computed by rules. On the other hand, single-mechanism theories argue that a single representational system, usually an associative memory, is enough to explain verb inflection. Both theories present some problems and thus, the controversial debate is far from settled.

English past tense inflection has been the focus of attention of many studies in the last years. However, not much work has been done to widen these studies to other languages with a much richer inflectional system. Spanish is one of these highly inflected languages. Spanish verbs can have about forty possible different suffixes (Alcoba, 1999) depending on mood, time, aspect, number or person. Moreover, this great amount of possible endings is not the only difficulty the Spanish inflectional system presents. Also its regularity is very striking compared to simpler verb systems (like that of English). In Spanish verbs, inflectional affixes are typically combined with stems and both parts of the final inflected word can be irregular. These particular features in combination with the pattern of errors presented by children suggest that the cognitive processes involved in Spanish verb inflection are more complicated than the English ones. This fact turns the modeling of Spanish verb inflections into a quite more challenging task.

In this paper we present a cognitive model of Spanish verb morphology acquisition based on dual-mechanism theories and implemented under the largely used cognitive architecture ACT-R (Anderson, 2007).

Single vs. Dual mechanism theories

Two competing classes of theories try to explain how inflected word forms are mentally represented, processed and acquired. The dual-mechanism theories (Pinker & Prince, 1988; Marcus et al., 1992; Ullman, 2001) argue that knowledge is somehow dissociated. Regular forms are built by a rule that appends an affix to the stem. Irregular forms are associatively listed in memory as entries in the mental lexicon. Within this representational framework, the three stages of U-shaped learning of irregular inflections are easily explained. In the first stage, when the regular rules are not yet available, the lexical entries of irregular forms that have been frequently heard can be retrieved. On a second stage, the regular rules are acquired and overregularization errors appear in cases in which the lexical entry for an irregular verb is not available (note that the memory retrieval process is noisy and depends on the frequency of the lexical item that is looked for). Finally, on the third stage, the overregularization errors slowly disappear as more correct examples of irregular verbs are learned. Many empirical studies have been performed that support dual-mechanism theories in many inflectional processes and some languages (Marcus et al., 1992; Clahsen, Rotweiler, Woest, & Marcus, 1992; Clahsen, Aveledo, & Roca, 2002). However, the dual-mechanism theories are still not widely accepted.

Alternative accounts are the single-mechanism theories (Rumelhart & McClelland, 1986), also called association-

ism. These approaches propose that both regular and irregular forms are computed by the same representational system, an associative memory usually modeled by a neural network. Following these theories, U-shaped learning is due to changes in vocabulary. The overregularizations occur because children have heard the regular pattern with many different verbs. So, before the first overregularization occurs, the children have to be familiar with many regular verbs. However, there is little evidence for these assumptions in empirical experiments with children. Another problem of single-mechanism models is that many of them need external feedback to adjust their weights. But actually, negative evidence (corrective feedback) plays little to no role in the process of recovery (Brown & Hanlon, 1970; Marcus, 1993), so this assumption does not seem to be adequate.

How do Spanish children inflect?

From middle 80's the acquisition of verb morphology by Spanish children has been largely investigated by many authors (Hernández-Pina, 1984; Radford & Ploennig-Pacheco, 1995; Serrat & Aparici, 1999). However, a systematic and detailed study of the development of overregularization, similar to the one carried out by (Marcus et al., 1992) for the English past tense, was not carried out until 2002 by (Clahsen et al., 2002). In this study the authors try to shed light on the question of whether or not the dual-mechanism model extends to Spanish child language. The study consisted of 64 samples of spontaneous speech from 15 children covering the age period of 1;07 to 4;07 (see (Clahsen et al., 2002) for a detailed breakdown of the data). There are longitudinal data from 4 children in the relevant age range and cross-sectional samples from 11 children.

Table 1 (extracted from (Clahsen et al., 2002)) shows the types of errors present in the children's speech and their frequency distribution.

Table 1: Distribution of error types in the study of (Clahsen et al., 2002)

A.	Stem Errors		B.	Suffixation Erros	
I.	Overregularizations	116	I.	Overregularizations (132)	
				a. 1st conj. Overapplications	8
				b. Conj.-internal regularizations	124
II.	Irregularizations	1	II.	Irregularizations	0
III.	Other errors	3	III.	Other errors	1
Totals		120	*Totals*		133

The first error type is overregularization. In such cases, an irregular stem or suffix is substituted by a regular one. As predicted by dual-mechanism theories, overregularizations are the main kind of errors that children present. Suffix overregularization errors are divided into two subtypes: overapplications of 1st conjugation suffixes to verbs pertaining to the other conjugations (for example, the second conjugation verb *tra-er*[1] (to bring) is sometimes conjugated in past as *traj-é** instead of *traj-e*, due to the 1st conjugation suffix *-é* is overapplied). The other suffix overregularization error is produced by substituting an irregular suffix by the regular suffix corresponding to its conjugation.

Also as predicted by dual-mechanism theories, irregularization errors are almost inexistent. Irregularization errors in the stem occur always with verbs that present irregular forms in the verbal paradigms for this same tense. No verb with a completely regular paradigm was irregularized. For example a child said *cay-í** (I fell) instead of *ca-í*. This is atributed to an overapplication of the third person stem (the third person inflection is: *cay-ó*) to the first person.

Making a deeper analysis of the errors, it is also important to note that the stem formation and inflectional processes are dissociated in Spanish children language. There exist mixed errors in which children combine correct irregular stems with incorrect inflectional endings (for example, to conjugate the third person singular of the immediate past of the verb *ven-ir* (to come), some children say *vin-ió** (he came) instead of *vin-o*) which is accepted to support that different processes come into play to form the two different parts of the final inflected word. This dissociation supposes a great difference with the English inflectional system. This fact significantly increases the complexity of the task and consequently, the complexity of the model compared to other similar models of the English past tense (Taatgen & Anderson, 2002).

U-shaped learning

The study of (Clahsen et al., 2002) clearly extends to Spanish the results obtained by (Marcus et al., 1992) for English. The development of irregular verb acquisition is not guided by a linear learning function but by a U-shaped learning function in which three stages can be clearly distinguished.

In a first stage, the child is able to inflect very little verbs but the inflected irregular verbs are correct. In a second stage, the children have acquired some kind of knowledge about the regular rule and start to overapply it to irregular verbs. In the third stage, the overregularization errors diminish until mastery is achieved. The learning of regular verbs is quite simpler. Children start inflecting correctly a very low number of regular verbs and their performance steadily grows until they master the task.

The model

In this paper we propose a dual-mechanism model implemented in the ACT-R cognitive architecture. The core components that are used for the model, including the declarative and procedural memory systems, are parts of the ACT-R architecture, which has been largely validated through extensive separate experiments not only related to language. Moreover, the main processes used, like instance-based learning and the use of analogy, are part of the ACT-R modeling tradition. The two basic strategies of memory retrieval and analogy are neither specific to the task of producing a past tense nor even specific to language but general domain cognitive strategies:

[1] Stem and suffix are shown separated in Spanish verb forms.

- Memory retrieval: This strategy simply consists in retrieving a fact from declarative memory.

- Analogy: This strategy forms the required knowledge using a similar retrieved fact as a template. As stated by (Salvucci & Anderson, 1998), analogy is probably one of the dominant human strategies for problem solving and discovery.

It is important to note that the strategies we suppose that children have at the moment they start learning a language are very basic strategies common to many cognitive tasks. Note that, at the beginning, the proposed model has nothing similar to a regular rule to inflect regular verbs. The proposed model will learn them later on as a specialization of the analogy strategy. These initial strategies are similar to the ones proposed by (MacWhinney, 1978; Taatgen & Anderson, 2002), who claimed that the basis of the learning of the regular rules is analogy.

Detailed description

The two main components of the model are described as declarative-memory chunks and production rules. The declarative-memory chunks represent verb forms as follows.

VERB-FORM	
ISA	VERB-TENSE
INFIN	CANTAR
CONJ	AR
INFIN-STEM	CANT
MTA	IND-PAST-PERF
NP	S3
STEM	CANT
SUFFIX	Ó

The chunk is of type VERB-TENSE. Its infinitive is *cant-ar* (to sing) and the infinitive stem and conjugation are *cant-* and *-ar* respectively. Moreover, given the characteristics of the Spanish verb inflectional system, it is necessary to store the mood, time and aspect of the verb form (in the slot MTA, the value IND-PAST-PERF stands for indicative mood, past tense, perfective aspect) and the number and person of the represented verb form (in the slot NP, the value 3S stands for third person, singular). The verb form corresponding to the information represented on the precedent slots is represented by the STEM and SUFFIX slots. Note that when the goal is to obtain a verb form, these two slots start with a NIL value and the task of the model is to fill them.

Procedural memory stores the strategies that guide the inflection process. As stated before, two basic strategies are the core of the model. However, given the dissociation between stem formation and inflectional processes that Spanish verb inflection presents, these strategies are also dissociated in different rules that try to form the stem or to find the correct suffix. The main rules of the model are:

- Rule 1 (verb form retrieval): When the model tries to find the verb form of a given verb with given MTA and NP slots, this rule simply tries to find a chunk in declarative memory that shares the INFIN, MTA and NP slots with the given one.

- Rule 2 (stem retrieval): This rule tries to find the stem of the goal verb form. To do that, it looks for a chunk in the declarative memory with the same INFIN and MTA slots.

- Rule 3 (stem analogy): When the model tries to find the verb form of a given verb, this rule just copies the INFIN-STEM of the goal verb form on the STEM slot only if the INFIN-STEM and the STEM slots of an arbitrary retrieved (i.e. the verb with a highest activation) verb are the same.

- Rule 4 (suffix analogy): This rule tries to find out the correct suffix of the goal verb form. To do that, it looks for a chunk in the declarative memory with the same CONJ, MTA and NP slots and, if the slots INFIN-STEM and STEM of the retrieved form are the same, it copies the value of the SUFFIX slot to the SUFFIX slot of the goal verb form.

These four rules cover the two basic strategies of the model and the two processes that Spanish speaking people are supposed to use when trying to inflect a verb. Figure 1 shows the processes that our model uses to inflect a verb. Dashed lines means that these processes are not available when the model starts working but they are learnt during the running.

Learning in ACT-R consists in the production of new rules. New rules are created by collapsing two rules that are applied in succession into a single rule. The basic idea is to combine the tests in the two conditions into a single set of tests that will recognize when the pair of productions can be applied. Also the actions of both rules are combined into a single action that will have the effect of both. The resulting rule is therefore a specialization of the two parent rules. The specialization, which is of particular interest, occurs when Rule 3 (stem analogy) fires first and Rule 4 (suffix analogy) fires secondly. In this case, the corresponding suffix is substituted into the rule, producing one of the regular rules. For example:

IF	the goal is to inflect a verb with CONJ = 'AR' MTA = 'IND-PAST-PERF' NP = 'S3'
THEN	set the SUFFIX slot to 'Ó' copy the INFIN-STEM slot to the STEM slot

Note that one of these rules has to be learned for each combination of the values of the slots CONJ, MTA and NP, given that each regular suffix is different. Also it is important to note that the initial utility of the learned rules is very low. This means that newly created rules are not used just after being learned. It is necessary to reinforce the utility of this rule. This reinforcement occurs every time the rule is recompiled because its two parents fire consecutively. This way, the most

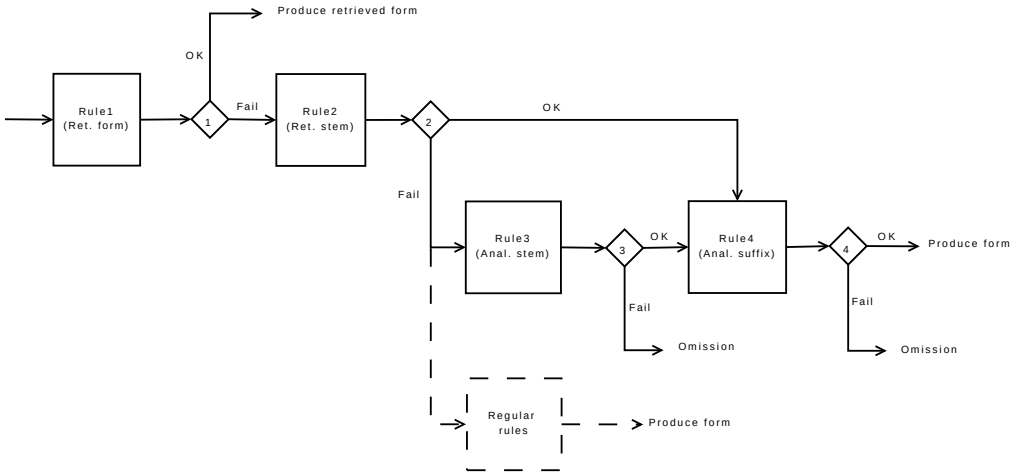


Figure 1: Processes used by the model. Dashed lines show processes that have to be learnt.

useful rules (the ones that are recompiled many times) are finally used by the model and those rules created just by chance are practically forgotten by the model. Moreover, ACT-R provides a way by which useful rules are reinforced: utility learning. This process reinforces the rules that have been used to reach to a specific inflection. When the model cannot inflect a verb, it propagates a lower reward than the one it propagates if the verb is inflected. This seems to be natural given that, when the model could not inflect a verb, it could not "say" what he wanted to "say". However, the reward received when a verb is inflected incorrectly is exactly the same as the one that is received when a verb is inflected correctly given that the model cannot know whether his production is correct or not. Note that one of the most important criticism to many connectionist models is that they need some kind of external feedback while, as stated before, it is widely accepted that children do not receive feedback when talking to their parents. Thus, the unique feedback our model receives comes from itself.

How does the model inflect?

Data and Procedure

The data we used as the input for the model consists of the verbs contained in the Spanish Verb Inventory[2] (SVI, (Rivera, Bates, Orozco-Figueroa, & Wicha, 2009)) which is made of 50 of the earliest acquired common Spanish verbs, with conjugations across person, number and 4 verb tenses (imperfect, immediate past, future, and present indicative), for a total of 920 unique verb forms. Future tense forms were discarded given its low frequency of use on child language and also imperfect forms were discarded given that they do not present almost any irregularity. So the final input for the model consists of the 220 immediate past forms and the 250 present tense forms of the Spanish Verb Inventory. Each of these forms has its associated frequency of use on children language.

In order to perform the different experiments we followed the design given by (Taatgen & Anderson, 2002). Every 200 simulated seconds two words are presented for perception and one word is selected for generation. These words are selected based on the frequency distribution given in the SVI. Also following the design of (Taatgen & Anderson, 2002), in each simulated month, approximately 1300 past tenses are produced. This number is chosen somewhat arbitrarily, but the model is not critically dependent on the exact rate of production.

Results

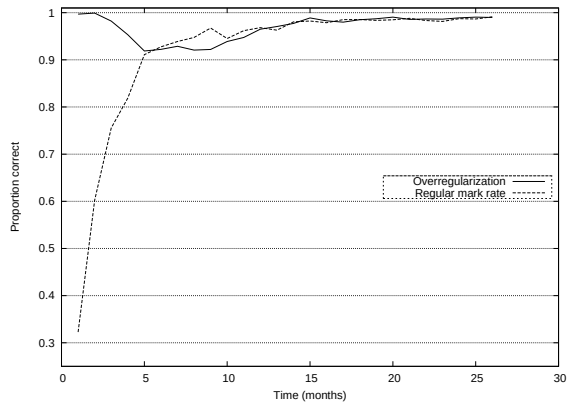
As stated before, the great majority of errors done by children are overregularization errors while only a few errors were due to irregularization of regular forms. According to (Clahsen et al., 2002), more of the 90% (94.7% in the stem and 92.5% in the suffixes) of the errors done by children are overregularization errors. Our model also presents a similar unbalanced distribution of errors between irregular and regular forms. The 93.3% of errors were overregularization errors. Moreover, the irregularization errors are mainly of the same kind of the ones done by children. As stated before, no verb with a completely regular paradigm was irregularized.

Figures 2(a) and 2(b) show the learning curves of the model and of María, one of the children from the study of (Clahsen et al., 2002) (It is important to note that the other children on that study have similar learning curves). Figure 2 shows the overregularization rate and the regular mark rate as they are usually plotted. Overregularization equals the number of correct responses on irregular verb forms divided by the sum of correct irregulars and irregulars inflected regularly. The regular mark rate shows the number of correctly inflected regulars divided by the total number of regulars produced. The development of the model clearly shows the U-shaped learning curve typical of children's learning of irregular verbs. As such, the results are quite similar to the ones of María. Our

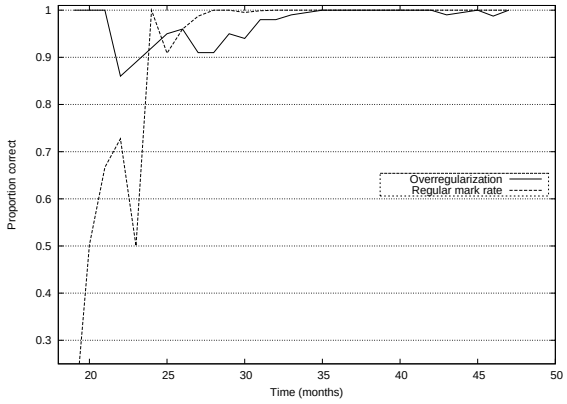
[2] Accesible at `http://crl.ucsd.edu/experiments/svi/`

model obtains a global 3.9% of overregularization, which is in line with children's performance. Spanish children studied by (Clahsen et al., 2002) present an average overregularization rate of 3.4% in the longitudinal samples and a 13.2% in the cross-sectional experiments. As pointed by (Clahsen et al., 2002) this difference could be due to the type of samples and the semi-structured style of the records.

Not only overregularization errors of our model are similar to the ones done by children. The percentage of irregularization errors done by our model was 0.5% while in children, overregularizations amount to 0.4% and in both cases no verb with a completely regular paradigm was irregularized.



(a)



(b)

Figure 2: Overregularization and regular mark rate presented by the model (a), and by María (b)

In order to better understand why U-shaped learning is achieved, we should go through the model's functioning in some more detail using some examples of irregular and regular vebs: a very frequent irregular verb form such as *pued-e* (he can) with a frequency in the SVI of 19269, a very frequent regular verb form such as *deb-e* (he should) with a frequency of 6955, a low frequency irregular verb form such as *jueg-an* (they play) with a frequency of 201 and a low frequency regular form such as *salt-a* (he jumps) with a frequency of 252.

At the beginning the model has no regular or irregular examples, so it fails every time it tries to inflect a verb. Gradually, high-frequency irregular verbs increment its activation on the declarative memory. If the model tries to inflect one of these high-frequency verbs, the retrieval strategy will find the correct form on declarative memory. On that first stage analogy usually fails given that it needs a regular form to work as a template. Regular forms are not as frequent as irregular forms (see that *deb-e* has a frequency of almost a third of the frequency of *pued-e*) and their activation is lower and so, analogy is not available on a first stage. Thus, verb forms such as *deb-e* or *salt-a* cannot be inflected. Moreover, there are no overregularization errors given that the source of overregularizations is also analogy. These facts explain the first stage of the U-shaped learning.

After some examples have been learned the number of regular verbs with enough activation in memory steadily grows up. Analogy is now a viable strategy, as there are examples that can be retrieved as templates. These uses of analogy lead to eventually learn the regular rules. However, most of the regular rules are not yet used given that its initial utility is not sufficiently high. At this stage, if the model has to inflect the form *jueg-an*, it is very probable that the retrieval strategy fails given its low frequency. If analogy finds suitable regular forms in declarative memory (suppose, for example, that the regular form *cant-an* (they sing) has enough activation) the model will produce the overregularization *jug-an**. Thus, at this stage overregularizations start to appear. However, they are still not very frequent because the regular forms that are used by analogy are not very frequent in memory and the regular rules do not have enough utility to be fired.

As analogy continues working, the utility of the regular rules increases to a point in which they start to be used. At this point, the rate of overregularizations, which start to appear on the previous stage, reaches a maximum. In the previous stage, verb forms such as *jueg-an* are rarely overregularized because analogy needs to retrieve a regular form from memory (and usually an irregular form is retrieved given that they are more frequent). However, regular rules do not need to retrieve a regular form. Thus overregularizations are much more frequent at this stage in which regular rules have a higher utility. For the same reason, the rate of correctly inflected regular forms highly increases. On previous stages, low frequency regular forms such as *salt-a*, could not be inflected because the retrieval strategy failed and it was difficult to find a regular form to do the analogy with the stem and another regular form to do the analogy with the suffix. As regular rules do not need any memory retrieval, the model just has to fire the corresponding regular rule to correctly inflect the form *salt-a*. From this point on, analogy strategy will be used very rarely, as it has to compete with the regular rules that become now the backup strategy given that they are more efficient.

On the last stage, irregular forms are stored in declarative memory with a sufficient and stable activation. This way, every time the model has to inflect an irregular form such

as *pued-e*, the retrieval strategy works blocking the regular rule. Moreover, high-frequency regular forms such as *deb-e* have a high activation at declarative memory and so, when the model has to inflect one of these forms, retrieval will be successful again. Regular rules will be used with medium and low-frequency regulars such as *salt-a*. Medium and low-frequency regulars have a lower activation and so, retrieval usually fails and they have to be inflected by the regular rules. At this point the utility of the regular rules is also high and stable, so analogy is hardly used anymore. When this stage is reached, one may judge that the model has mastered the task.

Conclusions and future work

In this paper we have presented a cognitive model of Spanish verb morphology acquisition based on dual-mechanism theories. The model we present is based on two basic strategies that neither are specific to the task of producing a past tense nor even specific to language. In fact they are general domain cognitive strategies such as memory retrieval and analogy. The core components that are used for the model, among which are the declarative and procedural memory systems, are parts of the ACT-R architecture, or part of the ACT-R modeling tradition, like instance-based learning and the use of analogy. Starting from these general strategies, the model learns the regular rules of the Spanish inflectional system while it takes into account the exceptions that represent irregular verbs. The results show that our model accomplishes to fit properly the U-shaped learning curve and some other typical aspects of the process of learning exhibited by Spanish children. Thus, our approach shows how a highly inflected morphology system can be acquired in terms of dual-mechanism theories and sheds light on the possible structures involved in general language acquisition.

Future work includes extending the declarative and procedural representations to take into account phonetic features that allow modeling the phonetic analogy processes that seem to be present in some cases. Moreover, the model could be extended to other tenses and to a wider range of ages in order to accomplish a general view of the complete process of Spanish morphology acquisition. Other trends of future work could be related to language impairments. This model could be used to model some of these impairments by modifying some of the parameters of the model. This way we can give some arguments in favor of the different hypothesis about the causes of these impairments just as these models can be used to propose some kind of therapies or methods to improve the acquisition of verb morphology and general language skills. Finally, it would be very useful to extend the existing empirical studies with children to have more data from which we can extract more general conclusions.

Acknowledgments

The authors would like to thank Prof. Harald Clahsen and Prof. Fraibet Aveledo for making available the full contents of their study with Spanish children.

References

Alcoba, S. (1999). Gramática descriptiva de la lengua española. In (pp. 4915 – 4991). Madrid: Espasa-Calpe.

Anderson, J. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Brown, R., & Hanlon, C. (1970). Derivational complexity and order of acquisition in child speech. In J. Hayes (Ed.), *Cognition and the development of language.* New York.

Clahsen, H., Aveledo, F., & Roca, I. (2002). The development of regular and irregular verb inflection in spanish child language. *Journal of Child Language*(29), 591 – 622.

Clahsen, H., Rotweiler, M., Woest, A., & Marcus, G. (1992). Regular and irregular inflection in the acquisition of german noun plurals. *Cognition*(45), 225 – 255.

Hernández-Pina, R. (1984). *Teorías psicolingüísticas y su aplicación a la adquisición del español como lengua materna.* Madrid: Siglo XXI.

MacWhinney, B. (1978). The acquisition of morphophonology. In *Monographs of the society for research in child development.* (Vol. 1, pp. 1–123).

Marcus, G. (1993). Negative evidence in language acquisition. *Cognition*(46), 53 – 85.

Marcus, G., Pinker, S., Ullman, M., Hollander, M., Rosen, T., & Xu, F. (1992). Overregularization in language acquisition. In *Monographs of the society for research in child development.* (Vol. 57, pp. 1 – 182).

Pinker, S., & Prince, A. (1988). On language and connectionism: analysis of a distributed processing model of language acquisition. *Cognition*(28), 73 – 193.

Radford, A., & Ploennig-Pacheco, I. (1995). The morphosyntax of subjetcs and verbs in child spanish: a case study. In *Essex research reports in linguistics* (Vol. 5, pp. 23 – 67).

Rivera, S., Bates, E., Orozco-Figueroa, A., & Wicha, N. (2009). Spoken verb processing in spanish: An analysis using a new online database. *Applied Psycholinguistics*, Accepted.

Rumelhart, D., & McClelland, J. (1986). On learning the past tense of english verbs. In J. McClelland & D. Rumelhart (Eds.), *Parallel distributed processing: explorations in the microstructure of cognition* (pp. 216 – 271). Cambridge, MA: MIT Press.

Salvucci, D., & Anderson, J. (1998). Analogy. In J. Anderson & C. Lebiere (Eds.), *The atomic components of thought* (pp. 343 – 383). Mahwah, NJ: Erlbaum.

Serrat, E., & Aparici, M. (1999). *Morphological errors in early language acquisition: evidence from catalan and spanish.* (Unpublished ms., Universities of Girona and Barcelona)

Taatgen, N., & Anderson, J. (2002). Why do children learn to say ”broke”? a model of learning the past tense without feedback. *Cognition*(86), 123 – 155.

Ullman, M. (2001). A neurocognitive perspective on language: the declarative/procedural model. *Nature Reviews*, 717 – 726.

Building Large Learning Models with Herbal

Jaehyon Paik[1], Jong W. Kim[2], Frank E. Ritter[3], Jonathan H. Morgan[3], Steven R. Haynes[3], Mark A. Cohen[4]

[1]Department of Industrial and Manufacturing Engineering
[3]College of Information Science and Technology
Pennsylvania State University, University Park, PA
[2]Department of Psychology, University of Central Florida, Orlando, FL
[4]Business Administration, Computer Science, and Information Technology, Lock Haven University, Lock Haven, PA

<jaehyon.paik, frank.ritter, jhm5001>@psu.edu, jwkim@mail.ucf.edu, shaynes@ist.psu.edu, mcohen@lhup.edu

Abstract

In this paper, we describe a high-level behavior representation language (Herbal) and report new work regarding Herbal's ACT-R compiler. This work suggests that Herbal reduces model development time by a factor of 10 when compared to working directly in Soar, ACT-R, or Jess. We then introduce a large ACT-R model (541 rules) that we generated in approximately 8 hours. We fit the model to learning data. The comparison indicates that humans performing spreadsheet tasks appeared to start with some expertise. The comparison also suggests that ACT-R, when processing tasks consisting of hundreds of unique memory elements over times spans of twenty to forty minutes, may have problems accurately representing the learning rates of humans. In addition, our study indicates that the spacing between learning sessions has significant effects that may impact the modeling of memory decay in ACT-R.

Introduction

In this paper, we discuss the rapid development of user models capable of dynamically representing behavioral constraints. Pew and Mavor (eds., 2007) advise using such user models as a shared representation meant to identify, predict, and when possible, mitigate risks. These representations are of various kinds (qualitative, quantitative, analytical, computational), and can describe interactions operating within or across multiple levels of analysis. These models in their various forms have proven useful in predicting and preventing significant losses whether human (e.g., Byrne & Kirlik, 2005; Pew & Mavor, 2007) or monetary (e.g., Gray, John, & Atwood, 1993) or both (e.g., Booher & Minniger, 2003).

There is a rich literature in user models. Classic user studies beginning with Card, Moran, and Newell's (1983) book have often represented psychological/behavioral constraints using the GOMS model; analyzing user behavior in terms of goals, operators available for accomplishing those goals, routinized sequences of behavior or methods, and rules for the selection of methods for instances where multiple methods apply. Grey et al. (1993) extended and validated the GOMS model through an empirical study of telephone operators working for the New England Telephone Company, introducing CPM-GOMS. The success of later-implemented versions of the GOMS model (John & Kieras, 1996; Kieras, Wood, Abotel, & Hornof, 1995), TAC-AIR Soar (Jones et al., 1999), and of embodied cognitive architectures generally (Byrne, 2001; Byrne & Gray, 2003; e.g., Ritter & Young, 2001; St. Amant, Horton, & Ritter, 2007) has intensified interest in agent-based user models for testing interfaces and for working in simulations as opponents and colleagues.

On the other hand, these efforts have been stymied in part by the significant integration costs and the detailed level of specification required by existing cognitive architectures to create models. While one of cognitive modeling's great strengths is its demand for computational entailment, the low-level abstractions required by mature cognitive architectures such as Soar and ACT-R have frequently proven expensive to create, resulting in a fewer models being created. Furthermore, these models have often proven difficult to maintain, extend, or merge (Pew & Mavor, 1998; 2007; Ritter et al., 2003).

Recognizing these issues, developers in recent years have released both re-implemented versions of Soar and ACT-R in Java that may be easier to integrate into systems, as well as creating high-level cognitive modeling languages that seek to provide a common framework and formal language for a variety of essentially similar cognitive modeling tasks (a review is available, Ritter et al., 2006). In the next section, we will briefly review these efforts before introducing Herbal (High-Level Behavior Representation Language). We will then discuss recent work on Herbal's ACT-R compiler and a large learning model we have generated and tested before concluding.

Related Work

Cognitive architectures realized as programming languages, as noted above, have operated at low-levels of abstraction, and consequently have made developing, implementing, and comparing cognitive models difficult. Two general approaches have emerged to address this problem, the reimplementation of existing languages and the development of high-level cognitive languages for these architectures. We will describe both briefly before discussing Herbal.

Re-implementing cognitive modeling languages

Reimplementation of existing cognitive languages into newer object-oriented languages offers several advantages: (a) smoother integration into systems created in those widely used languages, such as Java, supported by extensive libraries and tools; (b) a perceived and sometimes greater degree of implementation modularity, and thus the ability to more easily investigate changes and extensions to existing cognitive architectures; and (c) the opportunity to make comparative analyses, and thus discern the effect that previous implementation choices as opposed to theoretical commitments have had on the language in question. jACT-R (Harrison, 2002) and jSoar (Ray, 2009) have both contributed interesting comparative analyses, offer an array of GUI based debugging and organizational tools, and can increasingly support ongoing work in simulations and agent-based tools.

jACT-R and jSoar both rely heavily upon the syntax of their parent languages to represent the rules and knowledge, limiting their accessibility to some extent. Though Python ACT-R (Stewart & West, 2005) eliminates this syntax issue, all three languages are at various stages of completeness, and none to our knowledge has undergone extensive validation through a computational alignment or docking study (Axtell et al., 1996) or similar means (e.g., Burton, 1998; Louie, Carley, Haghshenass, Kunz, & Levitt, 2003). In addition, re-implemented cognitive modeling languages are neither able to support the comparative analysis of models across cognitive architectures, nor the fine-tuning of architectures at a constant high-level of abstraction. Thus, high-level cognitive modeling languages are attractive.

High-level cognitive modeling languages and approaches

High-level cognitive languages use abstractions to generalize common structures and processes found in existing cognitive architectures. These persistent commonalities are evident when one considers defining a high-level knowledge representation, building a structured task analysis, or implementing a decision cycle characterized by the perceive-decide-act mechanism (Newell, Yost, Laird, Rosenbloom, & Altmann, 1991). Cognitive architectures' shared dependence upon *least commitment* (or the making of control decisions at every decision point) and *associative encoding* (or the associative retrieval of potential courses of action and a conflict resolution process for choosing between solution paths) entail a set of core commonalities from which to abstract. The commonalities include: a declarative memory structure and retrieval method, goals, procedural memory frequently used for the achievement of those goals, mechanisms for responding to external events, and a iterative decision process (Jones, Crossman, Lebiere, & Best, 2006).

Where these approaches differ is in their representation structures. We will briefly summarize two existing candidate approaches for modeling more complex cognitive models: Jones et al.'s (2006) High Level Symbolic Representation Language (HLSR), and Herbal, a High-Level Behavior Representation Language (Cohen, Ritter, & Haynes, in press; Haynes, Cohen, & Ritter, 2009).

HLSR uses three primitives (relations, transforms, and activation tables) to derive micro-theories for representing cognitive architectures (and by extension, cognitive theories). Herbal characterizes common cognitive modeling tasks such as task analyses and problem solving using an ontology based upon the Problem Space Computational Model (PSCM, Newell et al., 1991). Each of these approaches is promising; each potentially allows for comparative analysis across architectures; and each, if fully developed, could promote model reuse across a diverse community of users.

Herbal's user focus, however, is unique in this area. HLSR supports both Soar and ACT-R, but is not yet available outside of its developers, and has, to our knowledge, not undergone either a docking or a usability study. Herbal, in contrast, is open source; supports three cognitive architectures across a set of common cognitive modeling tasks (Soar, ACT-R, and Jess); has undergone two usability studies (Cohen 2008; Cohen, Ritter, & Haynes 2009); has been used to create several models; and is currently undergoing a docking study. Next, we will describe Herbal and work related to Herbal more fully, focusing on Herbal's implications for HCI and the more rapid creation of user models.

Herbal

Herbal is based on the PSCM (Newell et al., 1991). Herbal's ontological representation defines behavior as the movement of operators modifying states, as well as movement through problem spaces. Within this framework, behavior is divided into bands of activity operating across three time scales: the elaboration cycle (10 ms), the decision cycle (100 ms), and activity occurring within a problem space (1 s). The elaboration cycle describes the process by which an agent modifies its state representation through the associative retrieval of information. The decision cycle in turn consists of repeated cycles of elaboration that persist until quiescence, or until no further productions can be fired. The levels of elaborations are, for the most part, hidden in and by Herbal.

The agent makes decisions based upon its state interpretation and preferences, choosing either a unique operator (actions capable of transforming the state) or generating an impasse if an operator cannot be selected due to insufficient knowledge. Agents resolve impasses by generating sub-states that enable the agent to retrieve the information necessary to specify the next operator. Problem spaces are thus representations describing a sequence of decisions (or a search in the event of limited knowledge) that can be further defined in terms of goals, states, and operators.

Herbal's ontology characterizes behavior in terms of classes that represent concepts such as states, operators, elaborations, impasses, conditions, actions, and working

memory. These classes furthermore entail basic relationships for instance—states can contain impasses, working memory, operators, elaborations, and other states while operators and elaborations can contain zero or more conditions and actions. Programming in Herbal thus involves instantiating objects using these ontological classes. Herbal also supplies additional attributes that enable future developers to discern the intent motivating creation of a given object, supporting models that in essence explain themselves (Haynes, Cohen, & Ritter, 2009).

The Herbal/ACT-R compiler

We have created an initial version of an ACT-R compiler in Herbal. Although several easy-to-use frameworks exist to develop ACT-R models: CogTool (John, Prevas, Salvucci, & Koedinger, 2004), ACT-Simple (Salvucci & Lee, 2003), and G2A (St. Amant, Freed, & Ritter, 2005), these tools cannot represent models of greater complexity than KLM-GOMS or GOMS models.

To support modeling in ACT-R, we added a declarative memory component to the Herbal environment because ACT-R uses declarative memories. With this component, we were able to also add hierarchical and sequential tasks to an ACT-R model—the relations among tasks are shown in a tree form in the user interface. Herbal then makes memories and production rules based on these relationships. Furthermore, to explore the flexibility of the high-level compiler, we added an ACT-R parameter pane. Through this pane, users can generate either a novice ACT-R model or eleven kinds of expert ACT-R models with varying degrees of expertise ranging from 0% to 100%.

The Herbal/ACT-R compiler takes the PSCM representation in Herbal and creates an ACT-R model from it. The compiler also uses these parameters to determine how to compile the model: as a novice, an expert, or somewhere in between. When implementing a task, we represented the level of expertise, or degree of proceduralization, as corresponding to the percentage of declarative memory retrievals necessary to complete the task. We then distinguished novice from expert models by this percentage. Novice models, in this framework, have no information regarding the next task step in procedural memory, and thus must retrieve each step from memory, whereas the expert models have the next task step incorporated as part of the operation. Novice models thus provide the maximum anticipated completion time while *normative expert* models (described below) provide the hypothetical minimum time.

Distinguishing novice from expert, we further divided expert models into two types: (a) *normative experts*, models where all the declarative memory elements for the task have been compiled into procedural knowledge, and (b) *practicing experts*, models that exhibit varying degrees of proceduralization. Models exhibiting 100% expertise (*normative experts*) provide a baseline, and do not use memory elements in declarative memory to perform the task because we assume that and the model has these elements fully proceduralized. Models ranging between 0% and 90% expertise (*practicing experts*) have a proceduralized task structure, but the number of declarative memory retrievals to walk the task structure varies. For example, if a model should be represented as having 10 declarative memories (DMs), the 0% expertise model would have 10 DMs while the 10% expertise model would have 9 DMs and 1 rule, and so on. *Practicing expert* models thus provide us a basis for making useful comparisons with the human data by providing incremental predictions of performance based upon expertise, and perhaps enable us to isolate the participants' actual average level of expertise at the onset of the trial.

A test of the Herbal/ACT-R compiler

To explore the Herbal ACT-R compiler, we implemented an ACT-R model using Herbal and compared the performance times with practice provided by the model with those of human participants performing the same series of tasks.

The Dismal spreadsheet task

We next provide a brief description of the participant data before discussing the model and its implications. For the purposes of examining variance in retention rates over time, Kim (2008) devised a sequential spreadsheet task consisting of 14 subtasks that participants learned using one of two different modalities (keyboard or vertical mouse). In addition, Kim examined what if any influence training intervals have on retention rates by comparing the performance of participants undergoing training at 6, 12, and 18-day retention intervals. These results are discussed in a forthcoming publication.

For this comparative analysis, we used a subset of Kim's data, modeling the decline in task completion times for all 14 subtasks over a four training sessions. Over the training iteration's time course (about 30-45 min. per session), Kim found that the average task completion time for participants (N = 30) using a vertical mouse to perform the spreadsheet task ranged from 1,366 s (SE = 60.76 s) on day 1 to 655 s (SE = 22.81 s) by day 4.

The change in performance over the four-day trial is as anticipated, a relationship between performance and practice ($y = 1339.7x^{-0.5}$, R^2 = 0.99) that follows the power law of learning. Examining the curve's progression, one also sees the final value is similar to the anticipated KLM-GOMS (Card, Moran, & Newell, 1983) value of 797.14 s for expert performance.

Modeling the Dismal spreadsheet task

Paik and Kim, working collaboratively, implemented the spreadsheet task model in ACT-R in four hours using Herbal. The resulting novice model consists of 9 rules and 542 declarative memory elements; the fully expert model consists of 541 ACT-R rules and no declarative memory elements; and the intermediate, practicing expert models interpolate between these two models. For example, the

50% expert model has 271 declarative memory elements and 541 ACT-R rules, and the 0% expert model has 542 declarative memory elements and 541 ACT-R rules.

The rate of development using Herbal was about 0.9 minutes per rule (240 minutes x 2 programmers/541 rules) in the expert model. This is approximately 20 times faster than writing in Soar (assuming that an ACT-R rule is approximately the size of two Soar rules to propose an operator and then to implement it), and also about 5 times faster than rates reported by Yost (1992) using TAQL (again, assuming that the ACT-R rules are larger). Unfortunately, we know of no comparative usability studies for ACT-R, but we suspect that Herbal also accelerates the rate of developing ACT-R models.

Results

Running the 12 models in ACT-R 6, we confirmed that the novice, expert, and intermediate models perform the task. We compared the performance rates over time provided by the model with those of human participants performing the same task. (In addition to the ACT-R model's times, we added interaction times for mouse moves and key presses). All the models also learn, with the novice models learning the most and the expert model the least. The predicted times are comparable both to the GOMS and KLM models, and to the data collected by Kim (2008).

Because the data was taken over multiple trials, the comparison becomes more interesting because we can use the model to predict the participants' levels of expertise at the onset of the first trial. By comparing the learning curves of the model with that of the 40 participants who performed the Dismal spreadsheet task (as depicted in Figure 1), we found that human completion times for the first trial corresponded with an expertise level of 20%, 60% at trial 2, 80% at trial 3, and a gradual increase up to full expert by the fourth trial. We thus see that the human performance data represents a faster learning curve than that displayed by any of the ACT-R models.

The difference between the two curves indicates that the model's learning rate remains too slow, as opposed to the participants' expertise being either too high or too low to be matched. Though the learning displayed by the model is already surprisingly fast and robust, these results suggest not only that the model will have to learn faster but also that it may have to include a new learning mechanism. For example, the learning rate exhibited by the human data between the first and second trials shows a sharp decline, meaning that the participants acquired more knowledge in the first trial than the model, and that this learned knowledge was already sufficiently activated to use for performing the task. The current Herbal/ACT-R models, in contrast, predict a more gradual learning curve. While the existing model includes declarative strengthening and procedural learning, another type of learning or stronger parameters on the existing learning mechanisms may be necessary.

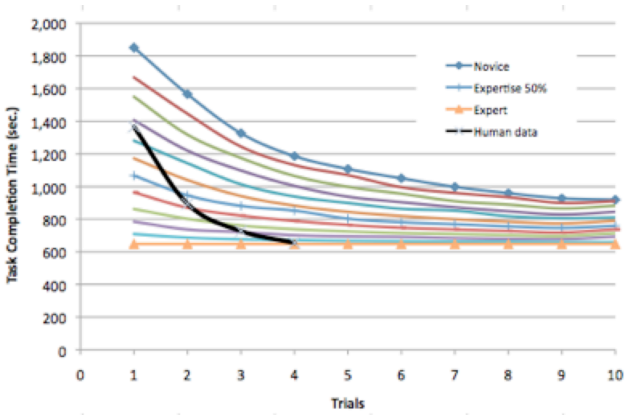


Figure 1: The human data shown with respect to the model's learning curves.

The results in Figure 1 suggest further, deeper problems as well. ACT-R does not appear to easily support modeling declarative memory decay, and the participants' learning sessions were separated by at least a day. If, for example, we attempted to model massed training (concentrated training blocks), the difference between the model's performance and that of the participants is likely to be have been even greater because no memory decay would have occurred, and if we modeled the effect of days between learning trials, the model would learn more slowly, matching the data less well.

Nevertheless, this model is unique in that we are able to begin to conduct these comparative analyses, and perhaps may eventually be better able to ascertain the participants' actual initial level of expertise for sequential tasks. Future work includes individual data fits, exploring these deep problems of decay, and devising ways to achieve faster learning.

Discussion and conclusions: Implications for future user models

To conclude, we would like to discuss four implications for modeling learning. First, the novice model and the expert models use different approaches to organize the task knowledge that then results in different task completion times. Novice models use a tree structure to organize declarative memories (each declarative knowledge element has a parent, a next-sibling, and a first-child); to walk this structure, the model uses a depth-first search approach. All the expert models, however, use a sequential representation of the declarative memory structure, in other words each of the declarative memory elements has its next step, so the models can walk through the entire structure by following the next step. This is the difference between the top 2 lines, novice and 0% expert models, in Figure 1.

Second, our model's representation of expertise differs from ACT-R. We represented expertise as a function of model's number of declarative memory elements. For example, the 0% expertise model, our novice, has 542 declarative memory elements while our normative expert model (100% expertise) has no declarative memory

elements. Consequently (at least as presently compiled), the number of retrieved declarative memory elements gradually decreases as expertise increases. The normative expert model, thus, does not retrieve its declarative memory elements to perform the task. ACT-R, on the other hand, represents experts with production compilation (the process of generating a new rule by combining two or more rules). So, the number of fired rules gradually decreased, but those rules still need to retrieve declarative memory to perform a task.

Third, we have presented a high-level cognitive modeling language that allows for the rapid development of complex user models. As we noted in the introduction, one reason why agent-based user models have not been more widely adopted is because of the relative difficulty associated with developing them. Cognitive architectures such as ACT-R and Soar use a low-level knowledge representation language that makes developing user models appear intractable to non-experts. Herbal, in contrast, is based on the Eclipse that is well-known development tool and provides graphical user interface, so it enables users to make three different kinds of cognitive models, such as Soar, Jess, and ACT-R, more easily. In addition, Herbal provides models that explain themselves by providing answers to questions that users frequently ask (Haynes, Cohen, & Ritter, 2009).

Nevertheless, we acknowledge that Herbal is far from mature, and that we will most likely have to refine our ontology further to fully support ACT-R. We also have to extend the Herbal/Soar compiler to use the task hierarchy pane; and we have yet to compare Soar and Jess models developed in Herbal to human data.

Fourth, the models we have developed with Herbal suggest new model types and new uses for models. A model (Herbal/Soar/Diag) includes a large number of strategies (M. B. Friedrich, 2008; M. B. Friedrich & Ritter, 2009). Another model (Herbal/ACT-R/Dismal) is perhaps the largest ACT-R model (as measured by rule count) created thus far. It is large partially because it performs a non-repetitive task. Many previous models have performed a repetitive task taking minutes to do (e.g., processing 100 planes). Doing a long non-repetitive task, however, requires creating a large knowledge set that has many components that are only used once.

While Herbal remains in some ways a modest step, it opens up new modeling approaches where a broad range of relatively shallow knowledge is needed, but within a cognitive architecture, and where learning is important.

Acknowledgements

This work was supported by the U.S. Office of Naval Research (ONR) under contract N00014-06-1-0164, N00014-09-1-1124 and the Defense Threat Reduction Agency under contract 1-09-1-0054.

References

Axtell, R., Axelrod, R., Epstein, J. M., & Cohen, M. D. (1996). Aligning simulation models: A case study and results. *Computational and Mathematical Organization Theory, 1*(2), 123-141.

Burton, R. (1998). Validating and docking: An overview, summary and challenge. In M. Prietula, K. Carley & L. Gasser (Eds.), *Dynamics of organizations* (pp. 215-228). Menlo Park, CA: AAAI.

Byrne, M. D. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*(1), 41-84.

Byrne, M. D., & Gray, W. D. (2003). Returning Human Factors to an engineering discipline: Expanding the science base through a new generation of quantitative methods. Preface to the Special Section. *Human Factors, 45*(1), 1-4.

Byrne, M. D., & Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology, 15*(2), 135-155.

Card, S. K., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

Cohen, M. A., Ritter, F. E., & Haynes, S. R. (in press). Applying software engineering to agent development. *AI Magazine*.

Cohen, M. A., Ritter F. E., & Haynes S. R. (2009), Evaluating design: A formative evaluation of agent development environments used for teaching rule-based programming. In proceedings of the Information Systems Education Conference 2009, 1542-7382, Washington , DC

Cohen, M. A. 2008. A Theory-Based Environment for Creating Reusable Cognitive Models. Ph.D. diss., College of Information Sciences and Technology, The Pennsylvania State Univ., University Park, PA.

Friedrich, M., Cohen, M. A., & Ritter, F. E. (2007). *A gentle introduction to XML within Herbal*. University Park, PA: ACS Lab, The Pennsylvania State University.

Friedrich, M. B. (2008). *Implementierung von schematischen Denkstrategien in einer höheren Programmiersprache: Erweitern und Testen der vorhandenen Resultate durch Erfassen von zusätzlichen Daten und das Erstellen von weiteren Strategien (Implementing diagrammatic reasoning strategies in a high level language: Extending and testing the existing model results by gathering additional data and creating additional strategies)*. Faculty of Information Systems and Applied Computer Science, University of Bamberg, Germany.

Friedrich, M. B., & Ritter, F. E. (2009). Reimplementing a diagrammatic reasoning model in Herbal. In *Proceedings of ICCM - 2009- Ninth International Conference on Cognitive Modeling*. Manchester, England.

Harrison, A. (2002). jACT-R: Beta and beyond. In *The 9th Annual ACT-R Workshop*. Pittsburgh, PA.

Haynes, S. R., Cohen, M. A., & Ritter, F. E. (2009). Designs for explaining intelligent agents. *International Journal of Human-Computer Studies, 67*(1), 99-110.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction, 3*(4), 320-351.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of CHI 2004 (Vienna, Austria, April 2004)*, 455-462. ACM: New York, NY.

Jones, R. M., Crossman, J. A. L., Lebiere, C., & Best, B. J. (2006). An abstract language for cognitive modeling. In *Proceedings of the 7th International Conference on Cognitive Modeling*, 160-165. Erlbaum: Mahwah, NJ.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27-41.

Kieras, D. E., Wood, S. D., Abotel, K., & Hornof, A. (1995). GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'95)*, 91-100. ACM: New York, NY.

Kim, J. (2008). *Procedural skills: From learning to forgetting*. Department of Industrial and Manufacturing Engineering, The Pennsylvania State University, University Park, PA.

Louie, M., Carley, K. M., Haghshenass, L., Kunz, J. C., & Levitt, R. E. (2003). Model comparisons: Docking OrgAhead and SimVision. In *NAACSOS Conference 2003*, Day 3, Electronic Publication, Pittsburgh, PA.

Newell, A., Yost, G. R., Laird, J. E., Rosenbloom, P. S., & Altmann, E. (1991). Formulating the problem space computational model. In R. F. Rashid (Ed.), *Carnegie Mellon Computer Science: A 25-Year commemorative* (pp. 255-293). Reading, MA: ACM-Press (Addison-Wesley).

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process: A new look*. Washington, DC: National Academy Press. books.nap.edu/catalog.php?record_id=11893.

Ray, D. (2009). jSoar: A pure Java implementation of Soar. In *The 29th Soar Workshop*. Ann Arbor, MI.

Ritter, F. E., Haynes, S. R., Cohen, M., Howes, A., John, B., Best, B., et al. (2006). High-level behavior representation languages revisited. In *Proceedings of ICCM - 2006-Seventh International Conference on Cognitive Modeling*, 404-407. Edizioni Goliardiche: Trieste, Italy.

Ritter, F. E., & Young, R. M. (2001). Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design. *International Journal of Human-Computer Studies, 55*(1), 1-14.

Rosenbloom, P. S. (2009). Towards a new cognitive hourglass: Uniform implementation of cognitive architecture via factor graphs. In *Proceedings of ICCM - 2009- Ninth International Conference on Cognitive Modeling*, 114-119. Manchester, England.

Salvucci, D. D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture. In *Human Factors in Computing Systems: CHI 2003 Conference Proceedings*, 265-272. ACM: New York, NY.

St. Amant, R., Freed, A. R., & Ritter, F. E. (2005). Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research, 6*(1), 71-88.

St. Amant, R., Horton, T. E., & Ritter, F. E. (2007). Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Interaction, 14*(1), 24 pages.

Stewart, T. C., & West, R. L. (2005). Python ACT-R: A new implementation and a new syntax. In *12th Annual ACT-R Workshop*.

Deductive Spatial Reasoning: From Neurological Evidence to a Cognitive Model

Marco Ragni, Thomas Fangmeier, Sven Brüssow
{ragni, fangmeier, sven}@cognition.uni-freiburg.de
Center for Cognitive Science, University of Freiburg
Friedrichstr. 50, D-79098 Freiburg, Germany

Abstract

Cognitive modeling aims more and more to explain, predict and integrate behavioral data with brain activations found in fMRI studies. In this article we analyze transitive inferences (e.g. A is left of B and B is left of C then A is left of C) during the spatial reasoning processes. Behavioral findings suggest that reasoners tend to construct a mental model from the premises, which they in turn use to inspect to draw inferences. A reanalysis of our own previous fMRI-study investigating such examples provided us with brain activations pattern. A cognitive model using the (restricted) Bold-function in ACT-R 6.0 can partially predict and explain the results. The findings, limits and potentials of the current representation of the Bold-function in ACT-R are briefly discussed.

Keywords: Deductive reasoning; fMRI; ACT-R

Introduction

Assume you receive the following information:

The door is to the left of the garage.
The car is to the right of the garage.

Given this set of premises it is easy to draw an inference like "the car must be to the right of the door". But how do we reason about such so-called three-term problems? Which role plays working memory in such tasks? There are competing and different theories in cognitive science to explain the actual human reasoning process.

The *Theory of Mental Logic* introduced by (Rips, 1994) argues syntactically. This theory claims that humans apply transitivity rules to a given set of premises without constructing spatial representations, e.g. "If A is left of B and B is left of C then A is left of C". Standing in the tradition of AI-Approaches, there are, however, a number of problems involved, e.g. with regard to memory burden or the number of rules necessary to solve tasks (Ragni, 2008).

In contrast, the *Theory of Mental Models* (MMT) argues that humans construct mental models which are an internal representation of objects and relations in spatial working memory, matching the state of affairs given in the premises. The semantic theory of mental models is based on the mathematical definition of deduction, i.e. a propositional statement C is a consequence of a set of premises P, if in each model A of the premises P, the conclusion C is true. The mental model theory (MMT) assumes that the human reasoning process consists of three distinct phases: (1) the model generation phase, in which a first model is constructed out of the premises, (2) the inspection phase, in which the model is inspected to check if a putative conclusion is consistent with the current model. And (3) the validation phase, in which alternative models are generated from the premises that refute this putative conclusion (Johnson-Laird, 2001). A mental model is constructed incrementally from its premises (Ragni, Fangmeier, Webber, & Knauff, 2007) following the principle of economicity (Manktelow, 1999). Such a model construction process saves working memory capacities because new information is immediately processed and integrated into the model (Johnson-Laird & Byrne, 1991; Rauh, Knauff, Kuß, Schlieder, & Strube, 2005).

Both theories can explain a number of results but MMT is more widely accepted as the explaining theory in relational reasoning (e.g., Rauh et al., 2005; Jahn, Knauff, & Johnson-Laird, 2007; Goodwin & Johnson-Laird, 2005).

A *cognitive modeling* of this theory has several advantages: (i) this theory is more formally presented, (ii) it is fully specified in terms of necessary operations to process such problems as described above, and with the new Bold-functions in ACT-R 6.0 (iii) it allows for a prediction and model of the underlying brain activations. Especially, the last aspect has become more and more important in recent years. Foundational work has been done by Anderson, Qin, Stenger, and Carter who conducted and analyzed simple algebra tasks and developed a first model integrating fMRI-findings in ACT-R (Anderson, Qin, et al., 2004). More precise, based on ACT-R 6.0 they developed an information-processing model to predict the blood oxygenation level-dependent (BOLD) response of functional MRI in symbol manipulation tasks. Base-level activation learning in the ACT-R theory can predict the change of the BOLD response in practice in a left prefrontal region reflecting retrieval of information. In contrast, practice has relatively little effect on the form of BOLD response in the parietal region reflecting imagined transformations to the equation or the motor region reflecting manual programming.

In this article, we present a cognitive model for three-term series problems of spatial arrangements integrating a previous fMRI study. It is structured as follows: In the next section, we briefly introduce the experimental design, settings, and the fMRI-findings. Then, we proceed outlining our ACT-R model. Finally we compare the model results with the empirical results.

fMRI During Visual Relational Reasoning

We briefly report a study from our group (Fangmeier, Knauff, Ruff, & Sloutsky, 2006) in which different neural networks for three phases of the MMT during spatial relational reasoning were supported.

Participants. Twelve right-handed male students took part in the study. All were instructed and trained outside the scanner in order to minimize the learning process while scanning and to increase their accuracy.

Materials. The presented material in the original study consists of two conditions, 32 reasoning and 32 maintenance verification tasks for each subject. Since we just want to model the reasoning process in ACT-R we report only the reasoning task in detail. One reasoning task consists of two premises with three letters (V, X, Z in random order) in a spatial horizontal configuration as well as an offered conclusion. Each premise and the conclusion consists of two letters with a spatial relation. The spatial relation between the two letters of each premise or conclusion was coded by placing it right or left from the center of the screen. A sentential version of the given example in Fig. 1 would be: "X is to the left of V (premise 1) and "Z is to the right of V" (premise 2). For these premises, it follows "X is to the left of Z" (mental model which was constructed). Participants were asked to decide if an offered conclusion was correct. One of two alternative conclusions were offered: a valid one (as in Fig. 1) "X is to the left of Z" or an invalid one "Z is to the left of X".

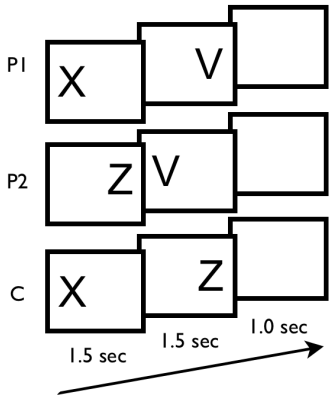


Figure 1: Sequential presentation and timing of the premises and the conclusion (cp. section Materials).

Procedure and Data Acquisition. The participants were trained outside the scanner with 12 similar problems and had to reach at least 75% accuracy for participation. The trials were presented in an event-related design with four separate runs. Each run consist of eight reasoning and eight maintenance tasks in a random order. As noted before we report in this article only the procedure and results of the reasoning tasks.

The timeline of the complete task was as follows: Each task was introduced with the letter "S" in the center of the screen ("Schliessen" in German) for reasoning followed by a pause for 1 sec. Each premise and conclusion began with the presentation of the first letter for 1.5 sec, followed by the second letter for 1.5 sec and a pause for 1 sec. Therefore each of the premises, and the conclusion lasted for about 4 sec. Overall the whole trial lasted for about 14 sec. In half of each premise or conclusion the first letter appeared on the left position, followed by the letter on the right position. In the other half of the tasks the first letter appeared on the right position. The term order variation prevented the participants from anticipating the next letter and from drawing the conclusion during the second premise. Further the variation of the term order is well established in the reasoning literature (Knauff, Rauh, Schlieder, & Strube, 1998). During presenting of the conclusion the accuracy was recorded via a two-button box. Scanning was performed on a 1.5 T Siemens Vision scanner. Functional images were collected with a gradient-recalled echo-planar imaging (EPI) sequence, allowing the sampling of 30 parallel slices covering the whole brain [TR repetition time): 4000 msec; TA (acquisition time): 3126 msec]. The exact scanning information can be seen in Fangmeier and colleagues (2006).

Design. Functional and anatomical images were reoriented so that the anterior commissure corresponded to the origin of the three-dimensional standard coordinate system used in the software SPM99 (1999). The four runs for each subject were separately realigned and corrected for motion, and underwent slice timing correction. Each subject's anatomical image was coregistered with a 40-slice EPI and the functional images of each run. The parameters for spatial normalization were determined from the anatomical images of each subject, and were applied to the corresponding functional images. Images were finally smoothed with an 8-mm full-width half-maximum Gaussian kernel.

fMRI Statistical Analyses. The hemodynamic response to the premises and conclusions was modeled with event-related delta functions, which were convolved with the canonical hemodynamic response function and its temporal derivative employed in SPM99. Low-frequency confounds were excluded from the model with a high-pass filter (192 sec cutoff), and an autoregression AR(1) model excluded the variance explained by the previous scan. The six realignment parameters for each run were included as covariates to avoid motion artifacts. First-level contrast images for every subject and contrast were then used for a random effects analysis to draw inferences on brain activation during the experimental problems. Only correctly answered problems were included in the analysis. All reported clusters within the conditions and the conjunction analysis are significant at the cluster level p .05, corrected for multiple comparisons (threshold t = 3.0).The contrasts were calculated as follows: premise processing phase (Premise 2 minus Premise 1), integration phase (Premise 2 minus Conclusion), validation phase (Conclusion minus Premise 2).

Further the beta values from the essential significant clusters were extracted. For each of the three different phases

(premise 1, premise 2, conclusion) a cluster with ±12 mm around the peak voxel was extracted from the beta images of the SPM statistic. The beta value for each phase represents the difference between brain activation during this phase and the overall mean derived from the whole brain, which is the actual value of the corresponding phase. The value is not a percent signal change but a difference to overall mean with an arbitrary unit. If the beta value is positive (or negative) the activation is higher (or lower, resp.) than the average activation as illustrated in the bar charts of the human data in Fig. 6.

Results. Our findings support the main assumptions of the MMT with respect to distinct phases.

During the initial *premise processing phase*[1] (see Fig. 2 A, B) for both presented premises occipito-temporal structures are activated with the following main Brodmann areas (BA 18, 19, and 37). These areas are active during tasks which are involved in visual working memory and imagery (Kosslyn, Ganis, & Thompson, 2001; Postle, Stern, Rosen, & Corkin, 2000) and with the ventral "what"-stream (Ungerleider, Courtney, & Haxby, 1998).

The following *integration phase* (see Fig. 2 B) shows an additional area in the anterior prefrontal cortex which covers the BA 32 and 10. Tasks in which multiple relations have to be hold simultaneously activated area 10 (Christoff et al., 2001; Prabhakaran, Rypma, & Gabrieli, 2001; Waltz et al., 1999) and a review of functions of the anterior prefrontal cortex assume that this area is responsible for the combination and coordination of multiple cognitive operations (Ramnani & Owen, 2004). Especially support for the premise integration comes from Kroger and colleagues (2002).

In the *validation phase* (Fig. 2 C) a putative conclusion has to be verified. The activation switched from the visual working memory (BAs 18, 19, and 37) to the posterior parietal cortex (BAs 7 and 40). This areas are frequently activated during spatial processing (Burgess, Maguire, Spiers, & O'Keefe, 2001) and the integration of sensory information from all modalities into an egocentric spatial representation (Xing & Andersen, 2000; Andersen, Snyder, Bradley, & Xing, 1997).

Cognitive Model

ACT-R is a cognitive architecture that consists of a number of modules each associated with certain cortical regions (Anderson, Bothell, et al., 2004; Anderson, Qin, et al., 2004; Anderson et al., 2008). When, for example, an ACT-R model that has been built is pressing some key on a keyboard, the manual module will be active and this predicts BOLD activity in the corresponding motor region in the brain. ACT-R's central executive—its procedural backbone—is a production system represented by the procedural module that is associated with the caudate region. Each time a production fires

[1]We denote the phases slightly different.

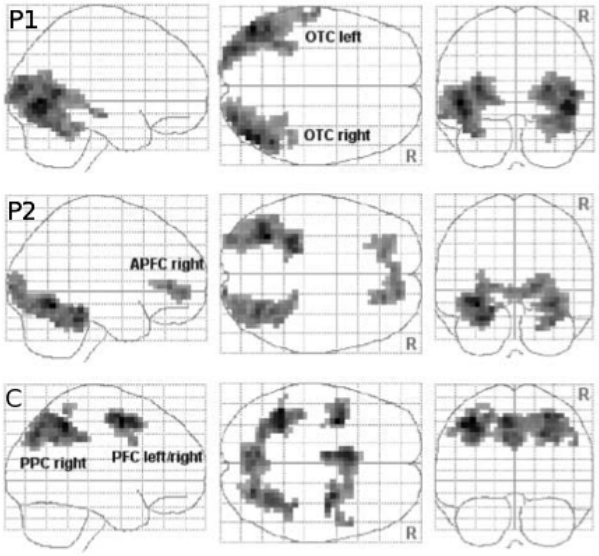


Figure 2: Brain activation during reasoning. Activated regions are contrasts for the three phases calculated with SPM: premise processing (P1), integration (P2), and validation phase (C). The activations were significant at the cluster level calculated with SPM99 ($p \leq .05$, corrected, threshold t = 3.0).

ACT-R predicts the BOLD rate in the caudate region is going up with a certain time lag as is known from real fMRI studies.

The procedural module controls ACT-R's strictly serial behavior; only one production in a time may fire. The modules, however, may operate in parallel and communicate over their buffers, each capable of holding one chunk of information. Hence, a production can require information from more than one module's buffer. Once a module is active, however, it only can become active again in a subsequent request, when it is free again.

The ACT-R model operates on three different kinds of chunks: (1) premise and conclusion chunks, (2) grid chunks, and (3) mental model chunks.

Premise and conclusion chunks are structurally equivalent and the corresponding chunk type defines two slots for the left and right term. Each time a term is presented the ACT-R model tries to integrate the term into the premise chunk or conclusion chunk respectively. After completion of P1 the corresponding chunk is integrated into the center of the second kind of chunk, a grid with four vacant positions (cp. Figure 3, P1). Once the mental model of P1 is complete it is cleared from the imaginal buffer by placing a new chunk of the type grid representing the position of the current model and adjacent free positions around it into the imaginal buffer. The first term of P2 is presented and if it has not been seen yet, the current grid is cleared from the the imaginal buffer. A new premise chunk with only one term is placed in the imaginal buffer instead.

This, however, is the first source of an possible error. Each

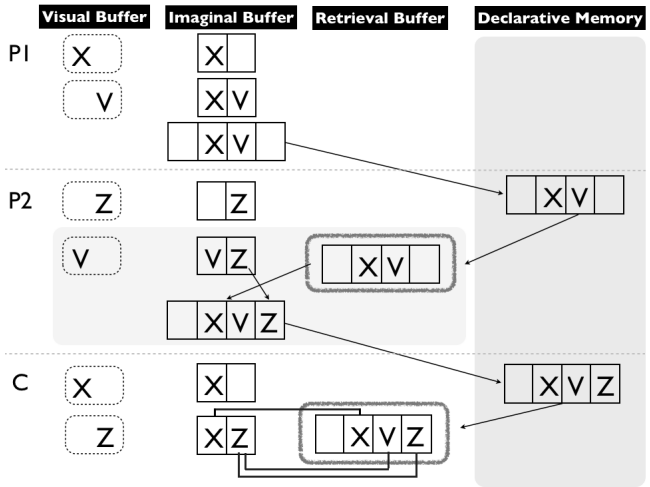


Figure 3: The ACT-R model processing the premises (P1, P2) and conclusion: the columns represent the different buffers each holding the respective chunks. While P1 is presented on the screen, both terms are successively placed into the slots of a two-model chunk that is generated in the imaginal buffer. In a subsequent step the information of this two-model chunk is integrated into a grid-chunk (indicated by 4 cells). After an analog processing of P2 the corresponding two-model is merged with the grid chunk. Then the conclusion chunk is built up. Finally, each occupied cell of the grid chunk is iteratively compared with the conclusion chunk.

time a grid chunk is released to declarative memory and an identical one is detected both get merged to one and its activation is boosted. Hence, the more often two chunks have been merged in the past, the more dominant the result gets and interferes with the grid chunk that has most recently been created. Hence, recency is not necessarily a guarantee for successful retrieval.

The next term is presented and integrated into the premise chunk in the imaginal buffer. The grid chunk is retrieved from declarative memory and is now placed into the retrieval buffer. Now both chunks can be tested on the right hand side of a production and finally the open position in the grid can be filled according to the position in the premise chunk (cp. Figure 3, P2).

In a next step the imaginal buffer holding the current grid chunk, however, has to be cleared again in order to build the conclusion chunk. At the moment the first term of C is presented the model clears the grid chunk from the imaginal buffer in order to make it free for the creation of the conclusion chunk. The creation of the conclusion chunk is analogous to the creation of the premise chunks as described above.

Each cell of the built model of C is iteratively compared with the grid chunk in the retrieval buffer (cp. Figure 3, C). Here, however, a second source of a possible error can occur. In case of the release of a chunk to declarative memory and the retrieval from it in a subsequent step, the same problem occurs as described above. If the time between clearing a chunk from the imaginal buffer and being retrieved again is at a minimum, it can be retrieved again in order to be compared with the conclusion chunk. Otherwise the most general chunk that has repeatedly been merged in the past and that consequently is most dominant in terms of it strengthened activation may get retrieved erroneously. This chunk may cause an error at the comparison stage, because the cues in its slots may not match those from the conclusion model.

Empirical Evaluation and General Discussion

In the sense of Anderson and colleagues the presented model is not an attempt to cover all aspects of deductive reasoning and mental model theory but to add to a methodology that has recently attracted the attention of researchers: the evaluation of cognitive models with fMRI data and vice versa (Anderson et al., 2008, 1325). Nevertheless, the accuracy of the human and the model data fits quite well (human = 93%, model = 94%).

Table 1 shows the brain regions that are supposed to be linked to the buffers of ACT-R modules and Figure 6 illustrates the predicted BOLD responses of the model.

Table 1: Brain regions and corresponding Brodmann areas associated with ACT-R modules (Anderson et al., 2008, 1327).

Region	Brodmann	Module
Motor1	2, 4	Manual
ACC	24, 32	Goal
PPC	7, 39, 40	Imaginal
LIPFC	45, 46	Declarative
Caudate		Procedural
Fusiform	37	Visual

BOLD responses have been computed of a model run that simulates 32 trials of deductive reasoning tasks. Figure 4 shows the overall mean values for each phase P1, P2, and C. Figure 5 shows the continuous course of the overall mean BOLD response predictions for selected ACT-R modules. In all three phases there is almost no change in the rates for the manual module. The reason is that there is a lag of up to four seconds until the corresponding BOLD activity reaches its maximum. This, however, happens in the 12 seconds time window after an answer key has been pressed and consequently cannot be seen in the presented time frame. When the next trial starts the BOLD activity has decayed to its normal rate. This is analogous to the human data and therefore there is also no prominent BOLD activity for the three phases.

The declarative module, too, shows only low activity, slightly increasing towards the end. The reason is that model heavily relies on involving the imaginal module. Only when there are two chunks that have to be tested in a production concurrently there is the need to temporarily clear one chunk from the imaginal buffer because each buffer can only hold

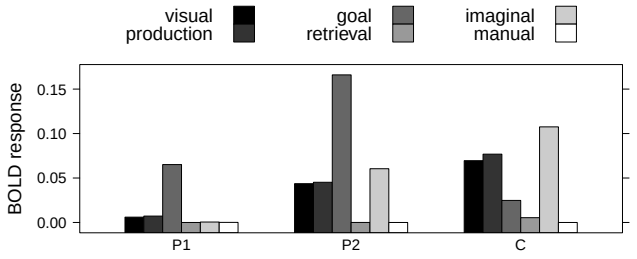


Figure 4: The overall mean BOLD response predictions for six ACT-R modules for the first premise (P1), the second premise (P2), and the conclusion (C).

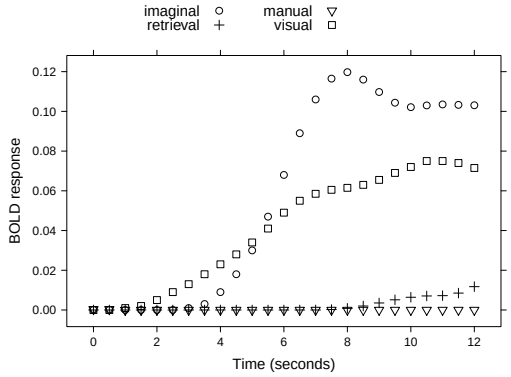
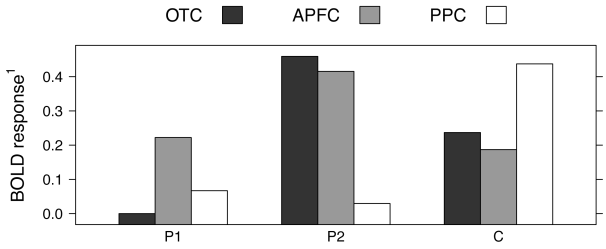


Figure 5: The course of the overall mean BOLD response predictions for selected ACT-R modules.

one chunk at a time. The model chunk gets retrieved immediately from declarative memory again via the retrieval buffer. There is, however, in most cases only one retrieval towards the end of P2 so that the predicted BOLD response is not comparable to that of the imaginal buffer. Only when the first term of the second premise presented on the screen has not been seen before a second retrieval is required. In addition, the maximum rate will, similar to that of the motor module, be in the lag of 12 seconds between two trials.

In the following, we concentrate the empirical evaluation of the correspondence between model and fMRI data on those three brain regions that have both been investigated in the study of Fangmeier et al. (2006) and that are also linked to the buffers of ACT-R modules. Figure 6 directly compares human data with model data. The scales for the human data, however, should be compared with caution, because typically in fMRI research the Δ-adjusted BOLD function with respect to mean activation is reported. For the present work this implied a transformation of the Δ-adjusted BOLD to absolute values in order to get comparable charts with the ACT-R BOLD response predictions. All predicted values of ACT-R were within an interval of [0.0-1.0], whereas in fMRI the beta means are not restricted to a fixed interval (i.e. values can also be negative or beyond 1.0). However, comparing the results at a qualitative level shows a similar pattern as is illustrated in Figure 6. An interesting difference between the predicted



[1] Calculated from the Δ-adjusted BOLD (Fangmeier et al., 2006).

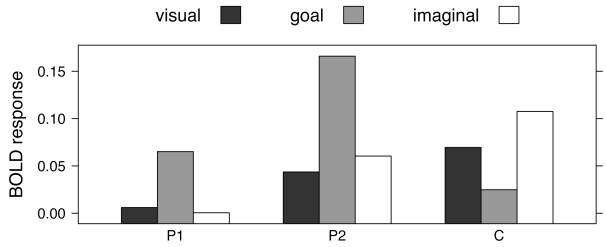


Figure 6: The overall mean BOLD responses for three brain regions (top) and the corresponding predictions of three ACT-R modules (bottom) for the first premise (P1), the second premise (P2), and the conclusion (C): the occipito-temporal cortex (OTC) overlaps with Brodmann area (BA) 37 and is linked to the visual module; the anterior prefrontal cortex (APFC) overlaps with the anterior cingulate cortex, BA 32), that is linked with the goal module; the posterior parietal cortex (PPC) overlaps with BA 7, 40 and is linked to the imaginal module. Each phase (P1, P2, C) lasts 4 seconds resulting in a total presentation duration of 12 seconds (cf. Fig. 1 and 5).

BOLD function and the experimental results is within P2: the difference between the BOLD linked to the visual module and the corresponding brain region of occipito-temporal cortex (OTC). This remains still an open question.

Taken together, ACT-R 6.0 offers a powerful possibility to predict behavior and associated brain activations. This allows to model the different levels from neurological evidence to symbolic modeling. Integrating neurological findings have a main advantage for cognitive modeling: The goodness-to-fit can be extended far beyond the behavioral data, especially for the domain of complex cognition (Anderson et al., 2008, 1324). Differences in the setting can be traced back to different modules (which have different activation patterns). Certainly, a main problem is to compare results of the fMRI studies with predictions of the BOLD-function since additional work is necessary to identify the different scaling and intensity of the activations. So in some sense, the predicted BOLD function gives a good intuition, especially for qualita-

tive comparison. Once a refinement of the modules in ACT-R is taken into account the fields of fMRI and cognitive modeling converge stronger.

Future work will integrate and compare the findings on the memory tasks to the deductive reasoning tasks.

Acknowledgements

This research was supported by the DFG (German National Research Foundation) in the Transregional Collaborative Research Center, SFB/TR 8 within project R8-[CSPACE] and the strategic project ActivationSpace. The authors are grateful to Matthias Frorath for assistance in the implementation of the ACT-R model.

References

Andersen, R. A., Snyder, L. H., Bradley, D. C., & Xing, J. (1997). Multimodal representation of space in the posterior parietal cortex and its use in planning movements. *Annual Review of Neuroscience*, *20*, 303–30.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036-1060.

Anderson, J. R., Carter, C. S., Fincham, J. M., Qin, Y., Ravizza, S. M., & Rosenberg-Lee, M. (2008). Using fMRI to test models of complex cognition. *Cognitive Science*, *32*(8), 1323-1348.

Anderson, J. R., Qin, Y., Stenger, A., & Carter, C. S. (2004). The relationship of three cortical regions to an information-processing model. *Journal of Cognitive Neuroscience*, *16*(4), 637-653.

Burgess, N., Maguire, E. A., Spiers, H. J., & O'Keefe, J. (2001). A temporoparietal and prefrontal network for retrieving the spatial context of lifelike events. *Neuroimage*, *14*(2), 439–53.

Christoff, K., Prabhakaran, V., Dorfman, J., Zhao, Z., Kroger, J. K., Holyoak, K. J., et al. (2001). Rostrolateral prefrontal cortex involvement in relational integration during reasoning. *Neuroimage*, *14*(5), 1136–49.

Fangmeier, T., Knauff, M., Ruff, C. C., & Sloutsky, V. (2006). fMRI evidence for a three-stage model of deductive reasoning. *Journal of Cognitive Neuroscience*, *18*(3), 320-334.

Goodwin, G. P., & Johnson-Laird, P. N. (2005). Reasoning about relations. *Psychological Review*, *112*(2), 468-493.

Jahn, G., Knauff, M., & Johnson-Laird, P. N. (2007). Preferred mental models in reasoning about spatial relations. *Memory & Cognition*, *35*(8), 2075–87.

Johnson-Laird, P. N. (2001). Mental models and deduction. *Trends in Cognitive Sciences*, *5*(10), 434-442.

Johnson-Laird, P. N., & Byrne, R. M. J. (1991). *Deduction*. Hillsdale, NJ: Erlbaum.

Knauff, M., Rauh, R., Schlieder, C., & Strube, G. (1998). Continuity effect and figural bias in spatial relational inference. In *Proceedings of the twentieth annual conference of the cognitive science society* (p. 573-578). Mahwah, NJ: Erlbaum.

Kosslyn, S. M., Ganis, G., & Thompson, W. L. (2001). Neural foundations of imagery. *Nature Reviews Neuroscience*, *2*(9), 635–42.

Kroger, J. K., Sabb, F. W., Fales, C. L., Bookheimer, S. Y., Cohen, M. S., & Holyoak, K. J. (2002). Recruitment of anterior dorsolateral prefrontal cortex in human reasoning: a parametric study of relational complexity. *Cerebral Cortex*, *12*(5), 477–85.

Manktelow, K. I. (1999). *Reasoning and thinking*. Hove, UK: Psychology Press.

Postle, B. R., Stern, C. E., Rosen, B. R., & Corkin, S. (2000). An fMRI investigation of cortical contributions to spatial and nonspatial visual working memory. *Neuroimage*, *11*(5 Pt 1), 409–23.

Prabhakaran, V., Rypma, B., & Gabrieli, J. D. (2001). Neural substrates of mathematical reasoning: a functional magnetic resonance imaging study of neocortical activation during performance of the necessary arithmetic operations test. *Neuropsychology*, *15*(1), 115–27.

Ragni, M. (2008). Human logic in spatial reasoning. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th annual conference of the cognitive science society* (pp. 933–939). Austin, TX: Cognitive Science Society.

Ragni, M., Fangmeier, T., Webber, L., & Knauff, M. (2007). Preferred mental models: How and why they are so important in human reasoning with spatial relations. In C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, & T. Barkowsky (Eds.), *Spatial cognition v: Reasoning, action, interaction* (pp. 175–190). Berlin: Springer.

Ramnani, N., & Owen, A. M. (2004). Anterior prefrontal cortex: insights into function from anatomy and neuroimaging. *Nature Reviews Neuroscience*, *5*(3), 184–94.

Rauh, R., Knauff, C. H. M., Kuß, T., Schlieder, C., & Strube, G. (2005). Preferred and alternative mental models in spatial reasoning. *Spatial Cognition and Computation*, *5*, 239-269.

Rips, L. J. (1994). *The psychology of proof: Deductive reasoning in human thinking*. Cambridge, MA: The MIT Press.

SPM. (1999). London, UK: Wellcome Department of Cognitive Neurology. (computer software)

Ungerleider, L. G., Courtney, S. M., & Haxby, J. V. (1998). A neural system for human visual working memory. *Proceedings of the National Academy of Sciences, U.S.A*, *95*(3), 883–90.

Waltz, J. A., Knowlton, B. J., Holyoak, K. J., Boone, K. B., Mishkin, F. S., de Menezes Santos, M., et al. (1999). A System for Relational Reasoning in Human Prefrontal Cortex. *Psychological Science*, *10*(2), 119–25.

Xing, J., & Andersen, R. A. (2000). Models of the posterior parietal cortex which perform multimodal integration and represent space in several coordinate frames. *Journal of Cognitive Neuroscience*, *12*(4), 601–14.

Accountable Modeling in ACT-UP, a Scalable, Rapid-Prototyping ACT-R Implementation.

David Reitter (reitter@cmu.edu) and **Christian Lebiere (cl@cmu.edu)**
Department of Psychology, Carnegie Mellon University,
5000 Forbes Ave, Pittsburgh, PA 15213 USA

Abstract

ACT-UP is a toolbox implementation of the ACT-R cognitive architecture, aimed at allowing rapid prototyping of complex models. With ACT-UP, we propose *Accountable Modeling*, where the only model components that are specified are those supported by empirical evidence and part of the model's theoretical claims. ACT-UP is a library providing a programmatic interface to the classical ACT-R functionality. Implemented in a functional programming paradigm, models are reusable in other contexts. The toolbox is demonstrated using five implemented and evaluated cognitive models.

Keywords: Complex Models, Cognitive Architectures,ACT-R

Introduction

Cognitive models have explained a great deal of behavioral and neurophysiological data. On the road to understanding the mind, cognitive architectures have specified a core set of representations and mechanisms common to a variety of models in order to separate general functional components and their abilities from domain-specific instantiations, such as knowledge and strategies. However, the tasks that classical cognitive models have taken on are mainly those that can be defined in a controlled environment. Process models of laboratory behavior are often overly specific and needlessly complex, while alternative models would yield similar fits. The model eco-system has diversified rather than converged, with specific rule sets developed for each given task and very seldom reused or generalized for other tasks. This leads to overfitting and lack of robustness. To robustly explain and predict behavior in complex real-life situations, model complexity has to increase further. Inevitably, humans execute much more complex tasks as well, drawing from a variety of knowledge and skills and contextualizing their observations and thoughts in light of both long-term experience and recently acquired knowledge.

The greater complexity of tasks may have a welcome effect on cognitive architectures. Current general architectures such as ACT-R (Anderson, 2007) or SOAR (Laird & Rosenbloom, 1987) are not as restrictive as human memory is. ACT-R, has, during versions 2 through 4, become more and more restrictive: large, very complex rules made way for smaller, granular ones that could describe less functionality each. Still, even its latest incarnation can implement a model that predicts excellent human performance at the most intricate N-back task, failing to explain the dismal human performance scalability at this task. This difficult task (Kirchner, 1958) requires subjects to keep a first-in-first-out queue of N items in memory. Sufficient architectural contraints may mean that modelers can no longer design functional models of existing tasks, let alone the more complex ones we have argued for. One solution to the dilemma is to constrain models by re-use of micro-strategies. It is hoped that the resulting convergence will eventually let us better reflect the architecture of the mind (Newell, 1973).

As task complexity increases, a careful analysis of the components of the model is necessary. Every rule, every data structure, and every knowledge access process can be seen as a claim that needs to be proved empirically. For anything but the simplest cognitive models, many of the procedures and data structures they define are often not evaluated: the specifics of many of the components of the model may be irrelevant to the story a model has to tell. The solution to this problem is *under-specification*. In what we call the *Accountable Modeling* paradigm, we suggest to apply Occam's razor and specify only what is meant to be directly or indirectly evaluated.

As a consequence, we arrive at models that can be more complex yet faster and easier to prototype, while still using the same core representations and mechanisms of the architecture. Until all portions of the model are fully specified, such models may fall short of Newellian *complete process models*. Yet, they honestly separate claim from conjecture and provide the same level of comparison to human data.

Accountable Modeling

Recent work has been undertaken to investigate the use of ACT-R to study the interaction of two, eight, or even thousands of cognitive agents. Scalability in this domain would make cognitive models applicable to new domains such as network science, for which a precise computational representation of human cognitive processes has been desirable but as to now unavailable. The modeling methodology in this paper follows *accountable modeling* within the ACT-R theory.The "Adaptive Control of Thought–Rational" framework (ACT-R, Anderson, 2007) defines a component-based architecture, in which specialized modules work largely in parallel to contribute to thought processes. In recent computational implementations, it requires *end-to-end* models, describing thought processes through a set of production rules controlling the interaction of cognitive (e.g., long-term memory) and perceptual components. We distinguish the ACT-R theory from its canonical implementation (ACT-R 6, Bothell, 2005). In the following, we assume familiarity with the basics of ACT-R.

Working within the ACT-R theory, we designed a new toolbox instantiation of the theory called *ACT-UP*. ACT-UP reflects ACT-R, but lets the modeler specify algorithms much like a programmer would. Functionality is compartmentalized in reusable functions (taking arguments and returning a value) and data is stored and retrieved as in ACT-R in chunks in declarative memory. ACT-UP is intended as a library for

modelers comfortable with basic programming paradigms.

Most of the cognitive functions that ACT-R makes available correspond to the *buffer*-based interface of the architectural modules in ACT-R: a *learn-chunk* function to commit a chunk to memory or boost its activation; a *retrieve* function to request a chunk from declarative memory, taking hard constraints, cues (to spread activation), and soft constraints (for partial matching). (In ACT-R, buffers represent interfaces between cognitive modules. Chunks are bundles of feature-value pairs, which can be stored temporarily in buffers, or, more long-term, in declarative memory.) But ACT-UP also makes more fine-grained cognitive functions available. Such micro-functions allow models to go beyond what is available to ACT-R models.

We intend to address several goals with ACT-UP. *Accountability* suggests to underspecify model components that are neither motivated by data or theory nor subject to empirical evaluation. *Rapid prototyping* allows modelers to quickly build and modify most parts of the model, even computationally complex ones, while focusing on learning and other cognitive effects predicted by ACT-R's theoretical assumptions. Crucially, it produces models that are reconfigurable so that systematic parameter search can be used to explore the space of possible models. *Reusability* results from clear input and output data structures, turning models into functions that can be re-used in other contexts: the *convergence* of models and cognitive frameworks is a long-term goal. *Scalability* allows models to run longer, apply to more complex tasks, and simulate agents in the context of larger multi-agent systems.

To describe the notion of accountability, let us consider some design decisions that a modeler has to make: where to abstract away from subtasks and surrounding tasks, and where to concentrate on the cognitive properties that ultimately explain variance in the data. Both ACT-R 6 as well as the ACT-UP toolbox allow for free computation outside the theory[1]. Even the more theoretically motivated buffers and chunks are storage means that are not limited in size. ACT-UP retains information in local variables, thereby acknowledging the lack of constraints.

Procedures are at the core of ACT-R. They initiate perceptual acquisition, declarative memory retrievals and motor actions and act as an information broker between all components of the architecture. They are implemented as *production rules*, defining a precondition that refers to the state of buffer contents, and a consequential action affecting the buffers and their associated modules. While all rules are eligible to match at all times in a model, only one of them is selected to fire.

Such a production rule system is capable of implementing complex algorithms, especially with the addition of state information in buffers. Thus, the question of whether a solution to the experimental task can be formulated as a set of production rules is less relevant than the question of whether the model's crucial decision-making can be cast as a pattern-matching task, or whether reinforcement learning of recognition patterns and associated actions (as in ACT-R's learning of production rule utility) can explain the observed data. Indeed, in typical models does the deciding learning effect occur only in very specific decision-making moments. The large majority of the model's production rules are in place to deterministically execute the task. These collections of production rules are difficult to develop, inspect, change, maintain and re-use. Therefore, ACT-UP's rules may be underspecified and implemented as a program. This will also often be the case whenever productions implement deterministic and static processes. Other productions may still be faithfully described: those that reflect the crucial pattern-matching tasks and reactions to recognized patterns, or the routinization of initially declaratively memorized processes. This is where the toolbox approach allows modelers to underspecify the model by reformulating productions in a more direct, computationally treatable manner. Underspecification may also occur for many methodological reasons. Data may be lacking to support an evaluation of the claims, if they were specified, or the lack of suitable data, or the task complexity, e.g., understanding of complex natural language instructions where it does not reflect the goals of the modeling work.

ACT-UP provides high-level interfaces to core simulation components of human cognition (e.g., *retrieval of a declarative chunk from a pattern specification*). It also gives modelers fine-grained control over such processes, by filtering chunks from declarative memory or choosing the most active chunks from a set. Thus, the functional toolbox approach integrates well with cognitive mechanisms that do not yet have a well-specified interface to the remaining buffer- and productions-based ACT-R architecture.

Wherever constraints are relaxed, cognitive plausibility comes into question. Traditionally, models have relied on their within-theory specification to provide constraints promoting cognitive plausibility (usually using ACT-R 6). As argued in the introduction, such constraints are not exhaustive. In order to constrain the computational resources available to the model, ACT-UP asks the modeler to focus on the crucial portion of their model, while using the computational power of a programming language for other parts. For those parts, parameters may be fitted that describe their (human) execution time and reliability. Since production systems are Turing-equivalent, we know that a production system can be defined to accomplish what an ACT-UP model does. Thus, ACT-UP models do not represent an implausible gain in power.

ACT-UP architecture

ACT-UP aims to implement a substantial subset of the ACT-R theory. The striking differences between ACT-UP and implementations such as ACT-R 6 or Stewart & West's (2007) Python variant do not lie in the theory: they pertain to the interface that is offered to the modeler. ACT-UP's interface is synchronous and does not yet implement parallelism (which is often not needed). ACT-UP is a toolbox providing ACT-R functionality in a piecemeal fashion as well as commands at a higher abstraction level, which would integrate well with Salvucci & Lee's (2003) motor, speech and perceptual mod-

[1] "eval" statements in ACT-R 6, for instance, allow the modeler to design model components in Lisp.

ule commands. The ACT-UP library is a stand-alone system, and independent of ACT-R 6. It provides a set of Lisp functions and macros; modelers interact with it on the basis of source code that follows Common Lisp syntax (see below for examples). ACT-UP models predict the two major behavioral outcome types: choice and timing.

Declarative memory system

ACT-UP's declarative memory (DM) embodies all the core elements of DM in ACT-R. Memory is accessed in the form of *chunks*, which are sets of feature-value pairs. Chunks are learned (or reinforced) with an explicit command; there is no automatic learning (*buffer clearing* in ACTR 6). Retrieval occurs with a (normally) synchronous command, in which the model specifies hard constraints (a set of feature value pairs), soft constraints (subject to partial matching), and a set of chunks as cues that spread activation. Thus, modelers gain better control over the context of the retrieval. In ACT-R, buffer contents that can spread erroneous activation have to be tightly controlled (or parameterized) in order to prevent unwanted misretrievals. In ACT-UP, assumptions about context elements for each retrieval are explicitly specified. Thus, ACT-UP currently forgoes some ACT-R constraints:

- *strict harvesting* (automatic buffer clearing and learning as chunks): chunks are learned explicitly
- *all-encompassing spreading activation* (all buffers may spread activation): cues are specified during retrieval in ACT-UP
- *unselective partial matching* (the full retrieval request is matched partially): ACT-UP retrieval distinguishes hard and soft constraints

To see a typical chunk creation, retrieval and learning cycle, suppose the model knows initially, via declarative memory, a fact such as *the lawyer is in the dungeon*:

```
(learn-chunk                                  (add to DM)
  (make-fact :name 'l-d-fact                  (new chunk)
             :person 'lawyer
             :location 'dungeon))
```

We can, at model run-time, retrieve and reinforce this chunk:

```
(let ((fact (retrieve-chunk                   (retrieve)
            '(:location dungeon))))           (constraints)
  (if fact (learn-chunk fact)))               (reinforce)
```

Key memory processes such as base-level learning and decay, cue-based memory retrieval, partial matching and their parametrization are equivalent to ACT-R 6. Also available are associative learning as in ACT-R 5 (Anderson, 1993) and Blending (Wallach & Lebiere, 2003). ACT-UP models may define a chunk type hierarchy, and they may derive data structures from chunk types in an object-oriented fashion.

Procedural skills

ACT-R defines procedural rules as fine-grained instructions of the form *If the buffers contain certain values, then change their values according to another template*. ACT-UP is situated at a higher level of abstraction. Modelers may specify complex *rules* that define sequences of actions and preconditions, similar to a Lisp function. Production rules are not usually evaluated in parallel, unless the modeler relies on *utility learning* to model effects through reinforcement learning. ACT-R's utility learning boosts the likelihood of successful productions being chosen in cases of ambiguity (multiple productions match). ACT-UP allows models to define rules and explicitly group them in *competition sets*. ACT-UP can chose a rule from a competition set. Rewards are explicitly back-propagated as in ACT-R 6 in order to let a model learn which rules lead to desirable outcomes. Thus, the production rule conflict sets used in ACT-R are made explicit in ACT-UP rather than being represented implicitly through overlap in production conditions. Routinization effects, where retrievals from declarative memory are side-stepped through specialized, acquired rules, can also be modeled in ACT-UP (the analogous ACT-R mechanism is *production compilation*). ACT-UP's rules consume simulation time (50ms by default), even though the precise predictions that fall out of an ACT-R model are lost, where the same cycle time is assumed, but where production rules are tightly constrained. In line with Accountable Modeling, we propose to fit the execution duration (within plausible bounds) to the data as free parameters.

(1) Validity: The Siegler Model

An ACT-UP model implementing the core of an ACT-R model should result in exactly the same performance results. To test such consistency of ACT-UP and ACT-R 6, we translated several ACT-R 6 models to ACT-UP. Here, we show the *Siegler* model from the ACT-R 6 tutorial. The model explains data by Siegler & Shrager (1984), who found patterns in arithmetic problem-solving in 4-year-olds. In making mistakes when answering addition problems, the children often closely under- or overshot the correct result ($2+3=6$), and their erroneous answers were more frequent and strayed further from the target for problems involving larger numbers. The ACT-R 6 model (following Siegler and Shrager's model) explains these data using a combination of partial matching and base-level activations in memory retrieval of arithmetic facts. Similarity between numbers is proportional to their absolute difference, so that close answers may be retrieved ($2+3=5$). Base-level activations for more frequent addition facts with lower results are higher, leading to more erroneous retrievals and more often for facts involving larger numbers.

The ACT-R 6 model implements a number of deterministic steps: aural presentation, encoding of the numbers, and decoding of the result. The ACT-UP model underspecifies these, as they do not contribute to the variance in the data. The key processing step of the model, the retrieval of arithmetic facts from memory, is accomplished by the following high-level function:

```
(defrule test-fact (arg1 arg2)
  (let ((fact
         (retrieve-chunk                      (retrieve)
           '(:chunk-type plus-fact)           (hard constraints)
           nil                                (no retrieval cues)
           (list :addend1 arg1                (soft
                 :addend2 arg2))))             constraints)
    (if fact (plus-fact-sum fact))))          (extract sum)
```

Model initialization sets base-levels and similarities (in 24 lines of Lisp code), using function calls largely compatible

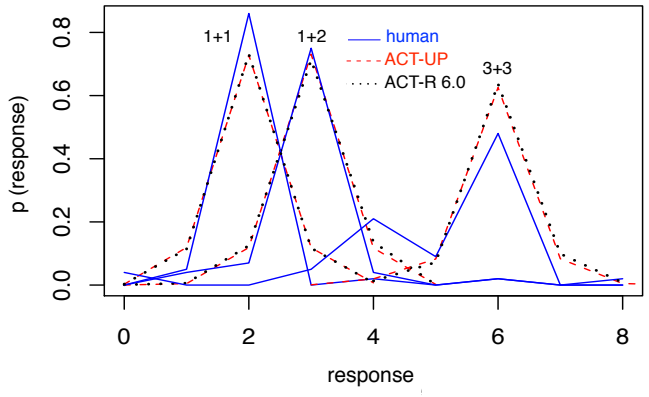


Figure 1: The Siegler and Shrager (1984) data, showing the three distributions of subject's answers to the arithmetic problems 1+1, 1+2 and 3+3, and the simulation results of the model implemented in ACT-R 6 and ACT-UP.

with ACT-R 6. Four architectural parameters are set equally in both variants (retrieval threshold, transient noise, base-level learning (off), and mismatch penalty coefficient).

Both model variants achieved the same correlation (0.966 in ACT-R 6 vs. 0.968 in ACT-UP) and mean deviation (0.053 vs. 0.052) with the data (1000 runs). Figure 1 shows the distribution of the subjects' answers to three of the six problems and demonstrates that the predictions of ACT-UP match closely those of ACT-R 6.

(2) Scalability: A Model of Language Evolution

A multi-agent model was implemented to reflect the emergence of a domain language common to a group of agents after repeated, goal-oriented interactions (Reitter & Lebiere, 2009). In the *Pictionary* games of the empirical study providing data for this modeling exercise, each participant had to convey given meanings via drawings (without words) to another participant. In this model, a language was defined as a set of concept-representation pairs, where a concept was one of 20 target concepts (e.g., *hospital*), and the representation consisted of three drawings of concrete objects (e.g., *building*, *ambulance*, *syringe*). For novel concepts, the model drew from an ontology (graph with weighted associations) linking concepts to related drawings; relatedness was inferred from co-occurrence information in a large text corpus. Known concept-representation pairs were stored in declarative memory. Prototyping the model in ACT-R 6 proved difficult for several reasons. The model was complex, and possible execution paths through the approximately 40 production rules were not evident. Further, the model needed many iterations to show convergence. Parallelization between eight agents and repeated model execution (without reset) was difficult to achieve for technical reasons. We estimate the expended time to be around two person-months. The prototype's results never approached an acceptable fit with the data on a qualitative or quantitate level.

A functional prototype of the model using an initial version of ACT-UP was developed in less than two weeks with the benefit of a task well understood. We focused on plausibility within the ACT-R theory: No data structures were held in memory beyond what could be stored in a buffer; the domain language used declarative memory as intended. Production rules were abstracted using loops, conditionals and ACT-UP commands, owing to the fact that skill acquisition was not part of the model. The model was split up into several functions which could be individually inspected and tested (e.g., "draw", "recognize"). ACT-UP functions were used to inspect the activation of target chunks at retrieval times (base-level, spreading activation) and export those to be visualized along a time-line. With this model, we were able to establish good qualitative empirical correspondence with data from experiments that compared a small community of eight participants interacting in changing pairs, to a set of participants interacting in four one-on-one dyads.

Recently, the model scaled well to multi-agent simulations with 1000 agents and 84 million game interactions (two stateful agents, one concept per game) in about 36 CPU hours. Further work is planned to evaluate scalability to memory-intensive long-term tasks.

(3) Efficiency: A Sentence Production Model

The third case study involves a model that was implemented in both ACT-R 6 and in ACT-UP. It involves a model of sentence production (Reitter, 2008), focusing on the syntactic process, and explaining syntactic priming data that show that subjects are more likely to choose one syntactic variant over another if that variant was presented as a prime ("The girl gave the dog a bone" vs. "The girl gave a bone to the dog."). The model begins with a simple semantic representation (Verb: <give>, Agent: <girl>, Theme: <bone>, Goal: <dog>). Beginning with the verb, it chooses words and their syntactic forms describing how those words can combine (the verb has two forms, yielding the two variants above). Base-level activation of the appropriate syntactic chunks held in DM and spreading activation from the meaning as described above determine which form of sentence is produced. Base-level learning and associative learning (in ACT-UP only) lead to a range of priming effects.

The ACT-R 6 model consists of 30 productions, 7 chunk types, and a variable number of chunks that are created for each word and for several syntactic forms. The ACT-R 6 production rules resulted in 720 lines of code. Base-level activations and associations between words and syntactic forms are initialized programmatically from a corpus of spoken, transcribed and syntactically parsed English. The model was evaluated according to its qualitative and quantitative predictions of syntactic priming effects using a small number of sample sentences. As in the empirical data, syntactic priming depends on the frequency of syntactic constructions and the distance between target sentence and prime.

Studies also show that syntactic priming is much increased when lexical material in the sentence is repeated between prime and target. The model postulated that this was due to learning of associations between lexical or semantic and syntactic chunks–a suggestion that was tested empirically in terms of its theoretical predictions, but associative learning

was not available to the ACT-R 6 model. Consequently, the sentence production model and various initialization and simulation functions were formulated in ACT-UP over the course of about 3 eight-hour workdays. The core function encoding procedural knowledge (sequences of retrievals, conditionals, a loop) has 82 lines of code.[2] The resulting model explained the data including the lexical repetition effect.

In head-to-head comparison, the ACT-R 6 model runs at a speed of 14 sentences per second. The ACT-UP variant produces 380 sentences per second, despite being more specific than necessary to explain the data.[3] This purely technical speed-up translates to a substantial advantage for the modeler: not only is the debugging and experimentation cycle considerably faster, but larger models of more realistic tasks can be run in larger multi-agent simulations, thereby significantly extending the applicability of cognitive models.

(4): Extensibility and Rapid Prototyping: the Dynamic Stocks&Flows Model

The fourth model is another case of rapid prototyping. It illustrates how we could quickly implement a well-documented approach to graded decision-making. *Instance-based learning* (IBL, Gonzalez et al., 2003) stores episodes encoding past decisions and their observed performance in declarative memory. Retrieval then blends those episodes together, weighing their recency and frequency in line with ACT-R's *base-level learning* and *partial matching*.

In an entry (Reitter, 2010) to the *Dynamic Stocks&Flows* modeling challenge[4], IBL was used in two ways. The task in this challenge had subjects extrapolate the change in a given quantity from previous observations. Change rates could be steady (the quantity following a linear function) or harder to predict, including non-linear changes or discontinuous and noisy sequences. IBL modeled the participant's estimates of the change rate and the future value of the quantity. Careful analysis of empirical data showed an interesting pattern: variability often suddenly decreased after about 20 iterations of the task; depending on subject and change function, variability could be grouped into very low and higher pools: subjects were highly precise, or not precise at all. This led to the second use of IBL: a metacognitive layer, which allowed the model to monitor its performance at the task and choose from one of several strategies. Some of these strategies led to precise estimates of the quantity through mental arithmetic, and other strategies used IBL, as described above, to make an educated guess.

The model used declarative memory for its core transaction: declarative chunks store the quantity estimates and the performance monitoring episodes. Blending of stored episodes is implemented (and available) at the ACT-UP level. One free parameter in the model specified the duration of calculations and the wait time between iterations (an underspecified model component); we fitted the parameter from available subject data. The results were plausible given the experimental design. Other parameters were held at their ACT-R defaults; blending parameters were optimized. The model won the challenge by best predicting transfer performance to a set of unknown conditions, indicating that accountable modeling has the potential of increasing generalization of models by focusing on the key processes underlying performance. The same model was later run in an extensive parameter exploration exercise, in which selected architectural and model parameters were systematically varied, with millions of model runs on a computing cluster (Gluck et al., 2010). The exploration included a manipulation that switched individual strategies on and off.

The DSF model exemplifies Accountable Modeling through the decision to not describe the visual and motor interaction with the experiment. While a portion of the data might have been explained by the subject's use of the graphical user interface, neither timing, eye-tracking or mouse movement data were available for validation. Thus, the model underspecifies motor and sensor components.

(5) Reusability: Lemonade Game Agent

The final test case illustrates the re-use of model components. We used a cognitive model in ACT-UP to explore the performance of metacognition in a multi-agent game competition[5]. The DSF challenge model (Reitter et al., 2010) provided the metacognitive layer choosing one of multiple strategies. The model plays a location game (*Lemonade Stand*), where the optimal choice of strategy depends on the strategies played by the two opponents. We designed a metacognitive model that chooses from a wide range of elementary prediction and action strategies based on their track record. The metacognitive model always outperforms all single-strategy models we implemented in a round-robin tournament. The metacognitive layer only had to be minimally adapted: the core functions for learning and blending retrieval were identical; only the task-specific objective functions were redefined. ACT-UP suggests useful compartmentalization: its functions take a set of arguments and return a value; they are intended to be side-effect free apart, of course, from changes to the state of the model. As a consequence, they are reusable in new contexts.

The Lemonade Game agent is not a classical cognitive model, explaining existing empirical data. Instead, its metacognitive layer generates predictions. Not all individual strategies are formulated fully within the theory; thus, we demonstrate a way to combine cognitive and purely algorithmic models. Difficulties arose when the model was readied for submission to a competition, which required Java: in such cases, we got the best use of ACT-UP as a prototyping tool, but had to re-implement the model once validated.

[2] The relatively faithful translation means that we do not fully follow *Accountable Modeling*: the model is overly specific.

[3] Both models keep a similar-size declarative memory, require similar retrievals; ACT-UP adds associative learning. Both models were run in the same LISP environment, without debug output. ACT-R 6 tracing and logging were off, decision tree building on. Optimized learning for ACT-UP and ACT-R 6 at 3 chunks.

[4] `www.hss.cmu.edu/departments/sds/ddmlab/modeldsf/`

[5] `tech.groups.yahoo.com/group/lemonadegame/`

Discussion

Most importantly, we want to propose a modeling paradigm that institutionalizes what is often already the case whenever cognitive models depend on the combination of just a few, specific properties of the architecture. A series of case studies provided the basis for our introduction to ACT-UP. We demonstrated the use of ACT-UP in high-fidelity models with up to 1000 parallel agents; we showed cases of rapid prototyping and of the re-use of model components. Quantitative predictions of ACT-UP parallel those of ACT-R.

The models are intended to integrate within one *architecture*. The emergence of more complex, perhaps unexpected behavior then follows from the reuse and combination of models that describe behavior in much more complex, perhaps even realistic environments. ACT-UP models are intended to be underspecified where data cannot account for the specific claims encoded by the model. Such a modeling paradigm appears not only sensible (as it is evidence-based): it also supports scaling up modeling efforts and extending them to new applications. Architectural flexibility is gained through liberal combination of components, not unlike what was proposed by Cassimatis (2002).

Are such models still models of *cognitive* processes, or are they merely computer programs? First, the execution directives (Lisp clauses) specify the model at a higher level than do production rules: both can be seen as computer programs. Importantly, production rules can implement any algorithm, and could, thus, be derived from the ACT-UP model. Thus, ACT-UP models are not theoretically more powerful. Second, temporary storage of variables and even complex data structures enables the modeler to write implausible ACT-UP models, just like large buffers provide a way to exceed what is cognitively believable. Plausibility is not guaranteed unless modeler discretion is entirely removed, which has not been accomplished under any implementation of the theory. Third, ACT-UP's and ACT-R's longer-term storage model (chiefly declarative memory) is an example of strong constraints on what a modeler can do in these formalisms, as opposed to a non-cognitively motivated program.

Conclusion

We see the current state of ACT-UP as an experimental step to scale up cognitive modeling and extend its areas of applicability. Much work remains to be done. Perceptual and motor components are not yet completed, and parallelism as in ACT-R as well as in its multitasking variant Salvucci et al. (2009) is desirable. The combination of pattern recognition algorithms with ACT-UP may provide for a plausible implementation of the IF part of production rules, possibly to automatically bootstrap and optimize models from sample runs. Larger-scale, long-term simulations will show the limits of the architecture. Still, the wide variety of test cases presented demonstrates scalability w.r.t. modeling effort and computations, and has taken a step towards the integration of high-fidelity cognitive models in complex cognitive systems.

Acknowledgements

The authors acknowledge funding for this work from the Air Force Office of Scientific Research (MURI 7 - FA95500810356).

References

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Erlbaum.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford, UK: Oxford University Press.

Bothell, D. (2005). *ACT-R 6.0 Reference Manual.* Retrieved 12/2009, from `act-r.psy.cmu.edu/actr6/reference-manual.pdf`

Cassimatis, N. L. (2002). *Polyscheme: A cognitive architecture for integrating multiple representation and inference schemes*. Unpublished doctoral dissertation, Massachusetts Institute of Technology.

Gluck, K. A., Stanley, C. T., L. Richard Moore, J., Reitter, D., & Halbrügge, M. (2010). Exploration for understanding in model comparisons. *Journal of Artificial General Intelligence (to appear)*.

Gonzalez, C., Lerch, F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*, 591-635.

Kirchner, W. K. (1958). Age differences in short-term retention of rapidly changing information. *Journal of Experimental Psychology*, *55*(4), 352-358.

Laird, J. E., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*(1), 1-64.

Newell, A. (1973). You can't play 20 questions with nature and win. In W. Chase (Ed.), *Visual information processing.* New York, N.Y.: Academic Press.

Reitter, D. (2008). *Context effects in language production: Models of syntactic priming in dialogue corpora*. Unpublished doctoral dissertation, University of Edinburgh.

Reitter, D. (2010). Metacognition and multiple strategies in a cognitive model of online control. *Journal of Artificial General Intelligence (to appear)*.

Reitter, D., Juvina, I., Stocco, A., & Lebiere, C. (2010). Resistance is futile: Winning lemonade market share through metacognitive reasoning in a three-agent cooperative game. In *Proceedings of the 19th Behavior Representation in Modeling & Simulation (BRIMS).* Charleston, SC.

Reitter, D., & Lebiere, C. (2009). Towards explaining the evolution of domain languages with cognitive simulation. In *Proceedings of the 9th International Conference on Cognitive Modeling (ICCM).* Manchester, UK.

Salvucci, D. D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture. In *Chi '03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 265–272). New York, NY, USA: ACM.

Salvucci, D. D., Taatgen, N. A., & Borst, J. P. (2009). Toward a unified theory of the multitasking continuum: from concurrent performance to task switching, interruption, and resumption. In *Chi '09: Proceedings of the 27th International Conference on Human Factors in computing systems* (pp. 1819–1828). New York, NY, USA: ACM.

Siegler, R. S., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? In C. Sophian (Ed.), *The origins of cognitive skills* (p. 229-293). Hillsdale, NJ: Erlbaum.

Stewart, T. C., & West, R. L. (2007). Deconstructing and reconstructing ACT-R: Exploring the architectural space. *Cognitive Systems Research*, *8*(3), 227-236.

Wallach, D., & Lebiere, C. (2003). Conscious and unconscious knowledge: Mapping to the symbolic and subsymbolic levels of a hybrid architecture. In L. Jimenez (Ed.), *Attention and implicit learning.* Amsterdam, Netherlands: John Benjamins.

Combining Procedural and Declarative Knowledge in a Graphical Architecture

Paul S. Rosenbloom (Rosenbloom@USC.Edu)
Department of Computer Science and Institute for Creative Technologies, 13274 Fiji Way
Marina del Rey, CA 90292 USA

Abstract

A prototypical cognitive architecture defines a memory architecture embodying forms of both procedural and declarative memory, plus their interaction. Reengineering such a dual architecture on a common foundation of graphical models enables a better understanding of both the substantial commonalities between procedural and declarative memory and the subtle differences that endow each with its own special character. It also opens the way towards blended capabilities that go beyond existing architectural memories.

Keywords:

Cognitive architecture; memory; graphical models; procedural; declarative; semantic; episodic; rules; constraints.

The distinction between procedural and declarative knowledge plays a central role in many cognitive architectures. ACT-R has long embodied distinct rule-based procedural and fact-based declarative long-term memories (Anderson, 1993). Early work with Soar instead leveraged a single rule-based long-term memory to support both procedural and declarative knowledge, with rules directly encoding procedures while also providing access paths to facts stored in their actions (Rosenbloom, Newell & Laird, 1991). Yet, Soar 9 has now followed ACT-R's lead, and in fact gone beyond it with distinct declarative memories for semantic and episodic knowledge (Laird, 2008). CLARION embodies the distinction in two different manners (Sun, 2006). It has a procedural Action Control System for controlling action and a declarative Non-Action Control System for general knowledge, but it also has a crosscutting distinction between explicit and implicit knowledge that applies to both of these modules and the whole architecture.

As part of an effort to investigate whether the potential of graphical models (Koller & Friedman, 2009) to unify signal, probability and symbol processing will enable development of simpler yet broader architectures than are seen today (Rosenbloom, 2009a), a new memory architecture with both procedural and declarative memories – but as yet without learning – has been implemented via a common graphical substrate. Guided by the functionality embodied in ACT-R's and Soar 9's long-term memories, the hopes for this implementation were to (1) achieve a straightforward mapping of these disparate memories onto the substrate, resulting in (2) a simpler and more uniform memory architecture, (3) embodying a blended functionality that can (4) exceed existing memory capabilities. The goal was not to model specific results from human memory research, but to understand the implications of graphical implementation and unification on such memory architectures.

Results to date have yielded a new blended memory architecture that is of interest for both the commonality among these memories that it leverages and the subtle differences among them that it exposes. The differences get at some of the most fundamental distinctions between procedural and declarative knowledge while continuing to drive research on their further unification. The next three sections describe the implemented memory architecture along with the commonalities it leverages; the differences this architecture reveals between procedural and declarative memory, as well as, as a bonus, those among different flavors of declarative memory; and what has been yielded so far in terms of blended functionality and new capability. The final section summarizes and looks to the future.

Memory Architecture

ACT-R and Soar 9 each embodies a procedural memory for rules plus a declarative (semantic) memory for facts. Soar 9 also goes a step further, implementing a second distinct declarative (episodic) memory for past history. Although ACT-R does not implement a separate episodic memory, there is work on how its existing mechanisms can yield comparable behavior (Sims & Gray, 2004). The focus here is on uniformly implementing all three of these long-term memory functionalities – one procedural and two declarative – via a common graphical substrate.

The memory architecture is built on top of a *graph layer* based on factor graphs and the summary product algorithm (Kschischang, Frey & Loeliger, 2001). Factor graphs are varieties of graphical models, like Bayesian networks, but enabling efficient computation with arbitrary multivariate functions by decomposing them into products of simpler subfunctions when suitable forms of independence exist; e.g., F(a,b,c) might decompose to $F_1(a,b)F_2(b,c)$. The reduced computation then maps to a bipartite graph in which there are *variable nodes* for variables and *factor nodes* for subfunctions (Figure 1). A variable node is linked to a factor node when the former's variable is used by the latter's function. The summary product algorithm passes messages along these links until quiescence is reached, with each message providing information about the possible values of the variable on the link. Each node computes its output messages by combining its incoming messages, plus its function if it is a factor node. The result is an inherently local computational model that can compute global results

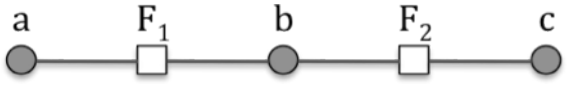


Figure 1: Factor graph for $F(a,b,c)=F_1(a,b)F_2(b,c)$.

across the cycles of message passing leading to quiescence, and that leverages independence for efficiency. It bears a relationship to neural networks, but combines additional breadth in some areas with more constraint in others.

The summary product algorithm is most often used to compute variable *marginals*, integrating information from across the graph to determine which values are legal, and what weights or probabilities are associated with them. When computing marginals, the algorithm typically uses *sum* for summarization, yielding the sum-product variant. When it is preferable to compute the *maximum a posteriori (MAP) estimation* – that is, the single most likely combination of values over all of the variables – *max* is used instead, yielding max-product. The graph layer here defaults to marginals (and sum), but can also compute MAP estimations and employ max when appropriate.

This graph layer is a reimplementation of the one developed in (Rosenbloom, 2009a) for rule match, with improvements in functionality, generality, and efficiency. The biggest change generalizes the representation for factor functions and messages from N dimensional Boolean arrays to N dimensional continuous functions (approximated as piecewise linear functions over rectilinear regions, as in Figure 2). Instead of just supporting symbol processing, this representation has the potential to support: continuous information for perception, imagery, and motor control; discrete distributions for uncertain information; and symbols for general reasoning. Starting from the continuous base, discrete distributions require discretizing variable domains; for example, breaking up the real line into unit segments, one per integer. Symbols then arise when the ranges of discrete variables are restricted to 0/1. A symbol table has also been added to map between unit segments and arbitrary symbols, but it is only for ease of programming and has no effect on the workings of the summary product algorithm.

$y\backslash x$	[0,10>	[10,25>	[25,50>
[0,5>	0	$.2y$	0
[5,15>	$.5x$	1	$.1+.2x+.4y$

Figure 2: Example (2D) piecewise linear function.

To implement the memory architecture, a *memory layer* was built on top of the graph layer that reifies a distinction between long-term and working memory, as in both ACT-R and Soar 9. Long-term memory structures compile into subgraphs that both store and access the knowledge. Working memory compiles into functions in peripheral factor nodes that remain fixed within a single cycle of memory access – i.e., within a single settling of the graph – but can be altered between cycles.

Long-term memory structures are specified at the memory layer as *conditionals*, generalized rules combining *patterns* and a *function*. Each pattern has a predicate plus one or more arguments specifiable as constants or variables; e.g., `Object(`s`,O1)` is a pattern with predicate `Object` plus the variable s (for states) and the constant `O1` (an object) as arguments. A pattern compiles into a linear graph structure that has a working-memory node at one end, a variable node at the other (for legal values of the pattern's variables), and factors that test pattern constants in between. This fragment corresponds to part of an *alpha network* in the Rete match algorithm, with the variable node acting as an *alpha memory* (Forgy, 1982). The big difference though is that in Rete messages always flow from working memory to the alpha memory. Here, messages can flow in either or both directions. As in Rete, the flow is away from working memory for *conditions* (Figure 3), but the flow is towards working memory for *actions*. *Condacts* – a neologism for *cond*itions and *act*ion*s* – are patterns for which the flow is bidirectional. A single conditional can have any combination of conditions, actions and condacts.

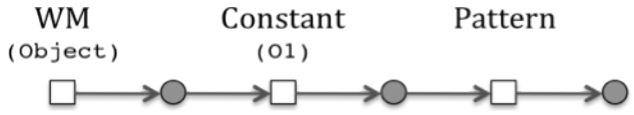


Figure 3: Alpha network for condition `Object(`s`,O1)`.

Patterns are combined into conditionals by a network of factor nodes that test equality of variable binding across patterns, plus variable nodes that represent combinations of variables across patterns. This portion of the factor graph corresponds to Rete's *beta network*, in which partial instantiations are joined to yield full rule matches. However, here the beta network connects conditions, actions, and condacts though bidirectional message flow.

Functions, when included, are defined over condact variables, and lead to new factor nodes that link with these variables. Functions can represent probability distributions over the cross products of the domains of condact variables, as is typical in many graphical models, but they also can represent other numeric and Boolean functions.

The conditional in Figure 4 uses a condition, a condact, and a function to define a prior distribution over the concept associated with object `O1` in the current state. Object `O1` can be a walker, a table, a dog or a person, each with its own prior probability. The variable in square brackets ($\alpha 1$) is a *pattern variable*. When multiple patterns, possibly across multiple conditionals, share a pattern variable, they compile to the same variable node within the graph. This enables chaining and local bidirectional communication among

CONDITIONAL ConditionPrior
Condition: Object(s,O1)
Condact: Concept(O1,c) [$\alpha 1$]

Walker	Table	Dog	Human
.1	.3	.5	.1

Figure 4: Concept prior over object O1.

conditionals within a single cycle of memory access, for, among other things, correct probabilistic reasoning. The factor graph for this conditional can be seen in Figure 5. Messages spread out from all nodes in this graph, following the directionality of the arrows. The primary constraint on this computation stems from local factor node functions, but as messages propagate, these constraints propagate as well.

If a conditional just has conditions and actions, it is a rule, and can form the basis for a traditional procedural long-term memory. Figure 6 shows a conditional defining a simple rule that performs a transitive computation. As with the earlier graph layer, match time per rule here has a worst-case bound that is exponential in the treewidth of the rule rather than the number of conditions.

```
CONDITIONAL Transitive
   Condition: Next(a,b)
              Next(b,c)
   Action: Next(a,c)
```

Figure 6: Transitive rule.

If a conditional only has condacts, we have symmetric flow among all of its patterns, and the basis for a declarative memory. Figure 7 shows a conditional (including part of the function) for a distribution over the weight of object `O1` given its concept. The `Concept` condact compiles to the same graph node created for it in conditional `ConditionPrior` (Figure 4). The function partitions the weight (in pounds) into a finite number of classes and assigns a linear function to each rectangular region defined by the cross product of the weight class and the concept.

A rule-based procedural memory consists of condition-action conditionals, such as the one in Figure 6. Given just this rule, the graph contains 7 factor nodes and 9 variable nodes. Match requires 47 messages to complete irrespective of the number of matching elements, since each message includes information about all matches. However, more matches may mean more calculation per message, yielding 5 ms elapsed time for one match and 16 ms for two.

A semantic memory implemented along the lines of Anderson's (1990) analysis of categorization and feature prediction includes conditionals for prior probabilities of concepts – such as the one in Figure 4 (although possibly without the condition) – and conditional probabilities of object attributes given concepts, as in Figure 7. By linking these conditionals through the concept's pattern variable, an object's cued features can yield a posterior distribution over its concept – based on conditional probabilities of cued features plus the prior probability of the concept – and this posterior concept distribution can then combine with the conditional probabilities of uncued features to generate probabilistic predictions of their values, all within a single memory cycle. In the particular example used, in addition to the continuous weight feature, there is one discrete numeric feature (legs) plus three symbolic features (color, alive, mobile). The graph comprises 47 factor nodes and 47 variable nodes. Given the cue that the color is silver, quiescence is reached after 634 messages, requiring 100 ms. It predicts that the concept is *walker* because almost all walkers are silver while only a small fraction of dogs and tables are. It also predicts that the cued object is mobile, not alive, has four legs and weighs 10 pounds.

```
CONDITIONAL ConceptWeight
   Condact: Concept(O1,c)[α1]
            Weight(O1,w)[α2]
```

w\c	Walker	Table	…
[1,10>	.01w	.001w	…
[10,20>	.2-.01w	"	…
[20,50>	0	.025-.00025w	…
[50,100>	"	"	…

Figure 7: Conditional probability of weight given concept.

In Soar 9, episodic memory retrieves the most recent episode that best matches the cue, effectively acting as a temporal instance-based semantic memory. This can be implemented much like semantic memory, but with alterations for recency and for retrieving the single best episode given a cue rather than predicting the most likely features given the cue. For recency, a discrete temporal variable replaces the concept variable, with a prior distribution that tails off exponentially into the past (Figure 8). To retrieve the single best episode, each feature conditional specifies the conditional probability of its values over the past history, and shares the `Time` condact with the temporal prior (Figure 9). The implemented example uses the same features as the semantic memory, but stores an object instance at each time step. The graph has 46 factor nodes and 46 variable nodes. Given the cue that the concept is human, it takes 433 messages, over 35 ms, to select the more recent of the two humans seen (at time step 3).

The straightforward implementation of these three varieties of long-term memory via the memory layer goes a long way towards realizing the first hope stated up front. In

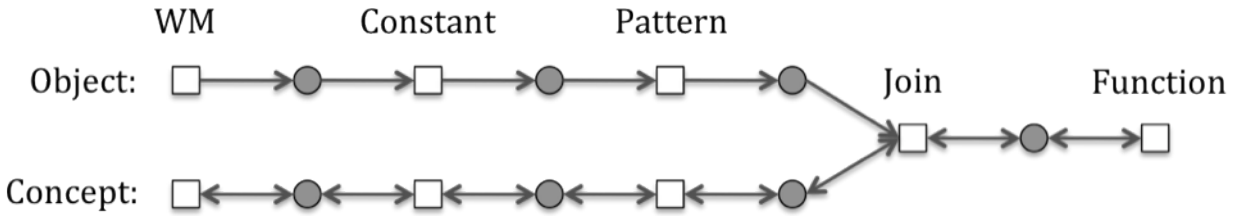


Figure 5: Factor graph for conditional in Figure 4, with a condition (`Object`), a condact (`Concept`), and a function.

CONDITIONAL TimePrior
Condact: Time(*t*) [$\alpha 3$]

0	1	2	3	4
0	.032	.087	.237	.644

Figure 8: Exponentially decaying, discrete, temporal prior.

CONDITIONAL TimeConcept
Condact: Time(*t*) [$\alpha 3$]
Concept(O1,*c*)

t\c	Walker	Table	Dog	Human
1	1	0	0	0
2	0	0	0	1
3	0	0	0	1
4	0	0	1	0

Figure 9: Conditional probability of concept given time.

comparison to the earlier implementation of just a rule-based procedural memory, there is additional complexity here in extending rules to conditionals, and in moving from symbolic to continuous values, but then very little more is needed to implement these particular variants of procedural, semantic and episodic memories. With respect to the second hope's appeal to simplicity and uniformity, there is indeed much in common across the implementations of these three memories: they all build on the distinction between working memory and long-term memory; long-term memory is uniformly represented as conditionals that compile into factor graphs, while working memory is encoded as evidence in peripheral factor nodes; and memory access is cued by working memory through the application of the summary product algorithm to the resulting graph.

One difference of note between this implementation and Soar 9 arises from Soar's ability to perform multiple cycles of procedural (rule) access within a single decision cycle, but only one cycle of declarative (semantic or episodic) access. The memory architecture here is instead limited to just one cycle of memory access per decision cycle for both declarative and procedural knowledge. In (Rosenbloom, 2009b), early experiments with graphical models led to the hypothesis that global computation in Soar should only happen over a full decision cycle rather than once per rule cycle, and that Soar was thus inconsistent in allowing global access to working memory each rule cycle. The current implementation abides by this constraint; however, using pattern variables still allows chaining of rules within a decision cycle, but now based on local communication between actions of earlier rules and conditions of later ones.

Differences

The most obvious difference between the implementations of procedural and declarative memory is the use of conditions and actions in procedural memory versus condacts in declarative memory. At the graph level this reduces simply to the directionality of information flow in the alpha networks, but it does yield a qualitative difference at the memory level. With unidirectional information flow, rules predefine what are to be the cues for retrieval (conditions) and what is to be retrieved (actions). This is particularly effective for procedures as it enables directional if-then programming. In contrast, with bidirectional information flow, both varieties of declarative memory dynamically determine at access time what aspects of an object are cues and therefore what aspects are to be retrieved (i.e., those aspects not cued). This significantly enhances the flexibility of access, but eliminates the directionality that is exploited in procedural programming.

A more subtle difference is whether a *closed world* or *open world* assumption occurs with respect to working memory. Rule-based systems use the former, assuming that anything not in working memory is false. The use of negated conditions depends on this assumption, as does the ability to keep working memory small and focused. On the other hand, declarative memories – and most logical and probabilistic models – use an open world assumption, that the truth of anything not explicitly in evidence is unknown. This enables values that are unknown prior to memory access to be retrieved/predicted by condacts during such access. With a closed-world assumption, this becomes impossible because any values not explicitly true prior to access would be set to false, leading to a conflict with any attempt to make a positive predication during access. Rules avoid this problem because their retrievals/predictions occur non-monotonically at the end of the access cycle, by actions that don't examine working memory during the cycle.

This difference is realized in the graph layer by declaring individual predicates to be closed or open world when they are defined; an idea adopted, along with the use of predicates, from earlier experiments with Markov logic (Domingos & Lowd, 2009) as a general *implementation level* for architectures (Rosenbloom, 2009b). Closed-world predicates are primarily used in conditions and actions and open-world predicates in condacts.

A third difference concerns whether memory access retrieves all cued results or only the best result. In Soar 9's rule-based procedural memory, all combinations of bindings of condition variables to working memory constants yield rule instantiations that fire in parallel. In contrast, cuing of either semantic or episodic memory should return only the best result. At the graph layer, this difference is interpreted in terms of distinct types of variable domains. When only the best result is desired, the variable's domain is declared *unique*, and messages about it are normalized to sum to 1. This yields a distribution over the variable's domain elements for the probabilities that they are to be retrieved. When all results are to be returned, the variable domain is declared to be *multiple*, and its messages are not normalized. In such cases, each domain element acts roughly as its own Boolean variable, with a value of 1 if it is to be retrieved and 0 otherwise; thus encoding all bindings of the variable

in each message. The summary product implementation then uses max to summarize over *multiple* variables, even when marginalizing, bounding the result above by 1.

These three differences – (1) the directionality of information flow in alpha networks, (2) a closed-world versus open-world assumption, and (3) unique versus multiple variables – jointly distinguish procedural from declarative memories in this implementation. Of these, the first appears to be the most fundamental, to the point where it justifies an explicit hypothesis that such a difference will always be found in comparing procedural and declarative memories. The other two are less clear. It may be possible, for example, to build an effective procedural memory based on an open-world assumption. If so, the second difference would not then be essential. Likewise, if an effective procedural memory can be based on returning only the best result – more like how rules work in ACT-R than in Soar 9 – the third difference may not be essential.

In addition to the differences just identified between procedural and declarative memory, three differences of note also showed up between the two implemented flavors of declarative memory: semantic and episodic. First, semantic memory searches for the most likely value for each attribute of an object individually – by marginalizing via sum-product – while episodic memory instead computes MAP estimation via max-product to retrieve the most appropriate single episode (where all of an episode's attributes jointly contribute to determining its appropriateness). Second, the probabilities of features in semantic memory are conditional on the concept while in episodic memory they are conditional on the time. Third, semantic memory is based on a general probabilistic representation of the values of attributes (see Figure 7), while episodic memory is based on the history of specific instances actually experienced (see Figure 9).

As with the differences between procedural and declarative memory, the first difference here appears to be fundamental, at least given this form of semantic memory. The other two differences appear less fundamental. It is possible, for example, to implement an instance-based semantic memory where the concept is just another feature. Sum-product can then dynamically compute more general feature distributions by summarizing over these instances. Interestingly, when max-product is used instead, the individual object that best matches the cues is retrieved, yielding something more like the semantic memory implemented in Soar 9. One intriguing implication is that the causative difference between generalization and analogy/CBR/nearest-neighbor may reduce to whether sum-product or max-product is used over an instance-based memory. The former generalizes over all instances, while the latter retrieves the single best instance.

Blended Functionality and New Capabilities

Beyond the three memories implemented above, the flexibility of the conditional representation enables blending of functionality across these memories (hope three) plus new capabilities beyond them (hope four). Blending arises from the flexibility with which conditions, actions, condacts and functions can combine within individual conditionals, plus the flexibility with which multiple conditionals can interact within long-term memory.

Conditionals by themselves enable combining procedural and declarative functionality within individual memory units. Semantic memory provides a good example. In addition to condacts and a function, each conditional can also include a condition that matches multiple objects in working memory. The prior is then represented by a conditional similar to the one in Figure 4, but with the constant `O1` replaced by a variable. The individual feature conditionals then resemble Figure 7, but with the condition added and the variable substituted (Figure 10). Like Soar 9, there is still a limit of one cycle of semantic memory retrieval per cycle of memory access – if quiescence of message passing in summary product is mapped onto quiescence of rule firing in Soar 9 – but unlike Soar 9, features of many objects can be predicted in parallel within this single cycle of memory access.

```
CONDITIONAL ConceptWeightGeneral
   Condition: Object(s,o)[α4]
   Condact: Concept(o,c)[α5]
            Weight(o,w)[α6]
```

Figure 10: Conditional distribution for semantic memory with condition to match objects (shown without function).

Other forms of within-conditional blends are also possible, such as combining conditions, actions and functions to yield weighted rules. Beyond this, to blend functionality across conditionals requires communication across conditionals that nominally belong to different memories, either via pattern variables within a single cycle of memory access or through working memory across cycles. The rule in Figure 11, for example, uses pattern variables to access the results of Figure 7's semantic retrieval, and generates a new `ConceptWeight` predicate. This also exploits within-conditional blending, but here in service of across-memory interaction.

```
CONDITIONAL ConceptWeightRule
   Condition: Object(s,o)[α4]
   Condact: Concept(o,c)[α5]
            Weight(o,w)[α6]
   Action: ConceptWeight(c,w)
```

Figure 11: Accessing semantic memory results in a rule.

Further work will be required to fully understand the range of capabilities this memory architecture might yield, and what the implications might then be for cognitive modeling. But at least one major new memory capability – for *constraints* – has already become apparent. Constraints are structures that specify restrictions on values assigned to variables (Dechter, 2003). Given a set of variables with

well-defined domains, and a set of constraints over these variables, constraint satisfaction determines which combinations of domain values are consistent with the constraints. Constraints are like rules in yielding all combinations of variable bindings, but like declarative memory in their flexibility of access, and thus in their use of condacts and an open world assumption. Figure 12 shows a constraint for the two-color problem, implemented via condacts and a Boolean function. The pattern variables for the two regions are shared with other constraints over those regions to enable appropriate propagation over the whole network during message passing. Although not a common form of long-term memory in cognitive architectures, except in neural systems based on "soft" constraints (Ackley, Sejnowski & Hinton, 1985), constraints do play a significant role in a variety of AI systems and languages.

```
CONDITIONAL TwoColorConstraint12
   Condact: Color(R1,c1)[α7]
            Color(R2,c2)[α8]
```

c1\c2	Red	Blue
Red	0	1
Blue	1	0

Figure 12: Two-color constraint between regions R1 & R2.

Summary

Basing a memory architecture on the uniform breadth of graphical models has enabled straightforward construction of four distinct memories: a rule-based procedural memory, semantic and episodic declarative memories, and a constraint memory that is functionally a hybrid between the two. These implementations reveal significant commonality among these memories, but also subtle differences. Of the differences, unidirectional versus bidirectional message passing appears to be most fundamental when comparing procedural and declarative memories, while marginalization versus MAP estimation appears to be most fundamental when comparing semantic and episodic memory.

Implementing memories in this manner also enables blending capabilities across memories and creating new unanticipated kinds of memories, such as a constraint memory. This general approach holds the promise of extending beyond memory architecture to full cognitive architectures with mechanisms for decisions, learning, and perceptuomotor behavior. The hopes for this larger effort would be to derive a better understanding of: the diverse mechanisms involved, including their commonalities and differences; how they can and should work together; and how to go beyond the kinds of combinations currently seen to simpler yet more comprehensive cognitive architectures.

Acknowledgements

This effort has been sponsored by the USC Institute for Creative Technologies and the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred. I would like to thank Bill Swartout for help in restructuring this work for publication.

References

Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147-169.

Anderson, J. R. (1990). *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1993). *Rules of the Mind.* Erlbaum.

Dechter, R. 2003. *Constraint Processing*. San Francisco, CA: Morgan Kaufmann.

Domingos, P. & Lowd, D. (2009). *Markov Logic: An Interface Layer for Artificial Intelligence.* Morgan & Claypool.

Forgy, C. L. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, *19*, 17-37.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press.

Kschischang, F. R., Frey, B. J. & Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498-519.

Laird, J. E. (2008). Extending the Soar cognitive architecture. *Artificial General Intelligence 2008: Proceedings of the First AGI Conference*. IOS Press.

Rosenbloom, P. S. (2009a). Towards a new cognitive hourglass: Uniform implementation of cognitive architecture via factor graphs. *Proceedings of the 9th International Conference on Cognitive Modeling.*

Rosenbloom, P. S. (2009b). A graphical rethinking of the cognitive inner loop. *Proceedings of The IJCAI International Workshop on Graph Structures for Knowledge Representation and Reasoning*.

Rosenbloom, P. S., Newell, A. & Laird, J. E. (1991). Towards the knowledge level in Soar: The role of the architecture in the use of knowledge. In K. VanLehn (Ed.), *Architectures for Intelligence*. Hillsdale, NJ: Erlbaum.

Sims, C. R. & Gray, W. D. (2004). Episodic versus semantic memory: An exploration of models of memory decay in the serial attention paradigm. *Proceedings of the 6th International Conference on Cognitive Modeling* (pp. 279-284).

Sun, R. (2006). The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction*. New York, NY: Cambridge University Press.

Modeling a Three Term Fan Effect

Matthew F. Rutledge-Taylor (mrtaylo2@connect.carleton.ca)
Institute of Cognitive Science, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Aryn A. Pyke (apyke@ccs.carleton.ca)
Department of Psychology, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Robert L. West (robert_west@carleton.ca)
Institute of Cognitive Science, Department of Psychology, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Hana Lang (hlang@connect.carleton.ca)
Institute of Cognitive Science, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Abstract

A fan effect experiment where participants perform recall and recognition tasks on a study set of sentences with three content words was conducted. The aggregate results confirm a fan effect (Anderson, 1974). A model of the recall and recognition tasks was created using Dynamically Structured Holographic memory (DSHM). A comparison to the human data is presented. A discussion of the current resonance based mechanisms in DSHM for generating recognition accuracy and reaction time data is presented. This is contrasted with a previously employed retrieval based mechanism.

Keywords: cognitive modeling; the fan effect; holographic reduced representation.

Introduction

The purpose of this paper is to report the results of a fan effect style experiment and to demonstrate that these results can be captured by Dynamically Structured Holographic memory (DSHM). The experiment conducted was similar to the classic fan effect paradigm (Anderson, 1974).

In Anderson's original experiment, participants studied a set of sentences that contained two content words: a person and a place (e.g., "the hippie is in the park"). Each content word appeared in one, two, or three different sentences. The number of sentences in which a word appears is the *fan* of that word. Each sentence is assigned a fan, which is the sum of the fans of the content words in the sentence. For example, if 'hippie' appeared in three sentences while 'park' appeared in one sentence, 'hippie' would have a fan of three, 'park' would have a fan of one, and the sentence 'the hippie is in the park' would have a fan of four. The results of a recognition task performed on the sentences (and an equal number of foils) demonstrated that the time required to affirm or reject a sentence as a member of the study set was correlated with the fan of the sentence.

The present work extends prior research on the fan effect, and models thereof. We explore the generality of the fan effect by examining memory performance for sentences with three content terms rather than just two (e.g., Anderson, 1974). Additionally, our sentences had a wider range of fans than have typically been studied (or modeled).

The Three Term Fan Experiment

Method

Twenty seven participants (12 males and 15 females: mean age 20.0 years, SD = 2.2) were recruited from introductory psychology courses a Carleton University to take part in the experiment. Participants received course credit for their time. Participants took part in the experiment one at a time. The experiment was divided into three main phases: A study phase, a recall phase and a recognition phase.

In the study phase each participant was assigned one of three unique sets of study sentences and was instructed to memorize the sentences in the list. Once the participant indicated that he or she was prepared to proceed, the recall portion of the experiment began.

The study set consisted of sixteen sentences of the form, "The *color thing* is in the *place*". The color term was one of ten colors; the thing was one of ten house-hold items; and the place was one of ten locations in/around a typical home. Very typical item/locations combinations, such as 'comb'/'bathroom', were omitted when generating the study set sentences. Eight terms from each category appeared in one study sentence each, while two terms from each category appeared in four sentences each. No two terms appeared together in more than one sentence. For example, if "The orange comb is in the garage" was a member of the study set, no other sentence in the study set described an orange comb, a different colored comb in the garage, or any other orange object in the garage. However, these combinations could occur in foil sentences.

The fan of a sentence is the sum of the fans of the terms in the sentence. Thus, the four possible sentence fans were:

3, 6 9, and 12. The fan effect predicts that judgments for sentences with higher fans should take longer (i.e., have higher reaction times) than for sentences with lower fans. Additionally, the truth of sentences with a higher fan should be recognized with less accuracy than sentences with a lower fan.

Recall Task Method

Each participant engaged in three iterations of the recall task. Each iteration began with the participant trading the study sentences list with the experimenter for a new list of sentences identical to the study set, but with one term from each sentence replaced with a blank, and the order of the sentences randomized. The participant's task was to correctly fill-in each of the blanks with the missing word. The participant was given as much time as he or she needed to do so. The experimenter then recorded the number of correct responses and for each error, provided the correct missing word to the participant. The participant was then given the opportunity to review the study set again. The three iterations were balanced such that each term from each sentence in the study set was replaced with a blank exactly once. After the third iteration the recognition phase began.

Recognition Task Method

The recognition task was conducted on a computer using the Experiment Builder software package from SR Research. Sentences were presented one at a time, centered on a 17" CRT monitor (in black font on a white background). Participants judged whether each presented sentence was a member of the study set, or not. To respond, participants hit either the z-key or the /-key, respectively. Accuracy and reaction time were recorded for each trial. After each trial, the screen blanked for 1 second, and then the word "READY" appeared for 1 second to prepare the participant for the next trial.

The participant was presented with 96 test sentences, which consisted of three exposures to each of the study set sentences, and 48 foil sentences which were not from the study set. Participants were told that they should consider sentences from the study set to be *true*, while all others should be considered *false*. Each false sentence was generated by replacing one of the three terms from a true sentence with another term from the same category (e.g., color, thing, or place) and with the same fan. For example, for a true sentence like "The blue hat is in the garage", one false counterpart might be "The green hat is in the garage". Each true sentence was used to generate three different false sentences. Thus, for each exposure of a true sentence there was a corresponding false test sentence with the identical fan.

Results

The data from one participant was excluded from the analysis below. This participant's recognition reaction time was significantly longer than all the other participants by a large margin ($P < .001$). The results below reflect the data collected from the remaining 26 participants.

Human Recall Performance

Performance in the recall task improved, on average, with each of three iterations. Table 1 presents the mean number of correct responses (out of 16), the standard deviation, and the accuracy measured as a percentage for each of the three iterations of the recall phase.

Table 1: Recall accuracy

	Iteration		
	1	2	3
Correct (/16)	10.9	13.4	14.6
SD	3.7	3.1	1.8
Percentage	68.1	83.8	91.4

This result is important because an intended purpose of the recall task was to confirm that the participants had memorized the study set before entering the recognition phase. By the end of the third iteration the participants were correctly completing the sentences 91.4 percent of the time.

Human Recognition Performance

Overall, participants' accuracy and reaction time results were consistent with the fan effect. For both true and false sentences, accuracy was negatively correlated with sentence fan. Also, accuracy was poorer for false sentences than for true sentences for all sentence fans ($ps < .05$).

Table 2: Recognition accuracy (%)

	Accuracy	
Sentence fan	True	False
3	97.5	95.5
6	95.1	91.7
9	92.1	86.3
12	82.7	77.6

Reaction time increased with sentence fan ($p < .001$) for both true and false sentences, and true sentences were judged more quickly than false ones ($p = .001$).

Table 3: Recognition reaction time (ms/char)

	True		False	
Sentence fan	Reaction time	SD	Reaction time	SD
3	59.0	18.5	64.1	19.9
6	63.6	20.0	69.3	22.6
9	74.2	21.8	86.2	31.1
12	91.3	31.5	102.5	43.6

Table 3 shows the reaction times (ms/char) for both true correct (i.e., the test sentence was true and was judged correctly) and false correct sentences of each fan.

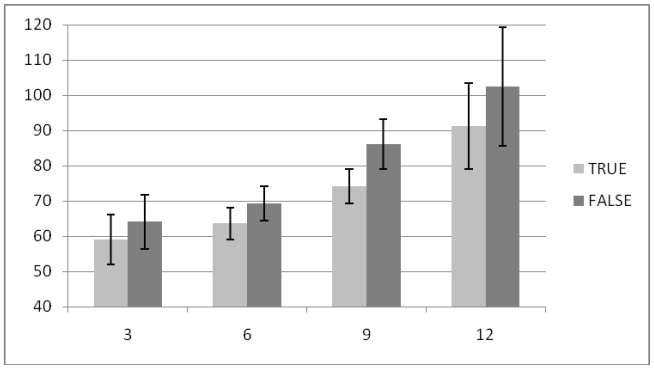


Figure 1: Recognition reaction time by sentence fan (ms/char) with confidence intervals

Figure 1 shows the mean reaction times, measured in ms/character, for both true correct and false correct sentences, for each sentence fan with confidence intervals. There was no interaction of truth and fan (p = .199). Table 4 presents the pairwise comparisons across fan using the Bonferonni adjustment.

Table 4: Pairwise comparisons for correct reaction times

Sentence fans	P (one-tail)
3 versus 6	0.129
6 versus 9	< 0.001
9 versus 12	< 0.001

In Summary

The results of the experiment confirm the fan effect as a robust phenomenon that generalizes from sentences with two content terms (Anderson, 1974) to sentences with three content terms (present research). Future work will examine whether statistically significant differences can be found in the relative contributions of the terms to the fan effect (e.g., does the *color* term contribute differently than the *thing* or *place* terms).

DSHM

The memory modeling system used to model the described experiment was Dynamically Structured Holographic Memory (DSHM) (Rutledge-Taylor & West, 2008). DSHM is based on the BEAGLE model of the lexicon (Jones & Mewhort, 2007). The details of the DSHM architecture and the similarities between BEAGLE and DSHM can be found elsewhere (Rutledge-Taylor & West, 2007).

For an account of the use of DSHM to model the classic fan effect, and a comparison to ACT-R (Anderson & Lebiere, 1998), see Rutledge-Taylor and West (2008). For those unfamiliar with DSHM, a brief introduction follows.

DSHM makes use of holographic reduced representations (HRR) to encode knowledge in memory. See Plate (1995) for a discussion of the sort of HRRs used by DSHM (and BEAGLE). A DSHM system is composed of a collection of items that are represented internally as two vectors of numbers: i) the environmental vector is static and uniquely identifies the item in the system; ii) the memory vector is dynamic and encodes all of the associations an item develops with other items. The lengths of these vectors are fixed for an instance of DSHM, but can be initially set to any positive integer which is a power of 2.

DSHM takes collections of items as input (called complex items; collections of items are items themselves). The structure of a complex item can be expressed using left and right brackets. For example the sentence "The red hat is in the garage" can be expressed [red:hat:garage]. The system can also allow items to have a hierarchical structure. Here, the context tags used to classify an item as background knowledge (false) versus experimental knowledge (true) applies to the sentence as a whole, and so is up a level in the hierarchy, expressed: [true [red:hat:garage]]. Items can bear ordered (delimited by colons) or unordered (delimited by spaces) relationships with one another.

Information is extracted from DSHM by presenting it with incomplete complex items. For example, a query for the color of an item might be expressed [true [?x:hat:garage]. Any missing items are called query items and in DSHSM syntax are always preceded with a question mark, (e.g., "?x"). A query item is like a variable that DSHM is tasked with resolving. DSHM makes use of information stored in the memory vectors of the provided items to generate a rank ordered list of candidate items for replacing the query item. Each candidate completion is accompanied by a numerical value ranging from 0.0 to 1.0 that indicates the strength of the completion. This strength is referred to as the confidence (i.e., how confident DSHM is in the completion being correct, or appropriate). It can also be thought of a context relative activation value, to use an ACT-R term.

A DSHM model is constructed by making choices about how information is represented in complex items, what vector size should be used, what training regime is used, and what sorts of queries are presented to the system.

The Model

Twenty-seven simulated participants were run (to correspond to the 27 human participants). It was found that a range of vector lengths allowed the simulated participants to produce reasonable recognition accuracy and reaction time results. However, fitting the recall data was more of a challenge. Uniformly using a vector length of 64 produced significantly poorer performance than the average for the human participants, while a vector length of 128 produced significantly better results. No value in between is possible (vector length must be powers of 2). In order to produce good average scores, nine of the simulated participants were given memory systems that made use of vector lengths of 64, while the other 18 used vector lengths of 128.

Study Phase

Prior to learning the study set, each simulated participant read 1026 background knowledge sentences, each encoded as a flat ordered list of three content terms associated with a tag 'false'; "[false [color:thing:place]]". The false sentences included either one or two of the content terms appearing in the study set. The remaining one or two terms were nonsense terms that did not occur in the study set sentences. The background knowledge was needed in order to give the simulated participants some basis for making errors. Without background knowledge there is nothing for DSHM to confuse the study set sentences with; DSHM does not make use of explicitly added noise.

The simulated participants read each sentence in the study set once or twice (to account for the differences in how well the human participants prepared themselves for the first task) prior to beginning the recall phase. Sentences from the study set were associated with a context tag representing 'true'; "[true [color:thing:place]]".

Recall Performance of the Model

Like the human participants, the simulated participants produced responses to fill-in the blank questions in the recall phase. For example, "The _____ hat is in the garage" was submitted to the DSHM participant as "[true [?x:hat:garage]]". The system outputs a list of candidate responses, in rank order. The one with the highest rank was considered to be the simulated participant's response. If the system's response item matched the correct missing term, the trial was scored as correct.

After each iteration the DSHM participant read each of the study set sentences once for every three incorrect responses on the previous iteration. The majority of human participants took the opportunity to review the study set, even after scoring perfectly on the previous iteration. Thus, the DSHM participants re-read the study set a minimum of once between trials.

Table 5: Model recall accuracy

	Iteration		
	1	2	3
Correct	11.1	14.1	14.1
SD	3.2	2.1	1.9
Percentage	69.4	88.2	88.2

Table 5 presents the recall accuracy for the simulated participants. Although, the accuracy plateaus after the 2nd iteration, there is an overall good match for accuracy and standard deviation, as demonstrated in figure 2 (only the standard deviations for the human data are shown).

Recognition Performance of the Model

The simulated participants were each tested on the same 96 test sentences as the human participants. In order to produce a truth judgment the simulated participant was presented with a query of the form "[?x [color:thing:place]]". If the system produced 'true' as its highest ranked completion candidate, the simulated participant was considered to have judged the sentence to be *true*, otherwise, the simulated participant was considered to have judged the sentence to be *false*.

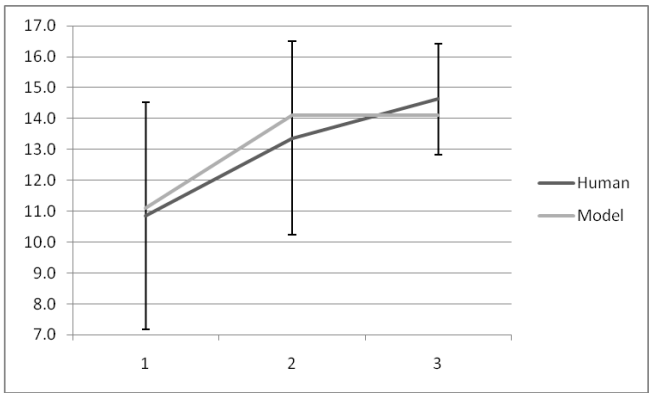


Figure 2: Recall accuracy (out of 16)

To determine the reaction time for the response, the model evaluated the degree to which the test sentence as a whole (without a context tag) ("[color:thing:place]"), resonated within the system. The sentence's resonance is produced by a built-in DSHM method, which essentially determines how closely associated the terms in the sentence are to one another. Here, the resonance value is interpreted as indicating how familiar the sentence seems to the simulated participant. Thus, if the sentence is judged to be true, a high resonance should make this decision easier. If it is lower, it should make the decision harder. The opposite is the case for judgments of false. It should be difficult to reject a sentence that seems familiar, and vice-versa.

The formula used for translating resonance to reaction time was RT = 32 / R, where R is the value provided by the memory system of the simulated participant, and RT is reaction time measured in ms per character. For true sentences R is the resonance value for the sentence. For false sentences, R is the resonance value for the sentence subtracted from an upper limit on resonance values. This upper limit was estimated to be the maximum resonance value calculated for any of the true sentences (0.64). Table 6 presents the reaction time data for the model.

Table 6: Model reaction time (ms/char)

	Reaction Time	
Sentence fan	True	False
3	61.1	67.4
6	69.0	69.5
9	79.6	77.1
12	87.8	94.5

Figure 3 presents a comparison of the human and model

data for correct trials. The solid lines correspond to the human data; the dashed lines correspond to the model data; the light lines correspond to the true data; and the dark lines correspond to the false data.

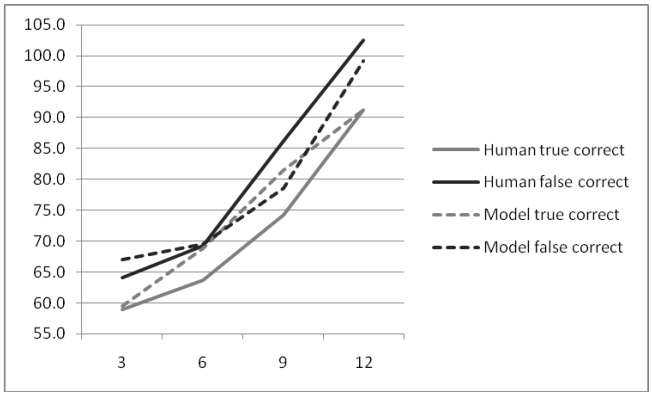


Figure 3: Recognition reaction time

In terms of judgment accuracy, the model outperformed the human participants. The simulated participants had a judgment accuracy of 100% for true sentences and 98.5% for false sentences. It is possible that this discrepancy may be due to the relatively small body of 'interfering' background knowledge in the simulated participants relative to real human participants.

In Summary

In general the model results provide a good match to the human data, in that 1) the false sentences take longer, on average, to affirm or deny than do true sentences (77.1 ms/char versus 74.4 ms/char); 2) a fan effect is observed for both true sentences and false sentences; 3) the model provided a good fit to recall performance as well as recognition performance; and 4) the formula used to convert raw model output to reaction time values is simple and provides a good fit to the recognition times using a single scaling parameter.

On-Going Work: Effect Of How Fan Is Distributed?

Part of the motivation for this experiment and model construction was to investigate whether each content word in a sentence contributes equally to the difficulty in recognizing a sentence as true (i.e., a member of study set). It was hypothesized that the color term may make a smaller contribution to the fan effect than the thing or the place. This is because the color terms are adjectives and more ubiquitous than the things or places, which are nouns. However, whether the thing or the place should carry more weight was not predicted given conflicting intuitions about why one or the other should be more influential. For example, the thing term might be the most influential because an object's type (e.g., hat) is a more intrinsic property than its location (or color). Alternately, place might be more influential: Grammatically, the color and thing share a common noun phrase, while the place does not share its prepositional phrase with any other content word.

The human data were not clear cut with regard to the influence how fan was distributed among content terms. By fan distribution, we are referring to the possible pattern of the fans of the words making up sentences with a particular fan. There are three different ways to make fan 6 sentences (color term fan = 1, thing = 1, place = 4; 1,4,1; 4,1,1), and three ways fan 9 sentences (1,4,4; 4,1,4; 4,4,1), while there is only one way to make fan 3 sentences (1,1,1) and one way to make fan 12 sentences (4,4,4).

No significant effects of fan distribution were found among fan 6 sentences. But, among sentences with a fan of 9, an ANOVA with revealed that fan distribution did have an impact on RT (p =.002). Specifically, RT was faster when either the thing or place was unique (i.e., fan 1) and slower when the color was unique. Put another way, when trying to judge whether a sentence is true (e.g., "The red hat is in the garage"), knowledge of other objects with the same color (red ball) adds less difficulty than knowledge of other items of the same type (hat) or other items in the same place (garage). Further, RTs tended to be faster when the thing type was unique rather than the location, though this trend did not reach significance.

Note: a simple variation in the representation of sentences in DSHM would be able to account for this effect because DSHM is capable of representing facts that have hierarchical structure. In fact, DSHM already leverages this capability in the current model. In the representation "[true [color:thing:place]]", the three term sentence as a whole aggregate is the hierarchical sibling of the context tag ('true'). In order to represent sentences where the thing term is dominant, the color and place need only be embedded in a list of peripheral properties as in the following representation: "[true [thing:[color:place]]".

Exploratory simulations confirm that using this type of representation predicts differences in reaction times among fan 9 sentences, where the thing fan dominates the fans of the other two terms. Similarly, for sentences with an overall fan of six and a thing fan of four are significantly slower than fan 6 sentences with a color fan of four, or a place fan of four. Additional human testing is required to gather more information about the effects of fan distribution. But it is noteworthy that such hierarchical effects could be naturally afforded by structural aspects of a DSHM architecture. This line of research is on-going.

Appendix

Relationship To the Two Term Model

Rutledge-Taylor and West (2008) presented a model of the fan effect, as described in Anderson (1974). This model provided a good match to the human data, but used a different mechanism for calculating recognition accuracy and reaction time values, than the one presented here. This mechanism, which we will refer to as the 'retrieval'

mechanism operates as described below.

Whether DSHM recognizes a sentence, or not, according to the retrieval mechanism is based on how strongly the words in the sentence are associated with one another. Specifically, if at least one of the words in the sentence (referred to as a target word) can be recovered using the other words in the sentence as cues, the sentence as a whole is recognized (as true), otherwise, it is not.

If the sentence is recognized, the reaction time is based on strengths (e.g., confidence values) of the recovered target words, which are high on average, resulting in low reaction times. If the sentence is not recognized, the reaction time is based on strengths of the words that were retrieved (but did not match the target words). On average the strengths of these retrieved words are lower, resulting in higher reaction times. Additionally, the fans of the words in the sentences affect the strengths of the retrieved words and it these strengths that are the basis for the fan effect in the DSHM model.

Using The Retrieval Mechanism In The Three Term Model

The retrieval mechanism for generating recognition and accuracy results for the DSHM model was initially tested on the current stimuli and without using background knowledge sentences, which are not necessary for this mechanism. The retrieval mechanism produced a 100% accuracy rate for identifying true sentences, but only a 36% accuracy rate for rejecting false sentences.

The retrieval mechanism produced a very good fit to the human true correct reaction times, including the characteristic exponential curve observed in the human data (for both trues and falses). However, the model results for false correct (i.e., correct rejections) reaction times were drastically different from that of the human data. See figure 4.

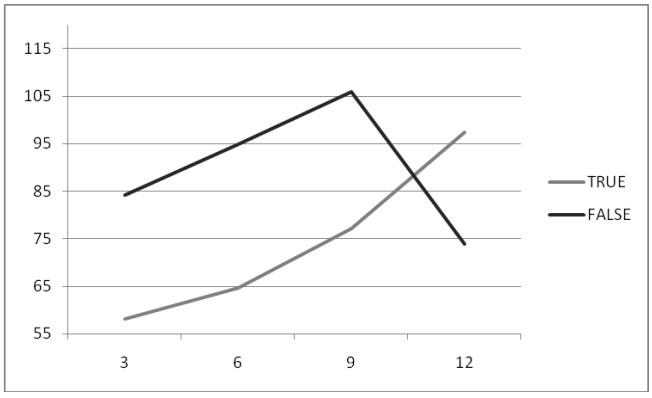


Figure 4: Reaction times (ms/char) using the retrieval mechanism

The explanation for the model's false correct data has to do with the number of true near neighbors the false sentences have. Here, 'near neighbors' are defined as two sentences that differ only by a single word. The number of near true neighbors a false sentence has is correlated with its fan. This is the result of the counter-balancing of true and false sentences. The existence of near neighbors makes little difference in the recognition results for false fan 3, 6 and 9 sentences. However, for fan 12 sentences there are true near neighbors that are retrieved (for each target word) with very high strengths. This results in low reaction times for false sentences with a fan of 12. For example, when presented with the false sentence "the black mug is in the garage", the true sentence "the grey much is in the garage" is retrieved with a high confidence value, when internally testing to see if "[?x:mug:garage]" retrieves 'black' as a candidate completion of the query term '?x'.

Due to the failure of the retrieval mechanism to provide a satisfactory account of the reaction times for correct rejections, the new mechanism described above was developed. It is the authors' belief that the retrieval mechanism ought to work for most DSHM models under most circumstances. However, in cases such as the one presented here, the new mechanism can be applied in order to generate recognition reaction times for correct rejections that are resistant to the effects of true near neighbors.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6,* 451-474.

Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and Order information in a composite holographic lexicon. *Psychological Review, 114*, 1-37.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks, 6,* 623-641.

Rutledge-Taylor, M. F. & West, R. L. (2007) MALTA: Enhancing ACT-R with a holographic persistent knowledge store. *Proceedings of the XXIV Annual Conference of the Cognitive Science Society*. Nashville, TN.

Rutledge-Taylor, M. F. & West, R. L. (2008) Modeling The fan effect using dynamically structured holographic memory. *Proceedings of the XXX Annual Conference of the Cognitive Science Society*. 385-390. Washington, DC

A Computational Account of Complex Mental Image Construction

Jan Frederik Sima (sima@sfbtr8.uni-bremen.de)
SFB/TR 8, Universität Bremen, Germany

Abstract

This paper presents a computational cognitive model of the construction process of complex, i.e., multi-part, visual mental images. The model is integrated into the cognitive architecture Casimir. The construction process is realized by the interplay of a spatial working memory structure and a passive quasi-pictorial visual representation. Both structures are successively build up on demand from long-term memory. The correct placement of new parts is guided by the inspection of the visual representation. The model has two main advantages: 1) it is an explicitly cognitive computational model that implements the two-fold structure of a spatial and a visual working memory representation and 2) it introduces an attention window structure in such a way that allows for direct predictions of eye movements during mental imagery processes. We discuss predictions and explanations offered by model.

Keywords: Cognitive Modeling; Visual Mental Imagery; Visual and Spatial Representations; Analogical Representations

Introduction

The experience of visual mental imagery is a well-known and widely studied phenomenon. For example, many people report to actively use mental imagery for common visuo-spatial tasks, such as planning a route. Additionally, imagery plays an important role in a number of diverse domains such as diagrammatic problem solving and creativity (e.g., Hegarty, 2004). Furthermore, the general efficiency and usefulness of a visual or quasi-pictorial representation compared to a purely symbolical, i.e., non-analogical, representation for several reasoning domains has been shown and argued for extensively from an artificial intelligence point of view (e.g., Chandrasekaran, Kurup, Banerjee, Josephson, & Winkler, 2004).

Almost all computational accounts of visual mental imagery that have emerged since Kosslyn's computational cognitive model (Kosslyn, 1980) thirty years ago were not designed as cognitively plausible accounts of human imagery processes, but adopted single findings, e.g., most prominently the existence and distinction of two, one spatial and one visual, representations involved in mental imagery (see for example Glasgow & Papadias, 1992). There has been work to extend well-established cognitive architectures, e.g., ACT-R and Soar, with the functionality of visual mental imagery and even though these accounts provided valuable insights, for example regarding the structural integration of imagery into an architecture, they remained on a conceptual level (Gunzelmann & Lyon, 2007) or were explicitly not designed as cognitively plausible models (Lathrop, 2008).

As Kosslyn's computational model (Kosslyn, 1980) is the most relevant and also closest in its approach to our model, it is worthwhile to make the major differences clear. First off, it is to note, that Kosslyn (1994) himself significantly altered his theory of mental imagery in the light of new empirical and neuroscientifc data. His new and very extensive conceptual model has, however, never been implemented. In contrast to his implemented model, we employ two working memory structures: the visual one roughly corresponds to Kosslyn's visual buffer, the other spatial one has no counterpart in his model. Another important difference is the existence of an attention window in our model, which implements the selective attention on the content of a mental images as well as the multi-scale property of the visual representation.

The aim of the presented model is to offer a plausible explanation of how complex visual mental images are constructed from long-term memory. The computational implementation allows the identification of open empirical issues as well as new predictions regarding the involved processes in mental imagery. By employing two working memory structures of different abstraction, we offer a straightforward account for the findings that suggest two distinct kinds of imagery (Levine, Warach, & Farah, 1985; Farah & Hammond, 1988) and also shed new light on the question of many imagery phenomena such as mental image reinterpretation. The implemented attention window of the model allows us to directly link attention shifts in the visual representation to eye movements made by subjects in imagery experiments and thus offers a new method of evaluation for models and theories of imagery.

The model is designed within the framework of Casimir (Barkowsky, 2007), a cognitive architecture for spatial knowledge processing with analogical representations.

In the following sections, we will describe the design of the model's representation structures and processes and how those are derived from general assumptions of human cognition as well as from empirical data on several related imagery phenomena.

The Model of Image Construction

To begin, we define the domain the model is applied to. When referring to mental images, we always mean consciously experienced visual mental images. The model is constrained to mental images that are generated from information that is retrieved from long-term memory with the absence of any other visual input, e.g., visual perception. The mental images we deal with are labeled "complex" in the sense that the visualized concepts consist of several parts. For example, the concept *house* consists of a main block, a roof, a door and a window; further, the parts themselves may have subparts, e.g., chimney is a part of the concept *roof*. The mental images are constructed so that they are "seen" from an egocentric perspective similar to an actual visual percept.

Figure 1 shows the basic components and interactions of the model. We have modeled the spatial and visual representations as well as the processes involved in the construction of a mental image. The long-term memory component

is an existing part of the cognitive architecture Casimir (see Schultheis, Barkowsky, & Bertel, 2006, for details).

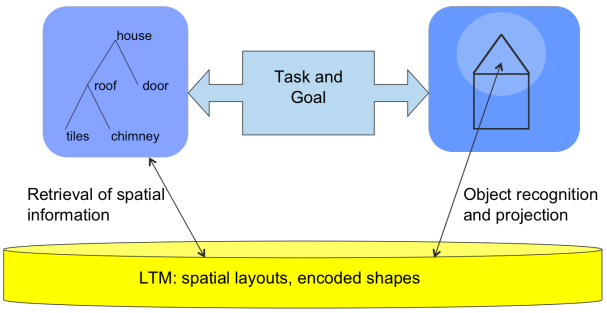


Figure 1: Representations and Processes of the Model. The visual representation serves as an extension of the spatial representation. Shapes are projected into the visual representation according to the spatial layout stored in the spatial representation.

Basic Design Constraints

In this section we will briefly elaborate the theoretical background upon which the general design decisions of the model are based. For this purpose, we describe the basic assumptions that the model makes about visual mental imagery and working memory in general.

Parsimony. The model is generally designed to keep the processes and corresponding representation structures as parsimonious as possible. The model's workflow is designed so that it works strictly on demand. This means, that each transfer and transformation of information between long-term memory, the spatial representation and the visual representation is only triggered when demanded by the current task. Accordingly, concepts can be visualized at different levels of granularity and enriched with more details when necessary.

Analogical representation structures. The model is based on the main assumptions of what is often labeled the quasi-pictorial theory of mental imagery, see (Kosslyn, 1994) for its most popular representative. That is, the structure or structures, which the experience of visual mental images relies on, at least partly represent/preserve the spatial properties of an actual image/the actual visual percept in an analogical format. Given the existing empirical support, there is widespread agreement on this hypothesis (e.g., Finke, 1989). As the visual representation in the model actually depicts shape it is apparently analogical, but also the spatial representation has an analogical format as it preserves the part-of relation of complex entities in its structure.

Distinction between visual and spatial knowledge processing. Within the model visual and non-visual information is distinguished on different but interdependent levels: 1) the model employs two working memory structures, 2) visual and non-visual information is retrieved separately by separate subprocesses from long-term memory.

Building upon the findings that the two cortical visual pathways first identified by Ungerleider and Mishkin (1982) can also be distinguished in human visuo-spatial working memory (Courtney, Ungerleider, Keil, & Haxby, 1996), it has been argued that two representations involved in imagery can be functionally and neurologically dissociated (Levine et al., 1985). This conclusion is based on studies with brain-damaged patients, who were able to perform normally on some imagery task but were impaired on other imagery tasks. These two groups of imagery tasks corresponded to what is usually considered to be visual imagery tasks and spatial imagery tasks respectively (Farah & Hammond, 1988).

As evident in figure 1 we assume two information pathways which together give rise to complex images in the visual representation. On the one hand, processes associated with the ventral pathway are responsible for the processing of shape information, i.e., the recognition of shape and the retrieval and projection of shape information from long-term memory into the visual representation during imagery. On the other hand, the spatial representation is associated with the dorsal pathway which processes the spatial layout of an entity or scene.

Besides fulfilling all other structural requirements for models of mental imagery as identified by Bertel, Barkowsky, Engel, and Freksa (2006), our model specifically fits into their category of hybrid models, as two representations of different qualitative structure are combined. They proposed that a computational cognitive model of mental imagery needs to have a hybrid structure in order to plausibly capture the "hybrid, exhibiting both visual and propositional traits" (Bertel et al., 2006) nature of mental images.

Evidence for a dedicated non-visual working memory structure involved in visual perception, has led to approaches (e.g., Nestor & Kokinov, 2004), which, similar our model, employ a visual and a non-visual working memory structure in this respective domain.

Components and their Interaction

Following, we will describe the structure of both the spatial and the visual representation in more detail as well as the interaction between them.

The visual representation is implemented as a graphics window, in which geometric shapes are drawn. The circular attention window determines which parts of the representation are currently attended to and can be processed. The attention window is defined by its position and by its resolution. The higher the resolution, the smaller the extent of the attention window and thus only a smaller part of the visual representation is accessible for inspection. Furthermore, the resolution also determines what contents of the visual representation are "visible", i.e., can be processed, depending on the size of the visualized shape. For example, small parts or details such as texture are only accessible if the resolution is high, whereas bigger parts are also visible at a low resolu-

tion. The attention window implements two concepts: 1) the selective processing of visual information and 2) the scale-resolution trade-off in the inspection of mental images, which goes along with the multi-scale property of the topographically organized areas of the visual cortex (Kosslyn, 1994).

The spatial representation contains the minimal necessary spatial layout information of a concept. For the concept *house* the minimal layout consists of a location[1], orientation and size of the basic shape of *house* as it is visualized or to be visualized in the visual representation. Note that these parameters can be set by the task, e.g., "Imagine a small house, that is tilted 90 degrees clockwise", but lacking any of those demands, the parameters will be set by associations from long-term memory. The spatial representation does not include the shape or any further information about the shape other than the rough size it is (to be) visualized in. The minimal layout further includes the direct and most strongly associated parts of the concept *house* as identifiers, their spatial relations to the basic part, e.g., "on the left top of", as well as their relative size compared to the basic part of *house*.

The relative size of a part is important to determine if and when it is visualized in case an elaborate image of the current concept is demanded, i.e., the bigger the part is relative to the basic shape[2] of a concept, the earlier it will be visualized. That is, if a detailed image of *house* is requested, *roof* will be visualized first, followed by *door* and *window*. We assume that this size-depended sequence might change if one particular part has a very strong association with the super concept, but as a default the relative size is assumed to determine the sequence. This is a consequence of the nature of the attention window and we further elaborate on this aspect below.

If a new part, e.g., the door of the house, should be visualized, the concept *door* is retrieved from long-term memory and extends the spatial representation. This means that it now includes information about *door*; orientation, size and location are in this case determined by the super concept *house*, e.g., if we imagine a small 90 degrees tilted house, all its parts and subparts will by default also have these properties. Parts of *door* are now also consciously available. The retrieval of new information is context-dependent as it is affected by the current content of the spatial and visual representation, that is, in particular the super concept, e.g., the model would produce a different mental image of a window by itself than of a window as part of a house.

Interaction between components. There is a hierarchical structure between the long-term memory, the spatial representation and the visual representation, that is, information is retrieved and transformed from long-term memory first into the spatial representation and parts of these informations are transferred on demand into the visual representation, where the resulting shape is visualized. As evident in figure 1, there is a direct connection between encoded shapes in long-term memory and the visual representation, but this projection process is triggered only if parts of the spatial representation need to be visually accessed. The represented information on these three levels differs quantitatively as well as qualitatively: 1) there is information available in the spatial representation which is not visualized, i.e., not represented, in the visual representation; similarly the information in the spatial representation is only a fraction of what is available in long-term memory; 2) furthermore, only the visual representation explicitly contains visual information, such as shape or texture, which by themselves lack semantics (which are contained in the spatial representation). Additionally, this hierarchical structure implies that certain tasks, which do not depend on visual information can be solved solely on the level of the spatial representation and do not have to use the visual representation.

The Image Construction Process

In order to describe the construction process of a multi-part visual mental image in the model, we will go through the individual steps taken to build an image.

- The model is given the command to imagine the concept *house*.
- The *spatial representation* (SR) queries the *long-term memory* (LTM) for the minimal spatial representation of *house*. As no further context is specified, a default location *L*, orientation *O* and size *S* are used for the query.
- The *attention window* (AW) is shifted to location *L* and its resolution adjusted to fit the size *S*.
- The *visual representation* (VR) retrieves the basic shape of *house* with the given size *S* and orientation *O* from LTM and it is visualized at the center of the AW.
- The SR is queried for direct parts of *house* that are of the same relative size as *house* and finds *roof*. It will automatically be visualized given the current resolution of the AW.
- The shape of *house* in the VR is inspected to find the coordinates where to place *roof* according to the given qualitative spatial relation between *roof* and *house* from the SR.
- The AW to the determined location.
- The SR retrieves further spatial information about *roof*; this information includes parts of *roof* and will allow for a later visualization of parts of *roof*.
- The shape of *roof* is retrieved from LTM with size *S* and orientation *O*, which are both inherited from the parental concept *house*. The shape is projected into the center of the AW.
- The SR does not find any other direct parts of *house* with a relative size that would allow visualization given the current resolution. Thus the model stops.

The above process sequence builds the minimal image of the concept *house*. Further parts such as *door* or *window* are

[1] Location within the visual representation.

[2] Following a similar principle the basic shape or main part of a concept figures to be the bigger than any of its parts.

not added, even though their existence, relative size and spatial relation are "known", i.e., are consciously available in the SR. The model always builds minimal images unless the task demands further details to be added.

Lets look at an excerpt of the construction process for a detailed image of *house*. We assume the state of the model to be the last described state of the previous process sequence, i.e., the basic shape of *house* and *roof* are visualized.

- When a detailed image is requested all directly related parts to *house* are visualized in the order of their relative size. The SR finds *door* as a part of *house*.
- The basic shape of *house* in the VR is inspected and the appropriate docking coordinates for *door* are calculated and the AW is shifted to this position.
- As the relative size of *door* is smaller than that of *house* the resolution of the AW is adjusted, i.e., higher resolution, lesser extent.
- The SR retrieves spatial information about *door*.
- The shape of *door* is retrieved and projected at the position of the AW in the VR.
- This process goes on similarly for all direct parts of *house*.

Explanations and Predictions

The model makes some novel assumptions which offer new and concrete explanations of common imagery phenomena and also lead to precise predictions about human behavior during mental imagery. We will briefly look at how the model is able to account for those common phenomena of mental imagery. We have not yet started to fit concrete empirical data, but the structure of the overall model and it's individual representations and processes strongly suggests that the model will at least reproduce the qualitative trends of the following phenomena. In the following, we will not cite single studies for each phenomenon, but we rather refer the reader to Kosslyn, Thompson, and Ganis (2006) for an overview of the mentioned studies.

Image generation. Empirical studies suggest, that the construction time of a mental image of a scene or object directly depends on the number of parts and the level of detail. The model offers a trivial and straightforward explanation, as it generates mental images piece by piece. What is more interesting and novel is the proposed sequence in which parts are added and we further discuss this point below.

Image scanning. Several different studies suggest that the time taken to mentally scan from one point of a mental image to another is proportional to the imagined distance between these points. The attention window of our model is shifted gradually over the visual representation to the respective portion of the visual representation that needs to be processed. Therefore again, the model provides a straightforward account of this phenomenon.

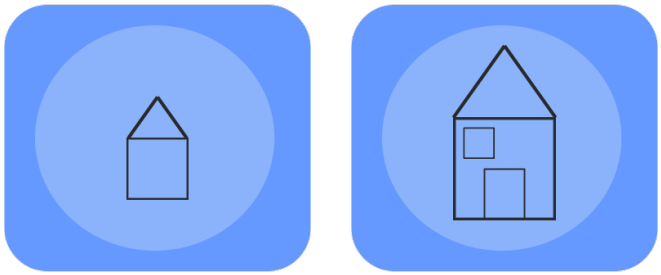


Figure 2: Example of Zooming. Resolution of the attention window is low and therefore only big (size==3) and medium (size==2) sized parts are visualized. Left side: the main shape of the concept *house* is imagined in medium size (size==2). The shape of roof is also visualized as it is of the same relative size (size of *house* plus the relative size of *roof*, i.e., $2-0=2$). *Door* and *window* have a small size (size of *house* plus relative size of *door*, i.e., size==$(2-1=1)$ and are therefore not visible given the current resolution.
Right side: The size of *house* was set to big (size==3) and therefore the size of *door* and *window* is now medium (size==2). Thus, they are now visualized.

Zooming. Zooming in or out of a mental image is realized by altering the size parameter of a concept or a part of the concept in the spatial representation. This parameter is used to determine the extent of the respective shape when it is projected onto the visual representation. Furthermore, if the size parameter is altered for a concept in the spatial representation and it is therefore re-visualized with a now bigger or smaller shape, this has automatic consequences for the visualization of the parts of this concept. The spatial representation stores a concept's parts with their relative size compared to the basic shape of the concept. This relative size again determines whether and when a part is also visualized in the visual representation given the current level of resolution of the attention window (see figure 2 for a visual example). The empirical findings regarding zooming in mental images express that it will take subjects more time to find a part of an imagined object if it is initially imagined at a small size than when it is imagined at a bigger size. These findings can potentially be explained in two ways by the model: 1) subjects employ a zooming process as described above or 2) the resolution of the attention window is increased which will also make some smaller parts of the image "visible".[3] Both accounts would results in increased processing time and thus qualitatively match the empirical results.

Image organization and reinterpretation. There is evidence that mental images have an underlying organization. It has for example been found, that the way presented stimuli are described, e.g., the star of david as either two overlapping triangles or as an hexagon with six small triangles, affects the

[3]For very small parts increasing the resolution will not work and zooming will be necessary.

way subjects later recreate this image mentally. That is, on the one hand, image generation takes longer when the image consists of more parts and on the other hand recognition of patterns that are congruent with the organization of the image is faster than for other valid patterns. A related phenomenon is the difficulty of mental image reinterpretation. That is, it is very difficult for subjects to reinterpret an ambiguous picture as a mental image, if that picture was preciously learned realizing only one of its meanings. Whereas, it is much easier to find the second meaning during normal visual perception of the same ambiguous picture. Both of these types of findings point towards the same direction of mental images being more than just a mental depiction of visual information but including semantics and depending on a more abstract structure and organization underlying the depictive structure. The two-fold structure of our model provides just that. As the spatial layout held in the spatial representation is used to build up the mental image in the visual representation, it is apparent that this consciously available organization affects how the content of the visual representation is inspected as well as interpreted. In order to successfully reinterpret an ambiguous image during mental imagery, the content of the spatial representation would have to be discarded, because even though the content of the visual representation might the be so that it depicts both meanings, the individual parts would need to be linked to different concepts. Furthermore, the retrieval of shape information from long-term memory is context-dependend regarding the currently held concept in the spatial representation. This means that a retrieved shape and its properties are affected by the concept it is linked to in the spatial representation; especially by background knowledge about a concept. For example, the mouth of the rabbit in the famous duck-rabbit image (see figure 3) might not be recalled when subjects are imagining a duck, because this visual feature is irrelevant for the shape of the back of a duck's head.

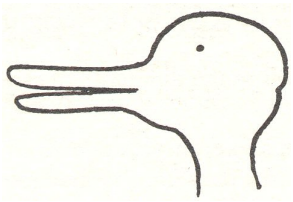


Figure 3: Ambiguous Duck-Rabbit Image

Predictions of the Model There are three main predictions we can draw from the model: 1) internal attention shifts are functional for the construction of complex mental images and are reflected by eye movements, 2) the sequence in which parts are added to a complex mental image is affected by the relative size of the parts, and 3) the visual representation is used only when demanded by the task.

1) Whenever a new part of an image is visualized in the model, the attention window is adjusted in its location and its resolution. That is, it is shifted to the location the new part will be visualized at. There are several studies (e.g., Johansson, Holsanova, & Holmqvist, 2005) that have shown a close correlation between eye movements and the currently processed contents of a visual image. The model implies that eye movements are linked to the shifts of the attention window during mental imagery. Furthermore, these attention shifts are functional to the process of mental image construction.

2) The construction process proposed by the model differs from previous assumptions about the sequence in which parts are added to form a mental image. A common default assumption seems to be that the sequence of parts is determined by the strength of association of the part with the main concept. Furthermore, this is often combined with the idea of choosing that part next, which yields the highest identification value for the concept. This idea stems from an analogy to top-down-hypothesis testing in object recogntion (see Kosslyn, 1994). In contrast, our model predicts a very different sequence for image construction, which is a direct consequence of the implementation of the attention window. The attention window has different scales of resolution, which determine whether a part is visualized and also whether a visualized part is accessible. That is, with the initial low resolution only big parts can be visualized and processed, whereas with a high resolution also smaller parts, i.e., details, are "visible". The model will according to its principle of parsimony not change its resolution, i.e., go into more detail, unless it is necessary. This means, that direct parts of the concept are visualized first when this is possible without a change of resolution, i.e., the ones that are closest in relative size to the concept's main shape.

3) Lastly, the hierarchical structure of the model allows for an on demand usage of the visual representation. That is, if visual information, like the exact shape, is not necessary to fulfill a task, the processing will remain on the level of the spatial representation. This concept fits nicely with the work of Sima, Lindner, Schultheis, and Barkowsky (2010), who found that the same spatial reasoning task is solved by either using mental imagery or by using a more abstract representation, e.g., mental models, depending on whether the instruction demands imagery or not.

Conclusion and further work

We have presented a computational cognitive model of human complex mental image construction and elaborated on the underlying assumptions as well as the predictions derived from the model. The model is able to offer plausible accounts for common mental imagery phenomena and findings about the dual nature of imagery. The model implements an attention window to select regions of the visual representation for processing. The defined role of this structure can be used to predict eye movements during mental imagery tasks and as a novel way of evaluating theories and models of mental im-

agery.

Important aspects whose effects on the model's behavior needs to be investigated include working memory restrictions and similarly decay processes for both employed working memory structures. Furthermore, we are preparing appropriate eye tracking experiments to test the model's predictions about the construction sequence of multi-part images.

Acknowledgments

In this paper work done in the project R1-[ImageSpace] of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition is presented. Funding by the German Research Foundation (DFG) is gratefully acknowledged. We are thankful to the reviewers for their helpful comments.

References

Barkowsky, T. (2007). Modeling mental spatial knowledge processing: An AI perspective. In F. Mast & L. Jäncke (Eds.), *Spatial Processing in Navigation, Imagery, and Perception.* (p. 67-84). Berlin: Springer.

Bertel, S., Barkowsky, T., Engel, D., & Freksa, C. (2006). Computational modeling of reasoning with mental images: basic requirements. In D. Fum, F. del Missier, & A. Stocco (Eds.), *Proceedings of the 7th International Conference on Cognitive Modeling (ICCM 2006)* (p. 50-55). Edizioni Goliardiche; Trieste.

Chandrasekaran, B., Kurup, U., Banerjee, B., Josephson, J. R., & Winkler, R. (2004). An architecture for problem solving with diagrams. In A. Blackwell, K. Marriott, & A. Shimojima (Eds.), *Proceedings of Diagrams 2004* (pp. 151–165). Berlin: Springer.

Courtney, S. M., Ungerleider, L. G., Keil, K., & Haxby, J. V. (1996). Object and Spatial Working Memory Activate Separate Neural Systems in Human Cortex. *Cereb. Cortex*, *6*(1), 39-49.

Farah, M. J., & Hammond, K. M. (1988). Visual and spatial mental imagery: Dissociable systems of representation. *Cognitive Psychology*, *20*, 439–462.

Finke, R. A. (1989). *Principles of mental imagery*. Cambridge, MA: MIT-Press.

Glasgow, J., & Papadias, D. (1992). Computational imagery. *Cognitive Science*, *16*, 355–394.

Gunzelmann, G., & Lyon, D. R. (2007). Mechanisms of human spatial competence. In T. Barkowsky, M. Knauff, G. Ligozat, & D. R. Montello (Eds.), *Spatial Cognition V - Reasoning, Action, Interaction* (p. 288-307). Springer Verlag; 14197 Berlin.

Hegarty, M. (2004). Mechanical reasoning by mental simulation. *Trends in Cognitive Sciences*, *8*(6), 280–285.

Johansson, R., Holsanova, J., & Holmqvist, K. (2005). What do eye movements reveal about mental imagery? Evidence from visual and verbal elicitations. In B. G. Bara, L. Barsalou, & M. M. Bucciarelli (Eds.), *Proceedings of the 27th Annual Conference of the Cognitive Science Society* (p. 1045 - 1059). Mahwah, NJ: Erlbaum.

Kosslyn, S. M. (1980). *Image and mind*. Cambridge, MA: Harvard University Press.

Kosslyn, S. M. (1994). *Image and brain: The resolution of the imagery debate*. Cambridge, MA: The MIT Press.

Kosslyn, S. M., Thompson, W. L., & Ganis, G. (2006). *The case for mental imagery*. New York: Oxford University Press.

Lathrop, S. (2008). *Extending cognitive architectures with spatial and visual imagery mechanisms*. Ph.d. thesis, University of Michigan.

Levine, D. N., Warach, J., & Farah, M. (1985). Two visual systems in mental imagery: Dissociation of "what" and "where" in imagery disorders due to bilateral posterior cerebral lesions. *Neurology*, *35*(7), 1010-.

Nestor, A., & Kokinov, B. (2004). Towards active vision in the DUAL cognitive architecture. *International Journal on Information Theories and Applications*, *11*, 9–15.

Schultheis, H., Barkowsky, T., & Bertel, S. (2006). LTM-C — An improved long-term memory for cognitive architectures. In D. Fum, F. del Missier, & A. Stocco (Eds.), *Proceedings of the 7th International Conference on Cognitive Modeling (ICCM 2006)* (p. 274 - 279). Edizioni Goliardiche; Trieste.

Sima, J. F., Lindner, M., Schultheis, H., & Barkowsky, T. (2010). Eye movements reflect reasoning with mental images but not mental models in orientation knowledge tasks. In C. Hölscher (Ed.), *Spatial Cognition VII* (pp. 248–261). Heidelberg: Springer Verlag.

Ungerleider, L., & Mishkin, M. (1982). Two cortical systems. *Analysis of Visual Behavior*, 549-586.

Toward an analog neural substrate for production systems

Patrick Simen (psimen@princeton.edu), Marieke Van Vugt (mkvan@princeton.edu)
Fuat Balci (fbalci@princeton.edu)
Princeton Neuroscience Institute, Princeton University, Princeton, NJ 08544 USA

David Freestone (David_Freestone@brown.edu)
Department of Psychology, Brown University, Providence, RI 02912 USA

Thad Polk (tpolk@umich.edu)
Department of Psychology, University of Michigan, Ann Arbor, MI 48432 USA

Abstract

Symbolic, rule-based systems seem essential for modeling high-level cognition. Subsymbolic dynamical systems, in contrast, seem essential for modeling low-level perception and action, and can be mapped more easily onto the brain. Here we review existing work showing that critical features of symbolic production systems can be implemented in a subsymbolic, dynamical systems substrate, and that optimal tuning of connections between that substrate's analog circuit elements accounts for fundamental laws of behavior in psychology. We then show that emergent properties of these elements are reflected in behavioral and electrophysiological data, lending support to a theory about the physical substructure of productions. The theory states that: 1) productions are defined by connection strengths between circuit elements; 2) conflict resolution among competing productions is equivalent to optimal hypothesis testing; 3) sequential process timing is parallel and distributed; 4) memory allocation and representational binding are controlled by competing relaxation oscillators.

Keywords: Production system; neural network; diffusion model; random walk; reinforcement learning.

A subatomic structure for productions

Production systems underlie the most successful theories of high-level cognition, exemplified by such capabilities as planning, problem-solving, reasoning and language. Productions — if-then rules that test the contents of a working memory and trigger actions or changes to working memory as a result — have accordingly been characterized as the 'atomic components of thought' (Anderson & Lebiere, 1998). The implication is that the complex chemistry of mental life arises from, and can more easily be understood in terms of, the interactions of these simple atoms. To make the most of this analogy, however, requires a biologically plausible theory about the subatomic structure that defines these interactions. Here we propose a subatomic theory in which productions arise from the behavior of 'elementary particles' — leaky integrators, or classic neural network units — whose interactions with each other are defined by connection strengths and structured network topologies.

Any computational theory of cognition faces several challenges: How well does it conform to known laws of behavior and classic patterns of brain activity? How well does data conform to new predictions entailed by it? And how much functionality does it give you (e.g., is it computationally complete)? Here we progressively build up a design for a neural network structure that emulates the most important features of production systems. We start with a critical core for individual productions, and then add on control mechanisms that adapt the core's behavior in order to maximize a reward function. We will attempt to show how each addition accounts for known laws, entails new (in some cases, successfully tested) predictions, and moves the resulting architecture toward full, production-system functionality. The result falls short of enabling the automatic translation of arbitrary production system programs into equivalent neural networks, but it suggests that such translations will be possible for a constrained set of such programs (and that the constraints thus identified may be of theoretical importance).

For the core, we review a specific, structured neural circuit with heuristically reward-maximizing connections that has previously been proposed as an implementation of productions (Polk, Simen, Lewis, & Freedman, 2002; Simen & Polk, in press). After outlining the remaining mechanisms underlying key features of a neural production system architecture, we review separately published results showing the conformance of its behavioral predictions to the matching law of operant conditioning, to the logistic/softmax choice function used in reinforcement learning, and to recent, tested theories of optimal perceptual decision making. We also review new evidence supporting its predictions regarding the lateralized readiness potential (LRP) that is observed in human electroencephalography (EEG).

To the core production implementation, we add a simple timing mechanism (allowing controlled sequential processing), and we outline a proof that it conforms to the law of scalar invariance in interval timing (Gibbon, 1977). We show that this mechanism predicts behavior observed in the differential reinforcement of low rates of responding (DRL) task.

We conclude the addition of mechanisms by outlining a potential solution to two major challenges facing a connectionist production system architecture: one is the need for a flexible memory management system; the other is the variable binding problem. This problem afflicts any system in which the semantics of a representation depend only on what is connected to what, so that the components of different representations must be shared. Our proposed solution involves relaxation oscillators with tunable frequencies and duty cycles. These enable the recruitment of memory resources through fast Hebbian learning by tagging and reserving allocated net-

work units. The result is the sort of oscillatory activity that is invariably observed in invasive electrode recording and scalp EEG.

The elementary particles

The basic building block we will use is a stochastic neural network unit. We begin its description by considering it as a deterministic system. At each moment, it computes a weighted sum of its current inputs, then computes an exponentially decaying average of recent weighted sums, and finally amplifies the result by a gain function that is approximately linear (but which saturates at very low and very high input levels). This quantity is broadcast to other units, over connections whose strengths determine their relative contribution in those units' weighted sum computations. Formally, the output of the ith unit is V_i, the leaky integral of summed input is x_i, and the dynamics are defined as follows:

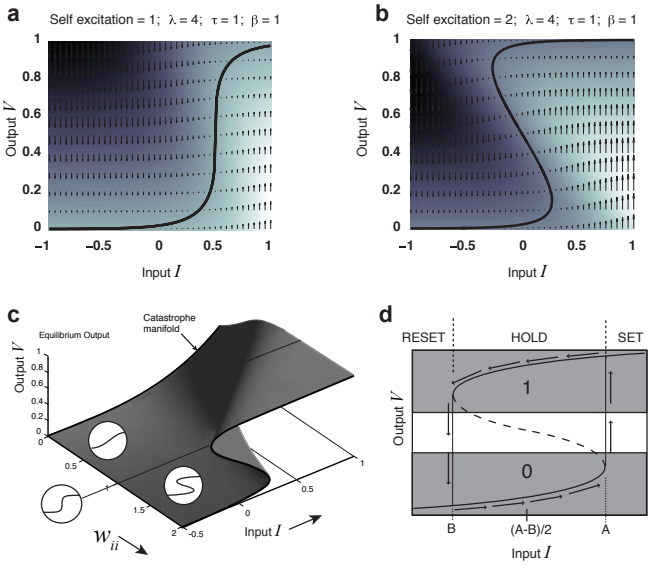


Figure 1: **a, b:** A neural network unit's rate of activation change (dV/dt) as a function of input I and output V for units with fixed I and balanced (**a**) or strong (**b**) excitatory, recurrent connections. Equilibrium curves are solid; velocities dV/dt are indicated by arrows and shading (light > 0, dark < 0). **c:** 'Catastrophe manifold' formed by the equilibrium curves of Eq. 3 as the self-excitatory, recurrent weight strength w_{ii} ranges from 0 to 2. Three network symbols are also illustrated. Sigmoid: weak self-excitation, leaky integration ($w_{ii} < 1$ for a unit with $\lambda = 4$). Rounded step-function: balanced self-excitation, perfect integration ($w_{ii} = 1$). S symbol: strong self-excitation, hysteresis and bistable switching (or 'latch') behavior ($w_{ii} > 1$). **d:** A latch based on hysteresis. States above the dashed curve converge to the upper solid curve; states below converge to the lower solid curve. This latch can store a 1 (upper gray region) or a 0 (lower gray region) as long as input is held between A and B. Bit-flipping during constant I is least likely when $I = (A+B)/2$.

$$I_i = \sum_{j=1}^{n} w_{ij} \cdot V_j, \tag{1}$$

$$\tau \cdot \frac{dx_i}{dt} = -x_i + I_i, \tag{2}$$

$$\text{and } V_i(t) = f(x_i(t)) = [1 + \exp(-\lambda \cdot (x_i - \beta))]^{-1}. \tag{3}$$

Parameters λ and β determine the steepness and position of the sigmoidal activation function f, and τ determines the decay rate of exponential averaging (large τ gives slow decay).

In addition to deterministic dynamics, we assume that noise enters the system from units that have direct sensory inputs, and also from the connections between units themselves. To model these assumptions, we use stochastic differential equations, in which we represent white noise with a useful abuse of notation as $\eta \equiv dW/dt$ (multiplication by dt then gives the standard notation dW in our equations; cf. Gardiner, 2004). This quantity represents the time-derivative of a Brownian motion, or Wiener process, $W(t)$.[1] The standard deviation of η is 1, but can be changed to any value c by multiplying by c. Here, we multiply η by the square root of the weighted input, an assumption which is consistent with an even more microscopic level of neural modeling: we assume that spiking neurons are Poisson processes, and that leaky integrators model their population-level behavior. The variance of sums of these independent processes is the sum of their variances. Thus, if we consider increases in a given weight w_{ij} to be equivalent to the addition of independent Poisson processes (because of the addition of noisy synaptic connections), we get a noise standard deviation equal to the square root of net input. Formally, then, the full, stochastic unit description is as follows:

$$\begin{aligned} \tau \cdot \frac{dx_i}{dt} &= -x_i + \sum_{j=1}^{n} \left(I_i + c_{ij}\sqrt{I_i} \cdot \eta \right) \\ \Rightarrow \tau \cdot dx_i &= \left(-x_i + \sum_{j=1}^{n} I_i \right) dt + c_{ij} \sum_{j=1}^{n} \sqrt{I_i}\, dW_{ij} \\ \Rightarrow \tau \cdot dV_i &\approx (-x_i + f(I_i))\, dt + c_{ij} \sum_{j=1}^{n} \sqrt{I_i}\, dW_{ij} \end{aligned} \tag{4}$$

(See Simen and Polk (in press) for justification of the last approximation, which moves the noise term outside the nonlinear function f.)

This system can be numerically simulated on a computer (and perhaps be more easily understood) as a discrete-time difference equation (Gardiner, 2004):

$$\tau \cdot V_i(t + \Delta t) \approx V_i(t) + (-x_i + f(I_i))\, \Delta t + c_{ij}\sqrt{\Delta t} \sum_{j=1}^{n} \sqrt{I_i}. \tag{5}$$

It is now critical for our purposes to consider the effects of recurrent excitation of a unit by itself ($w_{ii} > 0$). The strength

[1] W in fact is non-differentiable, but it is the limit of a sequence of slightly smoother, differentiable noise processes, so it can be used without danger.

of this self-excitation determines which of three, qualitatively distinct types of behavior a unit exhibits (Simen & Polk, in press). For $w_{ii} < 1$, the system acts like a leaky integrator; as w_{ii} grows, the leak is reduced. When the self-excitation exactly balances the leak ($w_{ii} = 1$), the unit acts like a perfect integrator (until it saturates). For $w_{ii} > 1$, the system is unstable and is forced upward against the upper ceiling on its activation (1), or downward toward its lower floor (0); thus it acts like a binary switch. Furthermore, such a unit displays hysteresis, so that it can both trigger abrupt changes and also store a bit. Fig. 1 shows the dynamics of such a unit.

In general, leaky integration (weak self-excitation) is useful because it low-pass filters its input, thereby removing much of the high frequency noise contributed by connections and by the environment. Perfect integration (balanced self-excitation) is needed for optimal hypothesis testing (Bogacz, Brown, Moehlis, Holmes, & Cohen, 2006). Bistability (strong self-excitation) is needed for triggering subsequent steps of sequential processes and for maintaining the current state of working memory. The behavioral and electrophysiological data we consider bears on the predictions made by these bistable units and the integrators that feed into them.

Neural productions, timers and oscillators

Fig. 2 shows the basic building blocks of the proposed architecture; in the remainder of the paper, we explain how each block functions, and assess how well each accords with known laws and new empirical data. The left column shows the 3 unit types (**a**,**b**,**c**). Simen and Polk (in press) detail how a complete set of logic operations (AND, OR, NOT) can be built from the bistable units in **c** by parameterizing their input strengths. Panel **d** shows a simple if-then rule structure: the leaky integrator filters noise from its inputs, and if the sum exceeds a critical level, the bistable unit switches from (approximately) 0 to (approximately) 1. This is analogous to the process of 'matching' the contents of working memory (which can be made to depend on arbitrarily many symbolic preconditions using a cascade of logic gates). The degree of match may be an analog quantity, and whether this is sufficient to cause a bit flip in the output unit determines whether the production will 'fire'. Furthermore, the weights on inputs to the if-stage may also encode preferences between productions that have an equal degree of supporting evidence.

If more than one production matches, however, there may be conflict between them. At least at the motor output stage (e.g., SOAR's 'operators'), such conflict must be resolved. Here we consider conflict resolution as a process of competition between matching productions (Fig. 2 **e**), with the outcome biased toward selection of the production with the strongest amount of preference-weighted evidence. Since noise is everywhere, this reduces to a well-defined hypothesis testing problem, for which simple, near-optimal algorithms exist. These algorithms — sequential probability ratio tests (SPRTs) — can be parameterized to maximize expected utility in the case of two-alternative choices (Bogacz et al., 2006),

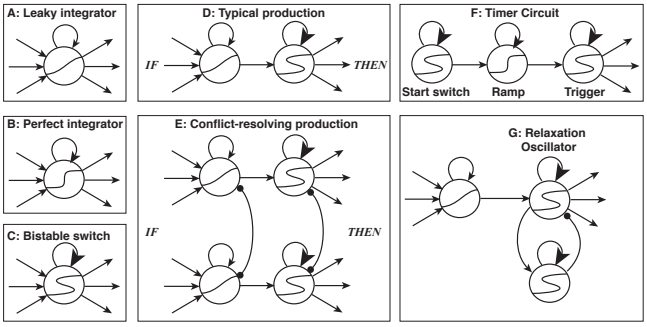


Figure 2: Basic building blocks. Arrowheads indicate excitation, circleheads inhibition. **a, b, c:** Elementary particles; arrows: excitatory inputs. **d:** Production topology. **e:** Conflict resolution via lateral inhibition (circles: inhibition); inhibition between switches is optional. **f:** Interval timer. **g:** Relaxation oscillator added to production output unit.

and can approximately maximize utility for a greater number of competing alternatives (McMillen & Holmes, 2006). For a difficult decision, the process of deciding via lateral inhibition (a form of attractor dynamics) can be parameterized to implement an SPRT. This requires only that the lateral inhibitory strengths between input units equal -1. An example of these dynamics is shown in Fig. 3. Thus, the firing of a single production is equivalent to a statistical hypothesis test.

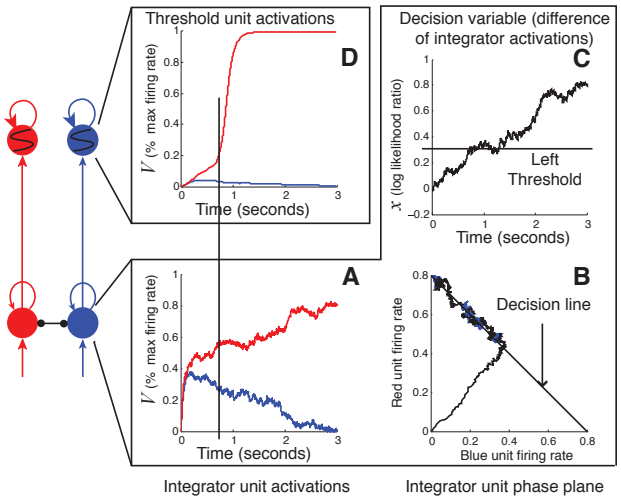


Figure 3: Hypothesis testing via lateral inhibition. The 2D system in the bottom layer reduces to a single dimension, along which a random walk to threshold occurs (implemented by attractor dynamics in the top layer).

A critical question facing the proposed architecture, however, is whether the timing of these firings can be coordinated and sequentialized without reference to a central system clock. Our problem is the same as that facing digital

circuit designers, who have long relied on a central clock and synchronous updating to preclude critical race conditions and other signal timing hazards. Our solution is to use these production implementations to form processing bottlenecks, and to use handshake completion signals between computing elements for asynchronous, distributed timing control (Simen & Polk, in press). The most difficult question is whether we can implement productions of the form: If A, Then B and Not A. Naively wiring up a system to implement such a production can cause critical race conditions or metastability.

Our solution derives from the hysteresis properties of our bistable units. Fig. 4 shows that a sequence of such units can be wired up so that an input unit stays active long enough to trigger an output unit, which in turn inhibits the input. If the input unit did not resist this inhibition, it could fail to latch the output before shutting off. Elsewhere we have detailed the specific conditions that ensure proper sequential latching. To ensure that timing issues can be handled, we use the timer circuit in Fig. 2 **f** to implement an analogue of the delay gates used in digital logic. This mechanism activates a 'start' switch unit on the left, then integrates that signal in a 'ramp' unit, weighted by the start-to-ramp weight, until it triggers the 'trigger' unit to flip from 0 to 1. The delay duration is equal to this threshold value divided by the start-to-ramp weight. These dynamics are very similar to those implementing hypothesis-testing in Fig. 3, but now the only evidence is the passing of time.

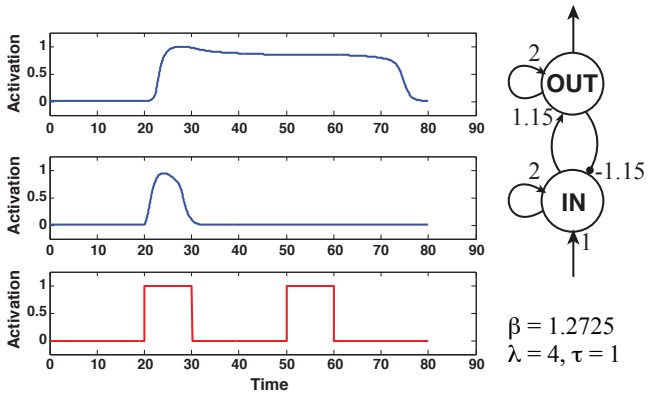


Figure 4: A production that negates its own if-condition. Bottom layer: input signal (red). Middle layer: IN unit activation. Top layer: OUT unit activation.

With these building blocks in hand, we can build arbitrarily complex circuits that implement logic gates and finite state machines, and thus special-purpose production systems. However, we still face the same critical problems facing all connectionist systems: if the semantics of a representation depend on what is connected to what, then how do separate representations share subcomponents? Or if their subcomponents conflict, then how are the proper subcomponents bound with the proper parent representation? Temporal synchrony has been widely considered to be a potential solution. The architectural assumptions are made that whatever is simultaneously active refers to the same entity, and distinct entities share different oscillation phases. We implement these assumptions using the same machinery that underlies productions which cancel their own if-conditions.

Fig. 2 **g** shows that for each production trigger, we can assign an inhibitor. If a production fires, its output unit activates and triggers its own cancellation after a controllable delay (depending on connection strengths). However, the firing of a production can trigger a stored, hidden variable in a third bistable unit, which forces reactivation of the production after the inhibitor falls silent. This process repeats, triggering oscillations. When productions compete with each other, they push their active periods out of phase with each other, as shown in Fig. 5. When they do not, excitation causes them to entrain to the same phase. Thus conflicting representations locally decide which gets to broadcast information globally. If we allow for a plasticity signal that globally increases the learning rate of Hebbian connection plasticity between units, and if we activate this signal only at critical times, then we can burn in connections (possibly temporary connections) between units simply by activating them.

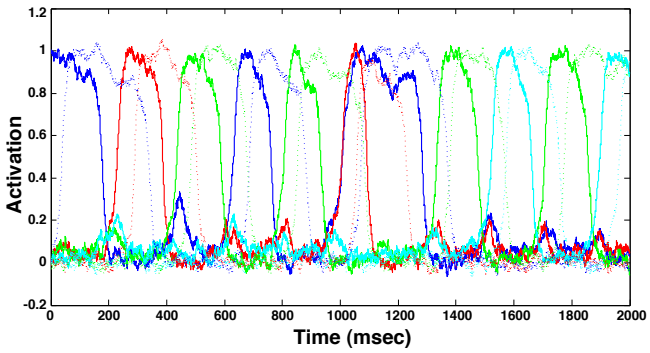


Figure 5: Relaxation oscillations among competing representations, allowing sharing of a single broadcast channel. Each solid color corresponds to one representation's bistable output unit; dashed curves correspond to the output's inhibitor.

More work will be needed to determine the scope of this approach to dynamic symbol and rule creation, to the implementation of a data type system such as exists in ACT-R, and to the binding problem more generally. With enough additional assumptions about the structure of the basic building blocks, it would be possible to translate between any given symbolic architecture and an architecture built from the components we have outlined. This must be the case in a trivial sense because our components are equivalent to circuits of resistors, capacitors and transistors. To stand as a psychologically plausible mapping, however, any subsymbolic theory will have to account for empirical data. We now focus on the kind of data for which our subsymbolic approach has something definite to say, leaving a more detailed investigation of the dynamics of connection-strength change for future work.

Laws of behavior

Two laws of behavior bear directly on the plausibility of the architectural building blocks. The first, known as the 'matching law', states the following: that the ratio of rates of two (or more) different types of behavior that an animal engages in equals the ratio of the rewards earned for those behaviors. We showed in Simen and Cohen (2009) that the network in Fig. 3 reproduces this behavior. That network involves conflict between two productions that are supported by exactly the same amount of perceptual evidence. The exponentially weighted reward history of each response is encoded in the weight between the input unit and output unit of a production. This effectively changes the random walk thresholds for each response, while the walk itself is unbiased toward any response. The average result, in the case of two alternatives, is a state of exact matching of the behavior and reward ratios. When, instead, the reward history is encoded in connections from sensory inputs to the laterally inhibiting units, and input-output unit weights are held fixed, the model implements a softmax or logistic choice function defined on the difference between the reward histories. Evidence abounds for one or the other choice function in the instrumental conditioning literature since the time of Skinner. Thus the basic implementation of preferences for certain responses over others in the architecture meets a well-known psychological constraint on learning from reinforcement.

The other law regards timed behavior. A variety of different timing experiments show that the standard deviation of response times in such tasks is equal to a constant times the mean. The distribution of such responses is usually approximately Gaussian. The timing model in Fig. 2 **f**, accounts for this law. When a unit balances its self-excitation against its leak, it acts as an integrator. The model uses a simple error-correction rule to set the connection strength w from the start switch unit so as to ramp up to a level sufficient to trigger a switch from 0 to 1 in the output trigger. The integrator acts as a drift-diffusion process, since it integrates a constant drift term, w, corrupted by noise of amplitude $c\sqrt{w}$:

$$dV = w \cdot dt + c \cdot \sqrt{w} \cdot dW. \quad (6)$$

The trigger unit at the end of the chain defines a threshold on this diffusion process; call it z. Such a process produces a Wald, or inverse Gaussian, distribution of first-passage times (Luce, 1986). The mean RT of this process is z/w, and the standard deviation σ is $c\sqrt{z}/w$. Given that the ramping integrator unit cannot rise above a certain activation because of its saturation nonlinearity, then if we wish to minimize RT variability, we have the choice of minimizing z or maximizing w. The square root in the numerator indicates that increasing w will effect a larger reduction in variability than an equal increase in z. This implies that for all intervals, we should set z to a constant value that is as large as possible, without requiring the integrator to enter its highly nonlinear activation range. This in turn implies that $\sigma = \gamma z/w$, with $\gamma = c/\sqrt{z}$. That is, RT standard deviation is in constant proportion to the mean. Furthermore, as long as c is not too great — with a psychologically plausible value of 0.1 to 0.2, for example – the Wald distribution has very little skewness, and looks almost normal (and a slight positive skewness is often observed in timing data anyway). Thus the model reproduces scalar invariance, and meets a second strong, empirical constraint.

Other behavioral and EEG predictions

We now examine two new predictions that regard the specific mechanism used to implement thresholds. In most decision making models (e.g., Bogacz et al., 2006), such thresholds are simply assumed to exist as a step function or Heaviside function, with a sharp discontinuity at the threshold. The bistable trigger mechanism described in Fig. 1 makes no such assumption, but nevertheless acts approximately as an all-or-none, digital device. Its hysteresis properties are critical for sequential processing, as we have shown, but does it make any testable predictions?

One is that if an input to a trigger unit with strong self-excitation is just below the point needed to trigger a transition from low to high activation, there will nevertheless be occasional triggerings due only to noise. This phenomenon — known as the escape from a double-well potential (Gardiner, 2004) — produces escape-time distributions defined in terms of exponential functions of the well depth (in our case, the remaining distance to the threshold).

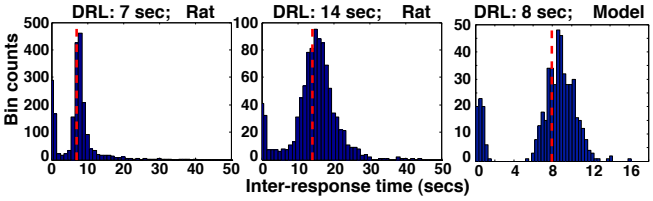


Figure 6: Two-component (exponential + scale-invariant Gaussian) response time distributions (rat, **a, b**, model, **c**).

In fact, a nearly exponential component of response times shows up in rat data collected in the differential reinforcement of low rates of responding (DRL) task. In this task, an animal must wait some minimum amount of time before making a response. Any response after this time is rewarded; any response that occurs prior to this waiting time relative to their last response resets the clock. Animals learn to wait in this task until shortly after the deadline, but they also emit a proportion of very fast responses that are apparently not controlled by a timer. Our model of this task involves using the timer circuit in Fig. 2**f** to implement the nearly Gaussian component of such RT distributions, but it also allows for direct connections between the start-switch and response trigger. This produces a proportion of fast responses that are nearly exponentially distributed. We reason that such a connection exists because of the way these tasks are acquired by animals: first, a contingency between some input stimulus and the response mechanism must be learned; second, a

learned delay between responses is shaped through training.

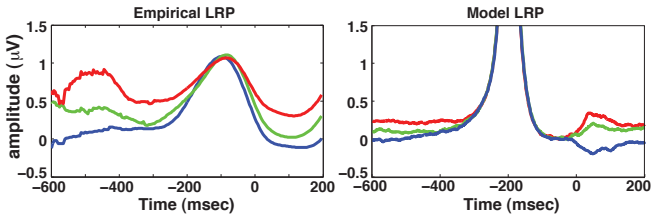


Figure 7: **a:** Average, response-locked LRP data (microvolts) from 8 human participants performing left vs. right dot-motion discrimination, with stimulus odds equal to 60:40 (blue), 75:25 (green), 90:10 (red). Behavioral responses occur at time 0. Data is baseline-corrected to align peaks. **b:** Model LRP (left threshold unit activation subtracted from right in Fig. 3, followed by a bilateral shutoff signal), with constant bias toward the right. The order of LRP differences between conditions is captured, but (as shown by the y-axis limits) capturing the smaller magnitude of the empirical peak requires additional assumptions.

The second prediction the bistable trigger mechanism makes regards the lateralized readiness potential (LRP) observed in any task with a motor response that occurs on one side of the body. Prior to the movement, a voltage builds up over the part of motor cortex that is contralateral to the movement. A voltage also builds up on the same side as the movement, but not to the same degree. Then, just before the response is made, the LRP returns to baseline, because the voltage on both sides of the head over motor cortex becomes large and equal. We hypothesize that motor cortex houses response triggers, and we examined what would happen to a circuit in which a prior probabilities favored, say, a left button press rather than right button press. Although our bistable switches are nearly binary, they do involve slow, graded changes in activation level prior to the point at which they transition to a high activation. Because of this, and because this happens to a greater extent for the response trigger that is about to activate than for a trigger for the other, competing response, a difference in trigger activations develops, as shown in Fig. 7 **a**. We interpret this difference as a readiness potential. As a result, a consistent bias toward one response over the other should show up as an LRP both before and after the response. Such biases are expected in two-alternative perceptual decision making tasks with rewards for correct responses in which one stimulus is more frequently presented than the other (Bogacz et al., 2006). Simen et al. (2009) showed that human behavior in such tasks is consistent with the predicted bias toward the more frequent stimulus.

New LRP data from the same task shows the predicted physiological signature of such a constant bias: for a condition in which a right button-press response is always more likely to be correct than a left button-press, a persistent LRP should occur, with magnitude increasing as the prior probability increases for a given response. Data from 8 participants confirms this stimulus bias prediction (Fig. 7).

Conclusion

Here we have shown strong behavioral and electrophysiological evidence for key components of a neurally implemented production system architecture. These include bistable response units, and competitive response selection and hypothesis testing that are equivalent to random walk attractor dynamics. Thus the assumptions we made in order to achieve basic production system functionality seem to be justified. Much work remains to determine just how much production system functionality can truly be emulated by such systems. However, it is clear from working examples that simple cognitive models of problem solving can be so implemented (Polk et al., 2002; Simen & Polk, in press), and we have outlined a mechanistic implementation of the temporal synchrony solution to the binding problem and the problem of dynamic linking among representations — problems which bedevil neural network architectures, but which are handled easily in standard production systems. We hope that future work on this topic will illustrate what constraints need to be imposed on production system programs in order for them to be 'compiled' into an equivalent neural network.

References

Anderson, J., & Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence-Erlbaum Associates.

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., & Cohen, J. D. (2006). The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced choice tasks. *Psychological Review*, *113*(4), 700–765.

Gardiner, C. W. (2004). *Handbook of Stochastic Methods* (Third ed.). New York, NY: Springer-Verlag.

Gibbon, J. (1977). Scalar expectancy theory and Weber's law in animal timing. *Psychological Review*, *84*, 279–325.

Luce, R. D. (1986). *Response Times: Their Role in Inferring Elementary Mental Organization*. New York: Oxford University Press.

McMillen, T., & Holmes, P. (2006). The dynamics of choice among multiple alternatives. *Journal of Mathematical Psychology*, *50*, 30–57.

Polk, T. A., Simen, P. A., Lewis, R. L., & Freedman, E. G. (2002). A computational approach to control in complex cognition. *Cognitive Brain Research*, *15*(1), 71-83.

Simen, P. A., & Cohen, J. D. (2009). Explicit melioration by a neural diffusion model. *Brain Research*, *1299*, 95–117.

Simen, P. A., Contreras, D., Buck, C., Hu, P., Holmes, P., & Cohen, J. D. (2009). Reward rate optimization in two-alternative decision making: empirical tests of theoretical predictions. *Journal of Experimental Psychology: Human Perception and Performance*, *35*, 1865–1897.

Simen, P. A., & Polk, T. A. (in press). A symbolic/subsymbolic interface protocol for cognitive modeling. *Logic Journal of the IGPL*.

Deriving Behavior from Personality: A Reinforcement Learning Approach

Christopher Simpkins (chris.simpkins@gatech.edu)
Georgia Tech Research Institute, 250 14th Street, NW
Atlanta, GA 30318 USA

Charles L. Isbell, Jr. (isbell@cc.gatech.edu)
College of Computing, 801 Atlantic Drive
Atlanta, GA 80332 USA

Nicholas Marquez (nicholas.marquez@gatech.edu)
College of Computing, 801 Atlantic Drive
Atlanta, GA 80332 USA

Abstract

Creating artificial intelligent agents that are high-fidelity simulations of natural agents will require that behavioral scientists be able to write code themselves, not merely act as consultants with the ensuing knowledge acquisition bottleneck. We are designing a system that will make it possible to create rich agents using concepts familiar to behavioral scientists, such as personality models from psychology. However, translating personality models into the concrete behavior of an agent using currently available programming constructs would require a level of code complexity that would make the system inaccessible to behavioral scientists. What we need is a way to derive the concrete actions of an agent directly from psychological personality models. This paper describes a reinforcement learning approach to solving this problem in which we represent trait-theoretic personality models as reinforcement learning agents. We validate our approach by creating a virtual reconstruction of a psychology experiment using human subjects and showing that our virtual agents exhibit similar behavior patterns.

Keywords: Agents; Reinforcement Learning; Personality

Introduction

There is tremendous interest in creating synthetic agents that behave as closely as possible to natural (human) agents. Rich, interactive intelligent agents will advance the state of the art in training simulations, interactive games and narratives, and social science simulations. However, the programming systems for creating such rich synthetic agents are too complex, or rather too steeped in computational concepts, to be used directly by the behavioral scientists who are most knowledgeable in modeling natural agents. Engaging behavioral scientists more directly in the authoring of synthetic agents would go a long way towards improving the fidelity of synthetic agents.

Our goal is to create a programming language that a behavioral scientist can use to write agent programs using concepts familiar to behavioral scientists. This task is complicated by the fact that the most popular and best understood personality models from behavioral science do not lend themselves to direct translation into computer programs. Requiring a behavioral scientist to specify behaviors in the detail required in even the most cutting edge purpose-built programming language would plunge the would-be behavioral scientist agent programmer right into a morass of complex computational concepts that lie outside the expertise of most dedicated behavioral experts. To solve this problem we need a way to get from personality models to behaviors, to derive specific agent actions in an environment from a personality model without having to program the derivation in great detail.

In this paper, we describe a way to model motivational factors from trait-oriented personality theory by reinforcement learning components. We describe a virtual agent simulation that reconstructs a human subject experiment from psychology, namely some of Atkinson's original work in achievement motivation and test anxiety, and show that our simulation exhibits the same general behavior patterns as the human subjects in Atkinson's experiments. First, we briefly discuss relevant personality research and provide some background.

Personality

Personality is a branch of psychology that studies and characterizes the underlying commonalities and differences in human behavior. Within psychology, there are two broad categories of personality theories: processing theories, and dispositional, or trait theories. Social-cognitive and information-processing theories identify processes of encodings, expectancies, and goals in an attempt to characterize the mechanisms by which people process their perceptions, store conceptualizations, and how those processes drive their interactions with others (Dweck & Leggett, 1988; Cervone & Pervin, 2009; Cervone & Shoda, 1999). A strength of processing theories, especially from a computational perspective, is that they provide a detailed account of the cognitive processes that give rise to personality and drive behavior. This strength is also a drawback – processing theories tend to be detailed and often low-level (though not as low-level as cognitive architectures, which we will discuss below), and this makes them less intuitive and less suited to describing personality in broad, easily understood terms.

Trait theories (Cervone & Pervin, 2009), the most well-known example of which is the Five-Factor model (McCrae & Paul T. Costa, 2008), attempt to identify stable traits (sometimes called "trait adjectives") that can be measured on numerical scales and remain invariant across situations in determining behavior. A strength of the trait approach is that

they are well-suited to describing individuals in broad, intuitive terms. Two drawbacks of the approach are that there is not yet widespread agreement on a set of truly universal traits (or how many there are), and it is not clear how trait models drive behavior. A promising line of research by Elliot and Thrash (Elliot & Thrash, 2002) is working towards solving these problems by integrating motivation into personality in a general way. The work of Elliot and Thrash particularly supports the approach we present here, as they show that approach and avoidance motivation underpins all currently popular trait theories.

While debate continues about the merits and drawbacks of the different approaches to personality, the psychology community is also attempting to unify personality and motivation theory (Mischel & Shoda, 2008). While the work we present here is focused on bridging the gap between the descriptive power of trait-oriented models and the behavior that arise from them, we consider this work to be complementary to work in encoding information processing theories. In the future, rich computational agents may be built by combining approaches.

Reinforcement Learning

One can think of reinforcement learning (RL) as a machine learning approach to planning, that is, a way of finding a sequence of actions that achieves a goal. In RL, problems of decision-making by agents interacting with uncertain environments are usually modeled as Markov decision processes (MDPs). In the MDP framework, at each time step the agent senses the state of the environment and executes an action from the set of actions available to it in that state. The agent's action (and perhaps other uncontrolled external events) cause a stochastic change in the state of the environment. The agent receives a (possibly zero) scalar reward from the environment. The agent's goal is to find a *policy*; that is, to choose actions so as to maximize the expected sum of rewards over some time horizon. An optimal policy is a mapping from states to actions that maximizes the long-term expected reward. In short, a policy defines which action an agent should take in a given state to maximize its chances of reaching a goal.

Reinforcement learning is a large and active area of research, but the preceding is all the reader needs to understand the work presented here. More detail can be found in (Sutton & Barto, 1998; Kaelbling, Littman, & Moore, 1996).

Modeling Personality with Reinforcement Learning

The essential idea behind modeling personality traits with reinforcement learning is that each motivational factor can be represented by a reinforcement learning component. In psychology, the inherent desirability or attractiveness of a behavior or situation is referred to as *valence*. For a person high in success approach motivation, behaviors or situations that provide an "opportunity to excel" will have high valence, while other behaviors will have lower valence. The notion of valence translates fairly directly into the concept of reward in reinforcement learning. Just as people with certain motivational factors will be attracted to high-valence behaviors, a reinforcement learner is attracted to high-reward behaviors. This is the basis for modeling motivational factors with reinforcement learning components. By encoding the valence of certain behaviors as a reward structure, reinforcement learners can learn the behavioral patterns that are associated with particular motivational factors. This is a powerful idea, because it allows an agent author to write agent code using motivational factors while minimizing the need to encode the complex mechanisms by which such factors lead to concrete behavior.

A critical aspect of trait theory is that traits can have interactive effects. It is clear that a person who is high in achievement motivation will "go for it" when given the opportunity and that a person who is high in avoidance motivation will be more reserved. But what happens when a person is high in both motivations? Such interactive effects cannot be ignored in a credible treatment of personality, but it is hard to predict the behavioral patterns that will arise from given combinations of motivational factors. One can imagine the code complexity that might result from trying to model such interactive effects with production rules or other traditional programming constructs. As we demonstrate later, our reinforcement learning approach handles such interactive effects automatically.

It is important to note that we are not creating a new theory of personality. We are creating a computational means of translating existing theories of personality from *psychology* (not computer science) into actions executed by synthetic agents. We are also not committing to a particular theory from psychology, but rather supporting the general category of trait theories of personality which, until now, have not been directly realizable in computer agents.

In the remainder of this paper we discuss some related work in agent modeling, present our virtual reconstruction of a human subject experiment using our reinforcement learning approach, and discuss the promising results and their implications for future work.

Related Work

There is a great deal of work in modeling all sorts of phenomena in synthetic agents. Cognitive architectures provide computational models of many low-level cognitive processes, such as memory, perception, and conceptualization (Jones, 2005; Langley, Laird, & Rogers, 2008). Cognitive architectures support scientific research in cognitive psychology by providing runnable models of cognitive processes, support research in human-computer interaction with detailed user models (John, 1998), and can serve as the "brains" of agents in a variety of contexts. The most notable and actively developed cognitive architectures are Soar (Laird, 2008) and ACT-R (Anderson, Bothell, & Byrne, 2004). Recently, some effort has gone into integrating reinforcement learning into Soar (Nason & Laird, 2008). While RL is used to improve

the reasoning system in Soar, we are using RL to support new paradigms of computer programming for agent systems. In general, our work differs from and complements work in cognitive architectures in that we are drawing on psychological theory that is expressed at a much higher level of abstraction. Cognitive psychology and AI have often built on each other. Indeed, cognitive psychology is the basis of cognitive architectures in AI. Our work is an attempt to bring in mainstream personality psychology as a basis for building intelligent agents, which we hope will complement the detailed models of cognitive architectures in creating rich synthetic agents.

There is a large and rich body of work in believable agents. Mateas and Stern built on the work of the Oz project (Loyall & Bates, 1991) in creating a programming language and reactive–planning architecture for rich believable agents. They implemented their theory in the computer game Facade, a one-act interactive drama in which the player interacts with computer simulated characters that provide rich social interactivity (Mateas & Stern, 2004). Gratch, Marsella and colleagues have a large body of work in creating rich simulations of humans for training simulations that incorporate models of appraisal theory and emotion (Gratch & Marsella, 2005; Swartout et al., 2006). A distinctive feature of the work of both Mateas, et. al., and Gratch, et. al., is that they are dealing with the entire range of AI problems in creating believable agents that sense, act, understand and communicate in natural language, think, and exhibit human-like personalities. Our work differs from other work in personality modeling in that we are not attempting to simulate personality, but using definitions of personality to drive the behavior of synthetic agents. We want to derive behavior that is consistent with a given personality model, but not necessarily to ensure that the agent gives the appearance of having that personality.

Experiments

To test our claim that personality can be modeled by reinforcement learning components, we created a population of simple two-component multiple-goal reinforcement learning agents and ran them in a world that replicated experiments carried out with humans by psychologist John Atkinson. First we describe Atkinson's original research, and then discuss our virtual reconstruction of his experiments.

Atkinson's Ring Toss Experiment

John Atkinson was among the first researchers to study the existence and role of approach and avoidance motivation in human behavior. Prior to Atkinson's work, it was believed that test anxiety was equivalent to low achievement motivation. However, Atkinson showed that test anxiety is actually a separate avoidance motivation, a "fear of failure" dimension that works against and interacts with achievement motivation (Atkinson & Litwin, 1960). To test his hypothesis, he administered standard tests of achievement motivation and test anxiety to a group of undergraduate psychology students and devised a series of experiments which examined the effort put forth in achieving success in tasks such as taking a final exam. It is important to note that he did not measure the outcomes of the task, but rather the effort put forth in doing well in them. Thus, his experiments examined the relationship between motivation and *behavior*, not necessarily competence. One of his experiments, a ring toss game, produced results that clearly show the interplay of approach and avoidance motivation and is particularly well-suited to computer simulation.

In Atkinson's ring toss experiment, subjects played a ring toss game in which players attempted to toss a ring from a specified distance onto a peg. Subjects made 10 tosses from any distance they wished, from 1 through 15 feet, and the distance at which each subject made each toss was recorded. For analysis, subjects were divided into four groups according to their measures of achievement motivation and test anxiety so that the relationship between these motivations and their behavior could be analyzed. For each of the two measures – achievement motivation and test anxiety – subjects were classified as either high or low, with the dividing line between high and low set at the median scores in each measure. (For example, a H-L subject is high in achievement motivation and low in test anxiety). Subjects were divided into four groups – H-L, H-H, L-L, and L-H – and the percentage of shots taken at each distance by each group was recorded. We discuss his results and our simulation below.

Computational Models of Atkinson's Subjects

We reconstructed Atkinson's ring toss experiment in a computer simulation. We created 49 virtual agents that corresponded to each of the 49 human subjects in Atkinson's experiments, with the same distribution of high and low measures of achievement motivation and test anxiety. Simplified code for a representative student subject is presented in Figure 1. Since we did not have access to Atkinson's source data, we modeled high motivation measures as having a mean of 1.5 and low motivation with a mean of 0.5, both with standard Normal distributions (mean = 0, variance = 1) scaled by $\frac{1}{2}$, so virtual test subjects did not all have the same measures.

```
1 object Student((Achievement, 1.5 + X ~ N(0, 1) / 2),
2                (TestAnxiety, .5 + X ~ N(0, 1) / 2))
3 }
```

Figure 1: An agent representing a success-oriented student in Atkinson's ring toss experiment, containing two RL components representing high achievement motivation and low test anxiety. The code snippets presented here are simplified versions of the Scala code we used to run our experiments.

As discussed earlier, each of the motivational dimensions of the virtual subjects was implemented with reinforcement learning components that learned to satisfy the preference for perceived valence of behaviors (modeled as reward). For example, in the achievement motivation component (see Figure 2), the greater the distance from the peg, the greater the reward because it represents greater achievement. Similarly, in

the test anxiety component (see Figure 3), greater reward is given to closer distances, because they minimize, or "avoid" the chance of failure from a long-distance toss.

```
object Achievement extends AbstractRlComponent {

  world = RingTossWorld

  rewards = (1_foot_line -> 1,
             2_foot_line -> 2,
             // ...
             15_foot_line -> 15)

  actions = (play_1_foot_line,
             play_2_foot_line,
             // ...
             play_15_foot_line)
}
```

Figure 2: A reinforcement learning component representing achievement motivation.

```

object TestAnxiety extends AbstractRlComponent {

  world = RingTossWorld

  rewards = (1_foot_line -> 15,
             2_foot_line -> 4,
             // ...
             15_foot_line -> 1)

  actions = (play_1_foot_line,
             play_2_foot_line,
             // ...
             play_15_foot_line)
}
```

Figure 3: A reinforcement learning component representing Test Anxiety ('avoidance motive, a.k.a. "fear of failure"). Note that the rewards are inverted from the achievement motivation component, that is, the valence of avoiding achievement is higher.

Internally, each personality component is implemented with the standard Q-learning algorithm (Sutton & Barto, 1998). The ring toss world consists of 16 states – a start state and one state for each of the 15 distances, and 15 actions available in each state that represent playing (making a toss) from a particular distance. For readers interested in such details, each reinforcement learning component used a step-size parameter of $\alpha = 0.1$, a discount factor of $\gamma = 0.9$ (though discounting wasn't important given that the 15 states representing playing lines were terminal states, since each play was a training episode), and employed an ε-greedy action selection strategy with $\varepsilon = 0.2$. (Readers familiar with reinforcement learning will also notice that this game is roughly equivalent to a 15-armed bandit problem.) We emphasize that the details of the reinforcement learning algorithms are not essential to modeling motivational factors, and those details are hidden inside the implementation of the components. Indeed a major goal of our work is to simplify the task of writing synthetic agents by taking care of such details automatically.

Recall that reinforcement learning algorithms learn an action value for each action available in a given state. An action value for a state represents the expected total reward that can be achieved from a state by executing that action and transitioning to a successor state. For each of the components – Achievement and TestAnxiety – the action values represent the learned utility of the actions in serving the motivational tendencies the components represent. The Student agents take into account the preferences of the components – represented by action values – by summing their action values weighted by their component weights to get a composite action value for each action in a given state. If we denote each component's action value by $Q(s,a)$ and the weights by W, then the composite, or overall, action value is:

$$Q_{student}(s,a) = W_{Achievement} Q_{Achievement}(s,a) + \quad (1)$$

$$W_{TestAnxiety} Q_{TestAnxiety}(s,a) \quad (2)$$

For the virtual experiments, each component – Achievement and TestAnxiety – was run to convergence and then the student agents simulated 10 plays of the ring toss game, just as in Atkinson's experiment. We discuss the results of the experiment below.

Model Validation

A model is a set of explicit assumptions about how some system of interest works (Law, 2007). In psychology the system of interest is (usually) a human or group of humans. Our virtual reconstruction of Atkinson's experiments constitutes a computational representation of Atkinson's two-factor model of personality. Thus, our agents are simulation models of Atkinson's subjects (the students in his ring toss experiment). While the work presented here is only a proof of concept, we do hope to achieve a high level of validity as we refine our approach, so it will be useful to validate our models using techniques from simulation science (Law, 2007).

As we described earlier, Atkinson divided his subjects into four groups according to their measures (high or low) on achievement motivation and test anxiety. For each of these four groups – H-L, H-H, L-L, L-H – he recorded the percentage of shots that each group took from each of the 15 distances. We ran 10 replications of our simulation and recorded the mean percentages for each group and distance. For each percentage mean we calculated a 95% confidence interval. We consider a model to be valid if the confidence intervals calculated on the simulation percentage means contain the percentages obtained by Atkinson in his experiments with human subjects.

The validation results are presented in Table 1. Atkinson analyzed his experimental data by aggregating the shots taken by subjects into three "buckets" representing low, medium, and high difficulty. In Atkinson's analyses the dividing lines between the three buckets were set in four different ways with each yielding similar results. For brevity we present the division obtained by using both geographical distance and distribution of shots about the median shot of 9.8 ft, in other words, the dividing line one would choose by inspecting the histogram for distinct regions. This strategy resulted in the three buckets listed in the left column of Table 1. Each cell of the four subject groups – H-L, H-H, L-L, L-H - contains the

Table 1: Validation Results. For each subject group the percentage of shots taken by Atkinson's human subjects and by our simulation from each of three ranges is presented along with a 95% confidence interval for the mean percentage of shots in 10 simulated replications of Atkinson's experiment.

Achievement:	High	High	Low	Low
Test Anxiety:	Low	High	Low	High
	Atkinson	Atkinson	Atkinson	Atkinson
	Simulation	Simulation	Simulation	Simulation
Range	Conf. Int.	Conf. Int.	Conf. Int.	Conf. Int.
1-7	11	26	18	32
	7.7	14.0	5.6	8.5
	(4.0, 11.4)	(5.6, 22.4)	(1.4, 9.7)	(4.4, 12.5)
8-12	82	60	58	48
	75.4	69.0	74.4	80.0
	(65.1, 85.7)	(61.1, 76.9)	(62.0, 86.9)	(74.1, 85.9)
13-15	7	14	24	20
	16.9	17.0	20.0	11.5
	(8.8, 25.0)	(9.4, 24.6)	(8.3, 31.7)	(6.9, 16.2)

percentage of shots taken by Atkinson's subjects, the mean percentage obtained by running 10 replications of our simulation of Atkinson's experiment, and a 95% confidence interval for the mean percentage. While our model did not achieve formal validation, the general patterns of behavior are quite similar to Atkinson's human subject experiment and we consider these results to be a good proof of concept. We discuss some reasons behind these results and strategies for improvement below.

Discussion

We made several assumptions in our models that affected the validation results. First, because we did not have access to Atkinson's original data, only summary presentations, we did not know the exact distribution of motivational factors among his subjects, or even the scales used in his measures. We assumed normally distributed measures and tried several different scales before settling on the values used in the simulations reported here. Second, it is not clear how the valence of behaviors should be translated into reward structures for RL agents. We chose a simple linear reward structure in hopes that the system would be robust to naive encodings. To make our approach widely useful we will need to address the manner in which reward structures are determined. Third, we calculated aggregate action values by a simple weighted sum of component action values. We are currently investigating optimal arbitration of multiple RL components and hope to report results within the next six months.

We chose the Atkinson ring toss experiment on the advice of psychologists who recommended it as a well-known example of trait-oriented behavior theory, and because of its simplicity. However, our goal is to create large agent systems, so future work will need to address scalability – to greater numbers of trait factors and more complex worlds – and generalizability, or transferability, to other domains.

The algorithms we used also employed no optimization. Reinforcement learning suffers from the curse of dimensionality, and many techniques are being actively pursued to cope with the size of state spaces for realistic-size domains. Profitably employing reinforcement learning in agent programming systems will mean integrating scaling techniques such as function approximation (e.g., of action-value functions or state spaces) and decomposition techniques.

Finally, notice that the example code presented in this paper contains no logic for implementing behavior. The agents and the components are defined declaratively by specifying a state space, an action set, and a reward structure. The runtime system derives the concrete behavior of the agents automatically from these specifications. This technique, sometimes called partial programming (Simpkins, Bhat, & Isbell, 2008), is a key concept that increases the usability of agent programming by allowing programmers to specify *what* an agent is to do without getting mired in *how* the agent should do it.

Conclusions and Future Work

Reinforcement learning provides a promising approach to modeling personality traits and motivational factors in synthetic agents. In particular, it provides us with a means to create agent programming systems that are accessible to behavioral scientists and harness their knowledge directly while minimizing the need for complex programming. Much work remains to make this vision a reality, and our work is progressing on three paths. First, the integration of reinforcement learning into agent programming systems needs to be studied further so that we know when it is useful and how much detail

can be hidden from the agent programmer. Second, the examples presented here were written together so that the reward signals of each agent were directly comparable. If we want to enable large-scale agent programming, we must be able to arbitrate the reward signals of separately-authored reinforcement learning components (Bhat, Isbell, & Mateas, 2006). We are currently working on such an arbitration algorithm and hope to have results in the very near future. Finally, once the implications of integrating reinforcement learning components into agent models are sufficiently well understood and separately authored components can be combined in a modular fashion using an appropriate arbitration algorithm, we believe the best way to realize these benefits is in a language that incorporates these features in a coherent design. We are currently working on such a language, initially implemented as an internal domain-specific language (DSL) in Scala.

Acknowledgments

The authors are grateful for the support of the National Science Foundation and the Georgia Tech Research Institute. Patrick McNiel in the psychology department suggested the Atkinson example and was very generous in discussing our work and helping us understand personality psychology. Dr. Doug Bodner assisted us in applying validation techniques from simulation science.

References

Anderson, J. R., Bothell, D., & Byrne, M. D. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036—1060.

Atkinson, J. W., & Litwin, G. H. (1960). Achievement motive and test anxiety conceived as motive to approach success and motive to avoid failure. *Journal of Abnormal and Social Psychology*, *60*(1), 52–63.

Bhat, S., Isbell, C., & Mateas, M. (2006, July). On the difficulty of modular reinforcement learning for real-world partial programming. In *Proceedings of the twenty-first national conference on artificial intelligence (aaai-06).* Boston, MA, USA.

Cervone, D., & Pervin, L. A. (2009). *Personality: Theory and research*. John Wiley and Sons.

Cervone, D., & Shoda, Y. (1999). The coherence of personality: Social-cognitive bases of consistency, veriability, and organization. In D. Cervone & Y. Shoda (Eds.), (pp. 3–33). New York: Guilford Press.

Dweck, C. S., & Leggett, E. L. (1988). A social-cognitive approach to motivation and personality. *Psychological Review*, *95*(2), 256–273.

Elliot, A. J., & Thrash, T. M. (2002). Approach-avoidance motivation in personality: Approach and avoidance temperaments and goals. *Journal of Personality and Social Psychology*, *82*(5), 804–818.

Gratch, J., & Marsella, S. (2005). Lessons from emotion psychology for the design of lifelike characters. *Journal of Applied Artificial Intelligence (special issue on Educational Agents - Beyond Virtual Tutors)*, *19*(3-4), 215–233.

Ho, Y.-C., & Pepyne, D. L. (2001, December). Simple explanation of the no free lunch theorem of optimization. In *Proceedings of the 40th ieee conference on decision and control* (pp. 4409–4414). Orlando, Florida USA.

John, B. E. (1998, June). Cognitive modeling for human-computer interaction. In *Invited paper in the proceedings of graphics interface '98.* Vancouver, British Columbia, Canada.

Jones, R. M. (2005). An introduction to cognitive architectures for modeling and simulation. In *Proceedings of the interservice/industry training/simulation and education conference.*

Kaelbling, L. P., Littman, M. L., & Moore, A. P. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237-285.

Laird, J. E. (2008). Extending the soar cognitive architecture. In *Proceedings of the first conference on artificial general intelligence (agi-08).*

Langley, P., Laird, J. E., & Rogers, S. (2008). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*.

Law, A. M. (2007). *Simulation modeling and analysis* (4th ed.). McGraw-Hill.

Loyall, A. B., & Bates, J. (1991). *Hap: A reactive adaptive architecture for agents* (Tech. Rep. No. CMU-CS-91-147).

Mateas, M., & Stern, A. (2004). Life-like characters. tools, affective functions and applications. In H. Prendinger & M. Ishizuka (Eds.), (chap. A Behavior Language: Joint Action and Behavioral Idioms). Springer.

McCrae, R. R., & Paul T. Costa, J. (2008). Handbook of personality: Theory and research. In O. John, R. Robins, & L. Pervin (Eds.), (pp. 159–181). New York: Guilford.

Mischel, W., & Shoda, Y. (2008). Handbook of personality: Theory and research. In O. John, R. Robins, & L. Pervin (Eds.), (pp. 208 – 241). New York: Guilford.

Nason, S., & Laird, J. E. (2008). Soar-rl: Integrating reinforcement learning with soar. In *6th international conference on cognitive modeling.* Pittsburgh, PA.

Simpkins, C., Bhat, S., & Isbell, C. (2008, October). Towards adaptive programming: Integrating reinforcement learning into a programming language. In *Oopsla '08: Acm sigplan conference on object-oriented programming, systems, languages, and applications, onward! track.* Nashville, TN USA.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Swartout, W., Gratch, J., Hill, R., Hovy, E., Marsella, S., Rickel, J., et al. (2006). Toward virtual humans. *AI Magazine*, *27*(1).

Dynamic Behaviour of a Spiking Model of Action Selection in the Basal Ganglia

Terrence C. Stewart (tcstewar@uwaterloo.ca)
Xuan Choo (fchoo@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
Waterloo, ON, N2L 3G1

Abstract

A fundamental process for cognition is action selection: choosing a particular action out of the many possible actions available. This process is widely believed to involve the basal ganglia, and we present here a model of action selection that uses spiking neurons and is in accordance with the connectivity and neuron types found in this area. Since the parameters of the model are set by neurological data, we can produce timing predictions for different action selection situations without requiring parameter tweaking. Our results show that, while an action can be selected in 14 milliseconds (or longer for actions with similar utilities), it requires 34-44 milliseconds to go from one simple action to the next. For complex actions (whose effect involves routing information between cortical areas), 59-73 milliseconds are needed. This suggests a change to the standard cognitive modelling approach of requiring 50 milliseconds for all types of actions.

Keywords: action selection; basal ganglia; spiking neurons; Neural Engineering Framework; cognitive cycle time

Action Selection

The basal ganglia are generally believed by both neuroscientists (e.g. Redgrave et al., 1999) and cognitive scientists (e.g. Anderson et al., 2004) to be responsible for action selection. Action selection consists of choosing one action to perform out of the many actions in an organism's repertoire. Selection is done on the basis of some sort of context-dependent utility signal for each possible action. Actions that are inappropriate for the current context may have low utility, and a task of the basal ganglia is to select the action that currently has the highest utility value.

Since such a mechanism forms the core of many cognitive models, including all of those based on production systems (where a single production much be chosen to fire), it is useful to develop a computational model of this process. Here, we develop a detailed spiking neuron model that takes into account a broad range of neurological details about the basal ganglia. Other spiking models of action selection exist, but tend to be organized unlike the basal ganglia (Belavkin & Huyck, 2009) and unconstrained by neural properties (Shouno et al., 2009; see Humphries et al., 2006 for an exception and alternate approach).

By directly connecting our model to neuroscientific results, we constrain our parameter values. Every parameter in the model reflects neurological data from the relevant brain areas, resulting in a model that has no free parameters (that affect the results shown here). Furthermore, having a biologically realistic model allows us to make predictions about a wide range of measures, including spike patterns, timing, variability, lesion effects, neural degeneration, the influence of various drugs, and so on. Importantly, all of these predictions can come from the same model, with no additional parameters.

Neural Structure

The basal ganglia are a group of subcortical structures that are ideally suited for an action selection operation, as they receive input from extremely broad areas of cortex and the limbic system, and send output back to these areas via the thalamus. The basic components are the striatum, the subthalamic nucleous (STN), the globus pallidus internal (GPi), the globus pallidus external (GPe), and the substantia nigra pars reticulata (SNr).

The classic way of thinking about the organization of the basal ganglia is shown in Figure 1A. It consist of a direct pathway, where excitatory inputs from cortex to the D1 cells in the striatum inhibit corresponding areas in GPi and SNr, which then in turn inhibit areas in the thalamus, and an indirect pathway from the D2 cells in the striatum to GPe, STN, and then GPi/SNr (Albin et al., 1989). However, more recent evidence shows other major connections, including a hyperdirect excitatory pathway straight from cortex to STN (Nambu et al., 2002), and other feedback connections, as shown in Figure 1B.

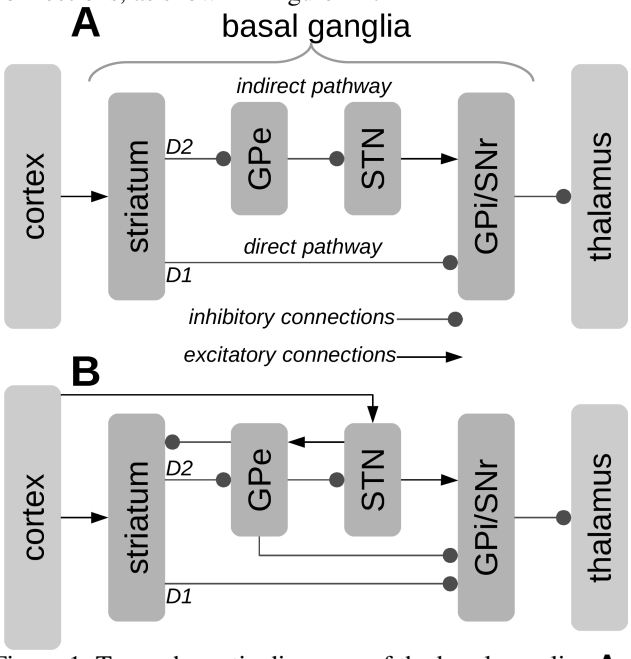


Figure 1: Two schematic diagrams of the basal ganglia. **A** shows the standard direct/indirect pathway. **B** includes the other major connections that have been discovered.

There is also a great deal of topological structure in the inhibitory connections in basal ganglia. Neurons in the striatum project to a relatively localized area in the GPi, GPe, and SNr, while the excitatory connections from STN are very broad (Mink, 1996). This is an important constraint for the model we discuss below.

Simple Action Selection Models

Two simple approaches to neurally modelling action selection are shown in Figure 2. The inputs give the utilities of three possible actions (0.3, 0.8, and 0.5), and the model's task is to chose one of them. Importantly, since the output from the basal ganglia is inhibitory, selecting an action consists of having that particular inhibitory output be zero. In other words, it will no longer inhibit the neurons to which it is connected, allowing the action to occur. Thus, in Figure 2, the selected action is the middle one, whose output value is zero in both cases.

The model in Figure 2A is the most straight-forward. Each input neuron inhibits its corresponding output neuron and excites all others. For the first action, this results in an output of -0.5*0.3+0.5*0.8+0.5*0.5=0.5. The action with the largest input will have the smallest output, and if the weights are in suitable ranges, only one output neuron will be turned off. One problem with this approach is determining suitable weights, although this can be helped by introducing recurrent connections, as in our earlier model (Stewart & Eliasmith, 2009). However, a more fundamental problem is that real neurons are typically either excitatory or inhibitory, and seldom both, as they are in this model.

An alternate approach is shown in Figure 2B. Here, instead of each neuron being both excitatory and inhibitory, a separate inhibitory interneuron is introduced. These are found throughout the brain, and can be used here to divide up the excitatory and inhibitory parts of the task. This approach is commonly used in neural models of action selection (e.g. Hazy et al., 2007; Stocco et al., 2010).

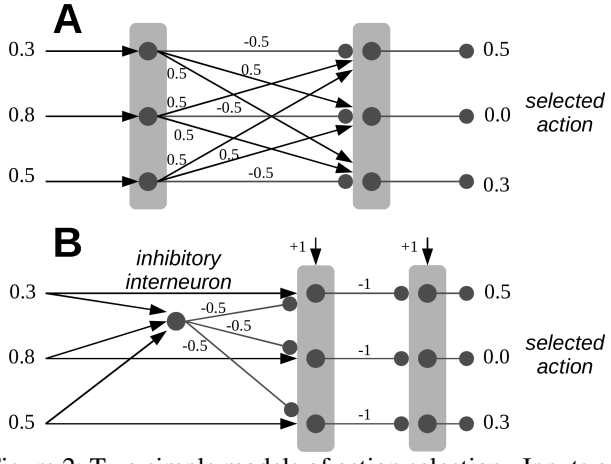


Figure 2: Two simple models of action selection. Inputs are the utilities of three possible actions, and an output of zero indicates the selection of a particular action. Each neuron (circle) outputs the sum of its weighted inputs.

A Realistic Rate Neuron Model

Gurney, Prescott, and Redgrave (2001) have developed a computational model of the basal ganglia that is well-suited to reimplementation using more realistic spiking neurons. While their model uses rate neurons, they have carefully followed the known biological constraints on the connectivity and types of neurons in the basal ganglia.

One of the main differences between their model and other computational models (e.g. Hazy et al., 2007; Stocco et al., in press) is that it does not make use of inhibitory interneurons in the striatum to perform action selection (as in Figure 2B). This is important for two reasons. First, while the striatum does include inhibitory interneurons, the actual behaviour and biological characteristics of these neurons is unclear, making them difficult to model. Second, there seems to be little evidence of the sort of broad, diffuse connectivity required by figure 2B (Gurney, et al., 2001). Tepper and Bolam (2004) identify three different types of striatal interneurons, and demonstrate their ability to affect spike timing in the rest of the striatum. These interneurons are highly influenced by dopamine (Bracci et al., 2002), acetylcholine (Koos & Tepper, 2002), and seratonin (Blomeley & Bracci, 2009), indicating that their role may be more to do with learning and other large-scale cognitive processes than with action selection.

Instead, Gurney, Prescott, and Redgrave (2001) present a model where the inhibitory output from the striatum and the excitatory output from the subthalamic nucleous (STN) combine to produce the desired output. That is, instead of treating the striatum as the primary input to the basal ganglia, neurological evidence shows that the STN receives excitatory connections directly from the cortex, and then produces diffuse excitation in the output nuclei. Figure 3 shows how this leads to an action selection mechanism that separates the inhibitory and excitatory connections.

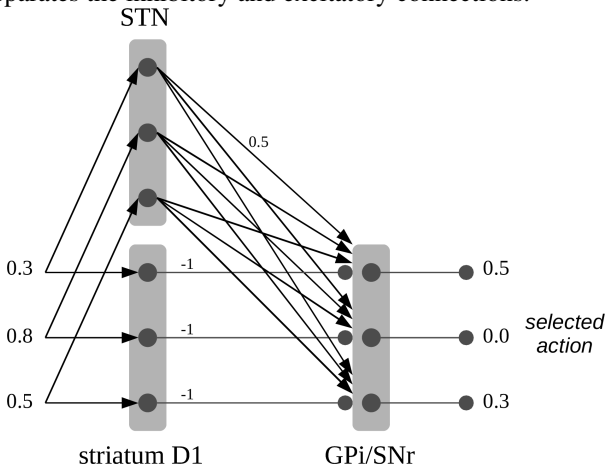


Figure 3: Action selection via the striatum D1 cells and the subthalamic nucleous (STN). Connections from the STN are all excitatory and set at a weight of 0.5. The input with the highest utility (0.8) causes the corresponding output in the globus pallidus internal (GPi) or substantia nigra (SNr) to drop to zero, stopping the inhibition of that action.

While the model shown in Figure 3 is sufficient for action selection in some circumstances, it turns out not to be fully general. In particular, it has difficulty adjusting to situations where there are many actions with large utilities or where all actions have low utilities. For this reason, a control system is needed to modulate the behaviour of these neural groups. Gurney et al. (2001) argue that the globus pallidus external (GPe) is ideally suited for this, as its only outputs are back to the other areas of the basal ganglia, and it receives similar inputs from the striatum and the STN as does the globus pallidus internal (GPi). In their model, the GPe forms a circuit identical to that in Figure 3, but its outputs project back to the STN and the GPi. This regulates the action selection system, allowing it to function across a full range of utility values. The final network is shown in Figure 4.

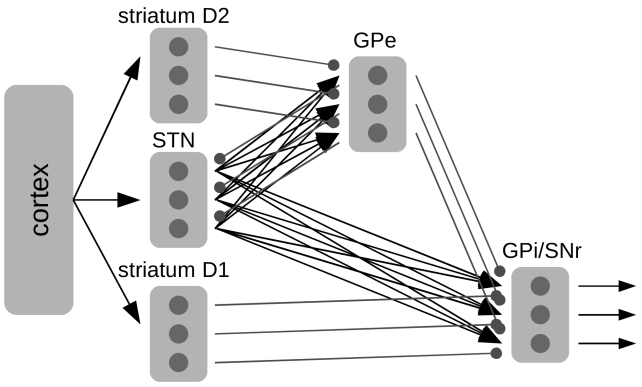


Figure 4: The model of action selection in the basal ganglia presented by Gurney, Prescott, and Redgrave (2001). The striatum D1 cells and the subthalamic nucleous (STN) are as in Figure 3, while the striatum D2 cells and globus pallidus external form a modulatory control structure.

Converting Rates to Spikes

The model discussed so far is capable of performing action selection and reproducing a variety of single-cell recording results from electrostimulation and lesion studies (Gurney et al., 2001). However, it does so with rate neurons; that is, the neurons do not spike and instead continually output a numerical value based on their recent input. This makes it difficult to make precise numerical timing predictions or to make use of more accurate neural models. Furthermore, the model has no redundancy, since exactly one neuron is used per area of the basal ganglia to represent each action. The model shown in Figure 4 uses a total of 15 neurons (dark circles) to represent 3 possible actions, and if any one of those neurons is removed the model will fail.

To make timing predictions and to constrain our model with a broader range of neurological details, we needed to adapt the rate model of the basal ganglia into one that uses spiking neurons. For the results shown here, we use the standard leaky integrate-and-fire (LIF) model of spiking neuron behaviour, although our initial results with a more detailed implementation of the medium spiny neurons in the striatum (Gruber et al., 2002) are similar.

For LIF neurons, current is constantly leaking out of the neuron as per the membrane resistance *R*. If enough input current is gathered to cause the voltage to be above a certain threshold, then the neuron will fire. After firing, the voltage is set to 0 for a fixed refractory period (~2 milliseconds) before starting to gather current again. Given a constant current input *J* and membrane resistance *R*, the voltage level of the LIF neuron changes over time as given in Equation 1 and shown in Figure 1. The timing of this behaviour is controlled by τ_{RC}, the membrane time constant of the neuron.

$$V(t) = J R (1 - e^{-t/\tau_{RC}}) \quad (1)$$

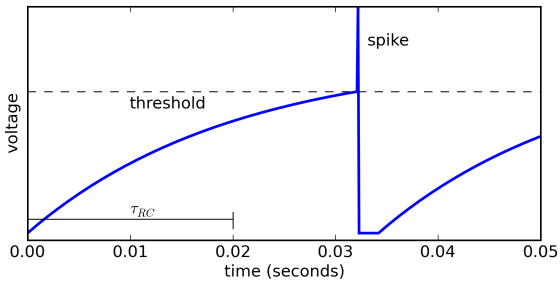


Figure 5: LIF neuron with constant input current.

For a constant input, we can measure the average firing rate of a given LIF neuron, and this will be dependent solely on the neurophysicological details of the resistance R and the membrane time constant, which tend to be fixed for any particular type of neuron. However, for real *in vivo* neurons, their output will also vary based on any background current flowing into the neurons, and their activity can be scaled by the strength of the incoming synaptic connection. Thus, even among neurons of the same type, their responses will vary, as shown in Figure 6A. The behaviour of a neuron as its input varies is known as its *tuning curve*, and the ones shown in Figure 6A are typical for neurons throughout the brain.

In Figure 6B, we show the tuning curve for the rate neurons used by Gurney et al. (2001). This does not look like the realistic tuning curves of Figure 6A. However, Figure 6C shows that we can implement the effects of such a tuning curve by adding together the realistic tuning curves of 6A. This allows a group of realistic neurons to provide a similar effect to that assumed by the model.

When adding the outputs of the spiking neurons, we scale each one by a factor d_i, producing a weighted sum. We can compute the optimal d_i values using Equation 2, where the integration is over all possible inputs $\boldsymbol{x}$, a_i is the average firing rate of neuron i given input $\boldsymbol{x}$, a_j is the same for neuron j, and $f(\boldsymbol{x})$ is the desired output (Figure 6B). This calculation determines the least-squared-error solution for mapping the neural tuning curves onto the function $f(\boldsymbol{x})$. The method extends to complex functions and multiple dimensions, making it the basis of the Neural Engineering Framework (Eliasmith & Anderson, 2003).

$$\boldsymbol{d} = \Gamma^{-1} Y \qquad \Gamma_{ij} = \int a_i a_j \, dx \qquad Y_j = \int a_j f(\boldsymbol{x}) \, dx \quad (2)$$

While there clearly must be a developmental or learning-based mechanism to determine these weights, we do not consider this here, just as we do not consider the developmental process for the creation of these separate brain areas in the first place. Instead, we assume that whatever such mechanisms exist converge to weights near the values determined by Equation 2.

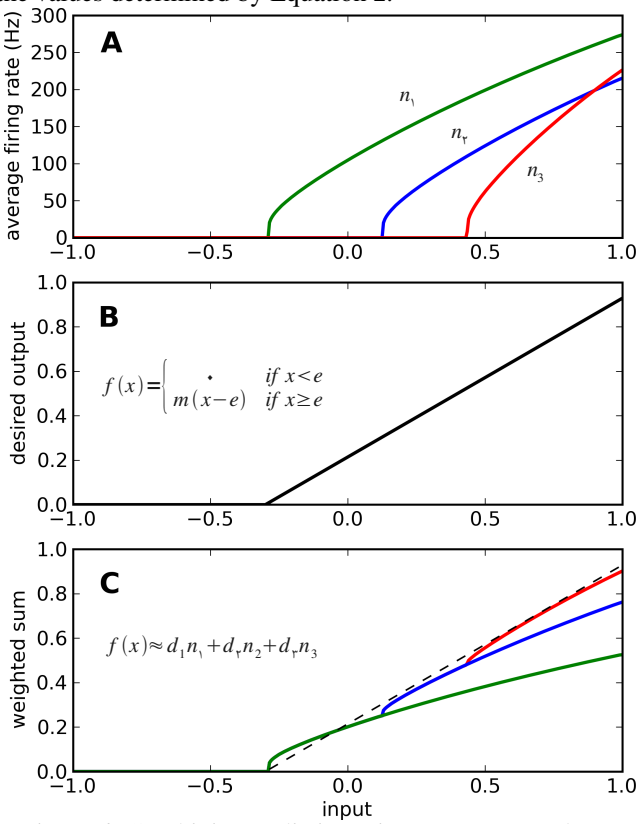


Figure 6: Combining realistic tuning curves to produce a desired function. **A** shows the average firing rate of three different neurons as the amount of input to the neurons increases. **B** shows the neural output function used by the rate neuron model. **C** shows how **B** can be approximated by taking a weighted sum of tuning curves in **A**.

Given these weighting values d_i, we can construct a spiking version of the model shown in Figure 4. Each single neuron in the original model is replaced by a set of 20 spiking neurons (increasing this value does not change our results). These all have the same time constant (τ_{RC}=20ms; common throughout the brain), but have varying background currents and scaling factors to produce the range of tuning curves seen in Figure 6A. Each connection in the original model from rate neuron A to rate neuron B is replaced by a set of connections from all of the spiking neurons replacing A to all of the spiking neurons replacing B. The actual synaptic connection weight from the ith neuron in A to the jth neuron in B is $w\alpha_j d_i$, where α is the neuron's scaling factor and w is the original rate model's connection weight.

Finally, the timing effects of a neuron firing must be considered. This is vital for producing realistic temporal predictions from a model of spiking neurons. When a neuron fires, it sends current into all of the neurons to which it is connected. This current $h(t)$ can be characterized by Equation 3, where τ_s captures the effects of neurotransmitter re-uptake and dispersal. As shown in Figure 7, a small τ_s provides a fast, short-lasting effect (~10ms), while others last for hundreds of milliseconds.

$$h(t)=t\,e^{-t/\tau_s} \qquad (3)$$

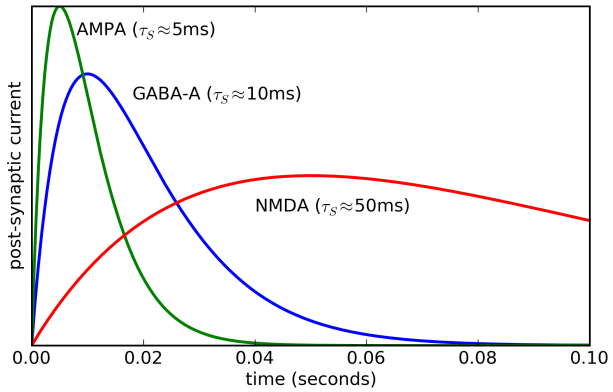


Figure 7: Post-synaptic currents for common synapses.

Importantly, different neurotransmitters are used by the different types of connections in the basal ganglia. All of the inhibitory connections involve GABA (τ_s=6.1ms to 10.5ms; Gupta et al., 2000), while the excitatory ones of concern for this model involve fast AMPA-type glutamate receptors (τ_s=2ms; Spruston et al., 1995). This means that the excitation and inhibition in the model act at different times scales, a factor not taken into account in the original model. As we show below, the time constants of these neurotransmitters have a strong impact on the temporal behaviour of our model.

Results

Figure 8 demonstrates that the model is capable of correctly performing action selection. Initially, action B has the highest utility, and the output shows that B is the only action that is not inhibited by the GPi/SNr outputs. In the middle, C is selected and has the highest activation, followed by A.

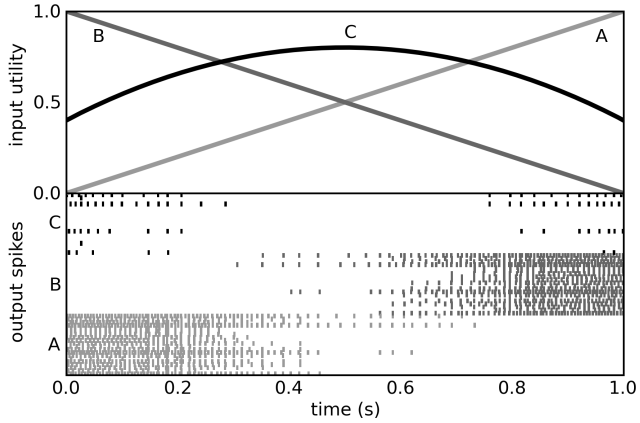


Figure 8: Spikes produced (bottom) for three possible actions (A, B, and C) as their utility changes (top).

Response Latency

One of the key advantages of using a realistic neural model is that timing predictions emerge from the neural parameters. We start by determining how long it takes the model to select an action when there is a sudden change in the input. Figure 9 shows the output for an action when its utility is suddenly increased at t=0. This matches empirical findings that in the rat basal ganglia, output neurons stop spiking 14 to 17 milliseconds after a similar input pulse (Ryan & Clark, 1991).

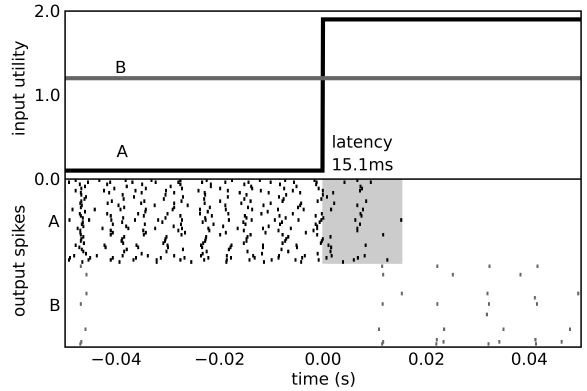


Figure 9: Spiking produced (bottom) for a sudden change in utility (top). Firing for action A stops 15.1ms after its utility is increased.

We can also examine how long it takes the model to decide between two actions as we adjust the difference between the top two utility values. Figure 10 indicates how the latency changes from very similar utility values (38ms mean latency, standard deviation 8.8ms) to highly differing utility values (14ms mean latency, standard deviation 1.5ms). As far as we are aware, this is a novel prediction.

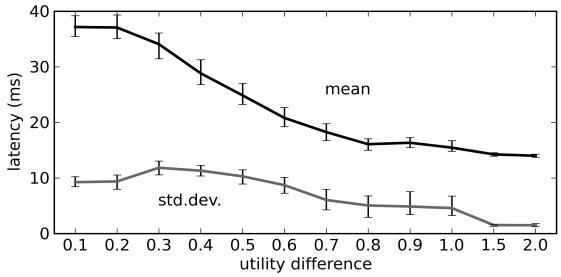


Figure 10: Mean and standard deviation of basal ganglia response latency as for varying differences between utilities. Error bars are 95% confidence intervals over 200 runs.

Cognitive Cycle Timing

In a full cognitive system, the output of the basal ganglia would be used to affect the firing of other areas of the brain (via the thalamus). This, in turn, will affect the input to the basal ganglia, perhaps causing a different action to be selected. This is the basis of our ongoing development of a full production system using spiking neurons (Stewart, Choo, and Eliasmith, 2010). To investigate how long this whole cycle requires, we need to include the thalamus and a simple cortical area in our model.

For the cortex, we create a group of 5000 spiking neurons representing the current state. These are connected to the inputs to the basal ganglia so that the utility input for each action will be the similarity (measured as the dot product) between the current state and the ideal state for that action. This is done using Equation 2, where *f(x)* is this similarity measure. For the thalamus, we create neurons representing the actions of switching to each possible state. They are connected to the cortex similarly, such that the firing of one group of neurons in the thalamus will cause the cortical neurons to fire in a pattern representing that state.

To implement the chaining of actions one after the other, we connect the output of the basal ganglia to the thalamic neurons such that if the basal ganglia selects action A, this will stop the inhibition of the thalamic neurons representing state B, thus causing the cortex to go to state B, and the basal ganglia to select action B. The actions are chained so that A leads to B, B leads to C, C leads to D, and so on. This can be thought of as a set of production rules of the form "If A then B; If B then C; If C then D; etc." The newly added connections are excitatory, using AMPA-type receptors (τ_s=2ms). All other parameters remain the same.

With this model, we can measure the time taken to change from one action to the next. This provides a measure of the minimum amount of time needed to go from one step to the next in a sequence of cognitive actions. In cognitive models that use production systems, extensive behavioural data has been gathered indicating that this value should be around 50 milliseconds (Anderson et al., 1995).

Figure 11 shows the mean and standard deviation of the cycle times produce by our model. The shaded area shows the timing produced when the correct realistic time constants for the inhibitory GABA neurotransmitter are used. Importantly, there are no parameters in our model that we can vary to affect this performance. In should be noted that our model predicts cycle times between 34 and 44 milliseconds, which is somewhat shorter than the standard 50 milliseconds value. However, this result is only for simple actions: more complex actions are considered next.

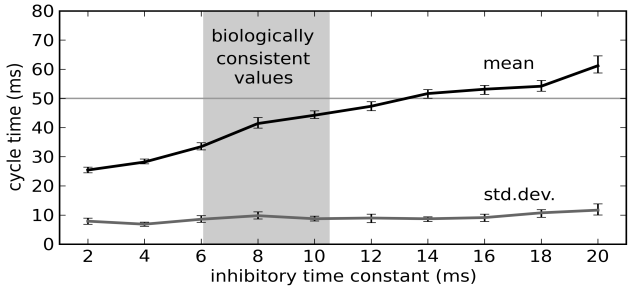


Figure 11: Cognitive cycle times produced by our model as the time constant τ_s of the inhibitory neurotransmitter GABA varies. The shaded area indicates parameter settings consistent with neurophysiology (Gupta et al., 2000). Cognitive models generally use a cycle time of 50ms.

To be cognitively useful, an action selection mechanism needs to be able to trigger more complex actions than those considered so far. In particular, production system rules generally allow actions that can send a value stored in one

brain area to another. To model this we can create connections between cortical areas such that driving a cortical area to a particular value causes a second cortical area to send its value to a third cortical area. This can be implemented using Equation 2 (see Stewart, Choo, and Eliasmith, 2010 for more details). The timing of these types of actions are shown in Figure 12. While simple actions require less than 50 milliseconds, complex actions require more than 50 milliseconds.

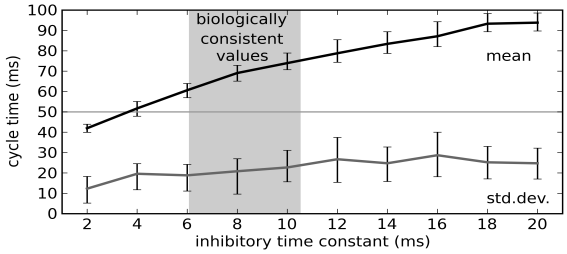


Figure 12: Cognitive cycle times produced for complex actions by our model as the time constant of the inhibitory neurotransmitter GABA varies.

Conclusions

We presented a spiking neuron model of action selection that matches the anatomy of the basal ganglia and does not assume the presence of diffuse inhibitory interneurons in the striatum. By constraining the neurons' behaviour to match that of real neurons in the basal ganglia, we produce timing predictions from our model without parameter fitting. Figure 9 shows that these predictions match well for single-cell recordings in rats, and Figure 11 shows a close match for a wide range of cognitive psychology results. Our model thus provides a neural explanation of the commonly used 50 millisecond cognitive cycle time (e.g. Anderson et al., 1995). It also produces novel predictions of increases to this cycle time for situations where two possible actions have similar utilities (Figure 10) and for actions involving information transfer between brain areas (Figure 12).

References

Albin, R. L., Young, A. B. & Penney, J. B. (1989). The functional anatomy of basal ganglia disorders. *Trends in Neurosciences*, 12(10): 366-375.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review 111*(4), 1036-1060.

Anderson, J. R., John, B. E., Just, M. A., Carpenter, P. A., Kieras, D. E., & Meyer, D. E. (1995). Production system models of complex cognition. *17th Annual Meeting of the Cognitive Science Society.*

Belavkin R. & Huyck, C. (2009). A model of probability matching in a two-choice task based on stochastic control of learning in neural cell-assemblies. *9th International Conference on Cognitive Modelling*.

Blomeley, C.P. & Bracci, E. (2009). Serotonin excites fast-spiking interneurons in the striatum. *The European journal of neuroscience,* 29(8):1604-1614.

Bracci, E., Centonze, D., Bernardi, G., & Calabresi, P. (2002). Dopamine excites fast-spiking interneurons in the striatum. *Journal of neurophysiology*. 87(4):2190-2194.

Eliasmith, C. & Anderson, C. (2003). *Neural Engineering: Computation, representation, and dynamics in neurobiological systems.* Cambridge: MIT Press.

Gupta, A., Wang, Y., & Markram, H. (2000). Organizing Principles for a Diversity of GABAergic Interneurons and Synapses in the Neocortex. *Science 287*(5451), 273-278.

Gurney, K., Prescott, T., & Redgrave, P. (2001). A computational model of action selection in the basal ganglia. *Biological Cybernetics 84*, 401-423.

Hazy, T.E., Frank, M.J., & O'Reilly, R.C. (2007). Towards an executive without a homunculus: computational models of the prefrontal cortex/basal ganglia system. *Philosophical Transactions of the Royal Society B.*

Humphries, M., Stewart, R., & Gurney, K. (2006). A physiologically plausible model of action selection and oscillatory activity in the basal ganglia. *The Journal of Neuroscience, 26*(50), 12921-12942.

Koós, T. & Tepper, J.M. (2002). Dual cholinergic control of fast-spiking interneurons in the neostriatum. *The Journal of Neuroscience,* 22(2), 529-535.

Mink, J. W. (1996). The basal ganglia: Focused selection and inhibition of competing motor programs. *Progress in Neurobiology*, 50, 381-425.

Nambu, A., Tokuno, H. & Takada, M. (2002). Functional significance of cortico-subthalamo-pallidal 'hyperdirect' pathway. *Neuroscience Research*, 43, 111-117.

Redgrave, P., Prescott, T., & Gurney, K. (1999). The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience 86*, 353-387.

Ryan, L. & Clark, K. (1991). The role of the subthalamic nucleous in the response of globus pallidus neurons to stimulation of the pre-limbic and agranular frontal cortices in rats. *Exp Brain Res, 86*, 641-651.

Shouno, O., Takeuchi, J., & Tsujino, H. (2009). A spiking neuron model of the basal ganglia circuitry that can generate behavioral variability. *The Basal Ganglia IX: Advances in Behavioral Biology.* Springer.

Spruston, N., Jonas, P., & Sakmann, B. (1995). Dendritic glutamate receptor channel in rat hippocampal CA3 and CA1 pyramidal neurons. *J. Physiology 482*, 325-352.

Stewart, T., Choo, X., & Eliasmith, C. (2010). Symbolic reasoning in spiking neurons: A model of the cortex/basal ganglia/thalamus loop. *32nd Annual Meeting of the Cognitive Science Society.*

Stewart, T., & Eliasmith, C. (2009). Spiking neurons and central executive control: The origin of the 50-millisecond cognitive cycle. *9th International Conference on Cognitive Modelling.*

Stocco, A., Lebiere, C., & Anderson, J. R. (in press). Conditional routing of information to the neocortex: A network model of basal ganglia function. *Psych. Review*.

Tepper, J.M. & Bolam, J.P. (2004). Functional diversity and specificity of neostriatal interneurons. *Current opinion in neurobiology,* 14:685-692.

A Temporally Asymmetric Hebbian Network for Sequential Working Memory

Jared C. Sylvester (jsylvest@umd.edu)
James A. Reggia (reggia@cs.umd.edu)
Department of Computer Science, A.V. Williams Bldg., College Park, MD 20742 USA

Scott A. Weems (sweems@casl.umd.edu)
Michael Bunting (mbunting@casl.umd.edu)
Center for Advanced Study of Language, 7005 52nd Avenue, College Park, MD 20742

Abstract

Recurrent connections combined with the appropriate dynamics enable oscillatory neural networks to produce rhythmic activity patterns. Such oscillatory activity can represent multiple stored patterns simultaneously, rather than the single pattern of a fixed-point network. However, retrieving these stored patterns in the same order as they were seen has proven challenging. In this paper we modify a recently developed simple oscillatory memory capable of storing temporal sequences so that it will now retrieve remembered items in the same order presented. This was achieved through the use of a temporally asymmetric weight matrix. The network is still capable of matching the recall performance of human subjects, reproducing the recency effect they exhibit in working memory tasks and displaying similar position-specific recall rates. We conclude that augmenting simple oscillatory neural network models with temporally asymmetric synaptic connections substantially improves their ability to match human short term memory properties.

Keywords: neural network models; autoassociative memory; short-term working memory; Hebbian learning; serial order

Introduction

There has been increasing interest in recent years in the development of oscillatory neural network models for a variety of tasks. In contrast to fixed-point attractor networks, which are typically limited to activating a single pattern in memory at a time, oscillating networks have dynamics characterized by recurrent connections leading to persistent rhythmic activity. This allows multiple patterns to be held in the same short-term memory concurrently as the model's state persistently switches between them.

A large variety of oscillating neural models exist. For example, some are based on underlying theta/gamma activity in the hippocampus or neocortex (Hasselmo, Bodelon, & Wyble, 2002; Ingber, 1995; Lisman & Idiart, 1995), while others use individual spiking neurons (Raffone & Wolters, 2001). Other more abstract approaches have also been used, for example Wilson-Cowan oscillators (Chakravarthy & Ghosh, 1996; Wang, 1995).

Here we concentrate on modeling short-term working memory, which is active over periods of time on the order of several seconds. A key characteristic of working memory is that it has a very limited capacity, unlike long-term memory (Baddeley, 2000). Recent studies suggest that this capacity is capped at around four items (Cowan, 2001; Cowan et al., 2005). More specifically we concern ourselves with modeling working memory for sequential tasks, or those for which the serial order of stimuli is important.

There is ongoing debate within cognitive psychology about the proper model of serial memory. Leading theories include the chaining model, ordinal theory, and positional theory (Henson, 1999). Recently focus has moved to connectionist neural network-based models (Brown, Preece, & Hulme, 2000; Burgess & Hitch, 1999). Here we present an approach that is reminiscent of the chaining model but avoids some of its drawbacks (see Discussion).

An elegant and parsimonious approach to oscillating working memory models is based on simple modification of Hebbian associative memories with fixed-point attractors to make them oscillatory. For example, Horn, D., Usher, M. (1991) developed a simple oscillatory memory by adding "dynamic thresholds" into Hopfield networks. With this approach, the thresholds used to determine the next activity state of a node are continuously changing such that it becomes increasingly difficult for a node to remain in the same state, and eventually it switches its activity state to the complementary value. When such a network is presented with multiple input stimuli it is found to oscillate between activity states representing these stored memory patterns.

We recently extended the Horn and Usher model to include a weight decay term so that the order of input pattern presentations could affect the network's recall (Winder, Reggia, Weems, & Bunting, 2009). This allows the network to accurately model the recency effect observed in human working memory on running memory span tasks. Stimuli which were presented later in the input sequence were more likely to be successfully stored and recalled by the network when using weight decay.

While the previous version of our model was able to match the position-specific recall rates of human subjects, the order in which the stimuli were recalled by the model was arbitrary. In this paper, we extend our oscillatory weight decay network to enable it to recall inputs in the order presented. The approach is to introduce a second set of temporally asymmetric weights into the model. By doing so we hypothesized that the network would be induced to oscillate between stored memory states in the desired order.

More specifically, we introduce into our simple oscillatory networks for the first time the use of temporally asymmetric Hebbian learning. Adaptation occurs in a fashion inspired by experimental evidence that synaptic efficacy in biological cortex and other brain structures is "temporally asymmetric" (Bi & Poo, 2001; Markram, Lubke, Frotscher, & Sakmann, 1997;

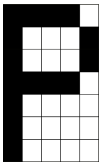


Figure 1: Stimuli to the model consist of 35 binary-valued inputs, conceived of as letters (such as the 'P' shown here) for ease of visualization and interpretation.

Zhang, Tao, Holt, Harris, & Poo, 1998). That is, synapses are strengthened (LTP) if presynaptic activity precedes excitatory post-synaptic potentials by 20-50ms, and weakened (LTD) if the time course is reversed. Our model, when extended in this fashion, not only captures the recency effect of the original model but also now largely retains the sequential order in which the stimuli were presented.

Methods

Model Description

Our model uses a fully connected network of N linear threshold units. Each node takes a binary value $a_i \in \{-1, 1\}$. The stimuli used are in effect arbitrary sets of N bits, though we consider them as being individual letters from A to Z for ease of interpretation. Figure 1 shows an input to a 35 node network interpreted as the letter 'P.'

The operation of the model occurs in two phases: first a temporal sequence of input stimuli are presented and the weight matrices learned according to Eqs. 1 and 2 below, and then the model is allowed to oscillate between states according to Eqs. 3 and 4 for a predetermined total number of iterations. One iteration, or time step, corresponds to asynchronously updating every node once.

There are two sets of connection weights, W and V. Both are $N \times N$ matrices composed of real values, and are initialized to zero before learning. The first of these, W, is the same symmetric weight matrix used in previous version of this model (Winder et al., 2009). The entries of W are updated as each stimulus is presented according to:

$$w_{ij}^t = (1 - k_d) w_{ij}^{t-1} + \frac{1}{N} a_i^t a_j^t (1 - \delta_{ij}) \tag{1}$$

where k_d is a decay rate ($0 \leq k_d < 1$), and δ_{ij} is Kronecker's delta, which ensures that weights on self-connections are fixed at zero. This is, at it's core, the same Hebbian weight change rule used in many previous neural network models. The difference is the addition of the decay term that reduces the influence of older stimuli in favor of more recent ones.

The new element of this model is the incorporation of a second weight matrix, V. The purpose of V is to allow the model to recall stimuli in the same order they were presented. In order to accomplish this, V is trained with a temporally asymmetric learning rule

$$v_{ij}^t = (1 - k_d) v_{ij}^{t-1} + \frac{1}{N} a_i^t a_j^{t-1} \tag{2}$$

inspired by the learning method used in some past neural networks for processing temporal sequences (Schulz & Reggia, 2004). This is similar to the Hebbian learning with decay given in Eq. 1, but it associates the activity of node i during the presentation of stimulus at time t with the activity of all other nodes j during the presentation of the *previous* stimulus at time $t-1$ in the sequence. This introduces a sense of temporal ordering to the weight matrix, potentially making it possible to recall the stimuli in order rather than randomly as was previously done. Note that the decay term is still present, although the Kronecker's delta factor is no longer used as it is desirable for a node's activity to be influenced by its activation state in the previous time steps.

After learning and before recall the network is initially set in a random activity state. It is not necessary to prime the network with a partial or noisy version of any of the input patterns. The calculation of inputs to each node is modified from the prior model to account for both sets of weights. The input to node i at time step t is given as

$$h_i^t = \sum_j \left(\beta_1 w_{ij} a_j^t + \beta_2 v_{ij} a_j^{t-1} \right) - \theta_i^t \tag{3}$$

where the constant coefficients β_1 and β_2 are used to weight the relative contributions of W and V ($0 \leq \beta_1, \beta_2 \leq 1$). As in the previous version of the model, θ_i is a dynamic threshold used to insure that the network oscillates between states rather than coming to rest at a fixed attractor. Its calculation has been simplified from previously, however, with it now being updated according to the following two rules. Every time step, θ_i decays according to $\theta_i^{t+1} = (1 - k_\theta)\theta_i^t$. In any time step in which the state of node i has remained unchanged from the previous time step a factor of $k_w a_i^t$ is also added to θ_i^{t+1}. This moves θ_i in the direction of the activity state of node i, making it more difficult for node i to remain in the same state. Both k_θ and k_w are constants chosen in advance, with $0 < k_\theta < k_w < 1$. We use $k_\theta = 0.09$ and $k_w = 0.175$ in the following computational experiments, though similar values gave qualitatively similar results. Equation 3 has been simplified from the prior model by dropping the K_i biasing term derived from Horn, D., Usher, M. (1991). This was previously used to account for the potentially uneven distribution of active and inactive nodes across potential stimuli and current network state. Computational experiments revealed that it added computational complexity to the model without significant impact on performance.

After the input to each node is calculated, the node's state is updated according to the following rule

$$a_i^t = \begin{cases} +1 & h_i^t > 0 \\ a_i^{t-1} & h_i^t = 0 \\ -1 & h_i^t < 0 \end{cases} \tag{4}$$

This is also a simplification of our earlier model, which used a stochastic updating process. We have found that the deterministic rule given above performs roughly the same with our data set and reduces computational cost.

Measuring Recall

We assess the network's recall by calculating the Hamming distance d_λ between its activity state $\vec{a}$ and $\vec{a}^\lambda$, where $\vec{a}^\lambda$ is a perfect representation of one of the 26 stimuli λ:

$$d_\lambda = \frac{1}{2}\sum_{i=1}^{N}\left|a_i^\lambda - a_i\right| \quad (5)$$

The greater the distance d_λ between $\vec{a}$ and $\vec{a}^\lambda$, the lower the similarity $s_\lambda = 0.85^{d_\lambda}$ will be. A value of $s_\lambda = 1.0$ indicates a perfect match between $\vec{a}$ and $\vec{a}^\lambda$. We call any such time step a "recall peak" for λ. An exponential function was used to define s_λ in order to emphasize the difference between some pairs of inputs with small Hamming distance between them. The choice of 0.85 in the definition of s_λ is essentially arbitrary, chosen because it produced visually reasonable results. Values such as 0.7 or 0.9 work just as well.

In order to compare versions of the model as to whether they successfully recalled the stimuli in the same order as they were presented, we track the transitions from one recall peak to the next and use this to generate a single scalar value. We count the proportion of these peak-to-peak transitions which occur between one stimulus and the stimulus which was presented to the network immediately following. A transition from the fourth-back to the third-back stimulus would be counted as a correct transition, while one from the third to the fourth, or fourth to second, would not. A higher proportion of such correct transitions is indicative of the recall being more well ordered in the sense that the model is cycling through the stimuli it recalls in the same order as they were initially presented. Transitions following the one-back stimulus (i.e. the final stimulus) are ignored because there is no "next" stimulus to correctly transition to.

The recall phase of the model lasts for hundreds of time steps, each one potentially generating the recall of a stimulus. This lengthy series of activity must be distilled into a single ordering of the inputs, in which each unique stimulus appeared no more than once. This is accomplished by consolidating any consecutive time steps in which in the network peaks for the same stimuli. (Neither human subjects nor the model were ever presented with duplicates of the same stimulus, so there was no cause for the model to report seeing the same stimulus repeated.) So, for instance, if a stimuli sequence of "A B C D E" were to result in the network oscillating between the states "B C C C D D E" then the recalled sequence would be taken to be "B C D E," and the second through fifth stimuli would be considered to have been remembered correctly. The requirement to remember the stimuli in the appropriate position is the same as what human subjects are faced with when doing running memory span tasks. Previous versions of the model were not subjected to this requirement; any recall peak for a stimulus was enough for it to be considered correctly stored.

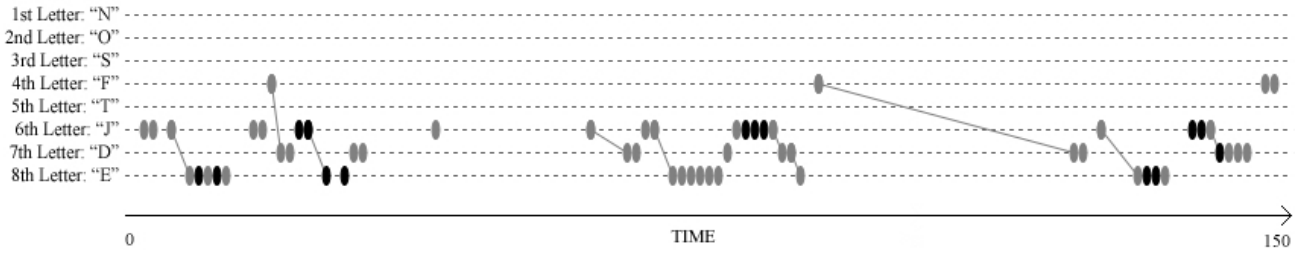
Human Behavioral Data

We used behavioral data that we collected previously (Winder et al., 2009) on a running memory span task for comparison with the model's performance, roughly following the designs of Pollack, Johnson, and Knaff (1959) and Bunting, Cowan, and Saults (2006). Our human experimental data was obtained from 38 adult subjects who were shown a rapidly presented, two per second sequence of 12 to 20 randomly ordered stimuli under computer control, and were asked to remember the most recent six items in the order of their presentation. Subjects indicated the stimuli that they recalled by clicking on a subsequent graphical display of all possible stimuli. Recall was measured by assessing accuracy of recall as a function of stimulus position. A stimulus was counted as accurately recalled only if: 1. it was presented in the retention window (e.g., the last six items, depending on instructions), 2. it was correctly recalled by the participant; and 3. *it was recalled in the same position as it was presented* (counting backwards from the final, most recent stimulus). Any item presented prior to the retention window that was recalled was considered a false positive, as was any item that was not presented at all but which was recalled. Any item from the retention window that was not recalled was considered a miss. Any item that was presented in the retention window, but which was recalled in the incorrect position was also counted as wrong (e.g., if the last six items presented were "1 2 3 4 5 6" and the subject recalled "4 3 2 6 5 1", then only "5" was counted as correct). A total of twelve trials were conducted for the task with each subject requiring roughly 20 minutes per trial; no time restrictions were placed on subject responses. All 38 subjects completed the task.

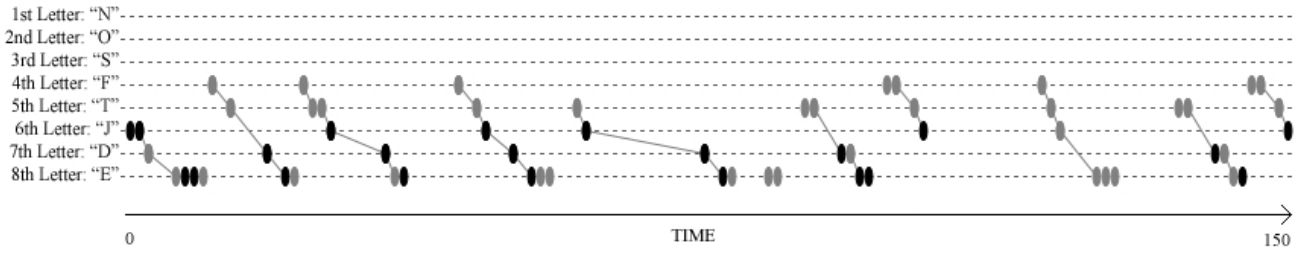
Results

In the previous version of this model (Winder et al., 2009), the network was given an advantage in that it did not have to recall stimuli in the correct temporal sequence for them to be counted as correctly stored. Any network activity pattern during testing with sufficient similarity to an input was considered successfully stored, no matter when that activity pattern occurred. Here we increase the difficulty of the task by requiring the network to also recall stimuli in the correct sequence.

Figure 2 shows an example of the effect that introducing asymmetric weights has on sequential recall. A plot of peaks in similarity for each of the stimuli presented is shown. In Figure 2a without temporally asymmetric weights the ordering of the peaks is largely random, with the network moving between the four stored memory states without regard to their original presentation order. In contrast, Figure 2b with asymmetric weights shows that recalled memory patterns are much more ordered in their progression, with activity tending to proceed from earlier to later input patterns. This ordered retrieval of stored memories is much closer to the human behavioral task described above than was our earlier model.



(a)



(b)

Figure 2: Plot over time of when the values of s reached their peaks for the eight stimuli during an example run of the model. Black marks indicate when s reached the maximum possible value of 1.0 and thus were counted as present, while gray marks indicate when s exceeded 0.8 but did not reach 1.0. The lines between activity peaks indicate transitions that occurred in the same order as the stimuli were presented. The first 150 time steps of the recall phase are shown here. Figure 2(a) is without asymmetric weights ($\beta_1 = 1.0$, $\beta_2 = 0.0$), and Figure 2(b) is with asymmetric weights ($\beta_1 = 0.5$, $\beta_2 = 1.0$). In the former, one can see that the oscillatory states alternate between the four recalled memory patterns for the 4th, 6th, 7th and 8th stimuli (F, J, D and E). Note that these peaks largely occur in an arbitrary order. In the latter case, the network state alternates between the five most recent stimuli, i.e. it has a propensity to recall input stimuli in the same sequence as that in which they were presented.

Table 1: Number of stimuli recalled.

		β_2				
		0.0	0.25	0.5	0.75	1.0
β_1	0.00	–	1.13	1.38	1.46	1.54
	0.25	1.18	1.84	2.01	2.22	2.12
	0.50	1.44	1.91	1.89	2.04	2.26
	0.75	1.72	1.88	1.95	2.02	2.08
	1.00	1.76	1.90	1.93	1.93	1.85

Table 1 shows the number of stimuli successfully stored and recalled by the network for various values of β_1 and β_2 when the network is presented with a sequence of six inputs. In constructing Table 1, five hundred random sequences were used for each simulation, and the network was allowed to oscillate for 250 time steps, with $k_d = .15$. The cell corresponding to $\beta_1 = 1.0,\ \beta_2 = 0.0$ is equivalent to running the network without any influence from the asymmetric weights. The best results were achieved with $\beta_1 = 0.5,\ \beta_2 = 1.0$, which gave a capacity of 2.26 items and with $\beta_1 = 0.25,\ \beta_2 = 0.75$, which gave 2.22 items. For comparison, human subjects had a memory capacity of 2.73 items and our previous model had a capacity of 2.69 (Winder et al., 2009). Note, however, that in the latter case the model's recall was not required to be in the same temporal order as the stimulus.

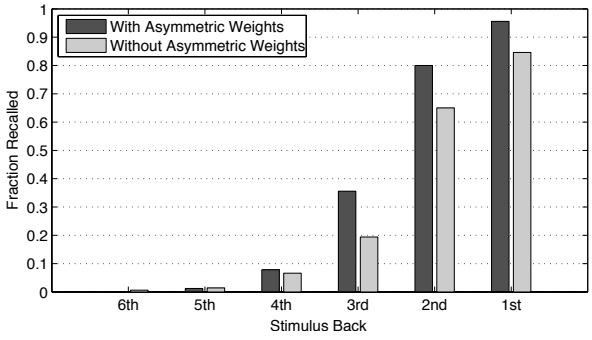


Figure 3: Recall rates for each position with and without temporally asymmetric weights. Five hundred random stimuli sequences were run using a decay rate of $k_d = 0.2$. Networks with asymmetric weights enabled used $\beta_1 = 0.5$, $\beta_2 = 1.0$.

In addition to increasing the total memory capacity relative to baseline ($\beta_2 = 0$), asymmetric weights also increase correct position-specific recall of the network. Figure 3 shows the recall rate at each stimulus position for networks both with and without asymmetric weights. Asymmetrically weighted networks were significantly more likely to retain the three most recent inputs.

Figure 4 shows that the network is capable of modeling

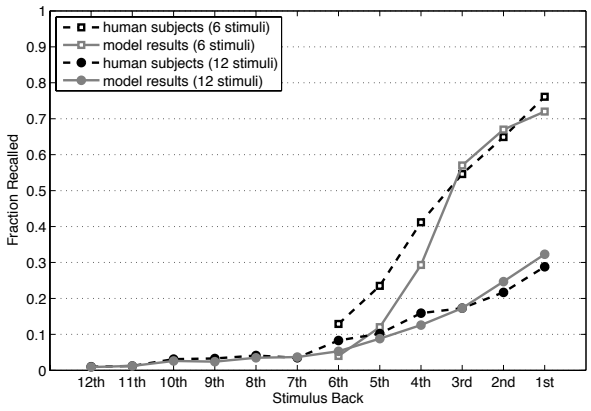


Figure 4: Comparison of the position-specific fraction of recalled simuli by the model and human subjects for both 6-back and 12-back tasks.

human recency behavior on running span tasks when using asymmetric weighting by properly tuning the decay parameter, β_1 and β_2. The model provides close matches for human performance on both 6-back and 12-back running span tasks. (For the former $k_d = 0.05$, $\beta_1 = 0.5$ and $\beta_2 = 1.0$ and for the latter $k_d = 0.075$, $\beta_1 = 0.63$ and $\beta_2 = 0.37$) Fitting data derived from human subjects is a simple matter of tuning these three coefficients, which was accomplished here with a simple iteratively-refined grid search, minimizing the RMSE.

In addition to having higher total and position-specific capacity, asynchronous weighted networks also retained the ordering of the input sequence more effectively. Table 2 gives the proportion of peaks in similarity s that occur in the correct order, using the same parameters as Table 1. That is, those that progress from the fifth-back to the fourth-back, for example. A high proportion of such transitions is achieved when the synchronous weights are ignored completely (i.e., when $\beta_1 = 0$), but note that the number of stimuli recalled by such networks is significantly lower (Table 1). The fewer items stored at all, the easier it becomes to get them into the correct sequence. Limiting the results to those networks which stored more than two of the six stimuli on average, we again find that $\beta_1 = 0.5,\ \beta_2 = 1.0$ gives the best result with 85% of the peaks in s transitioning correctly, compared to between 50 and 56% for the fully temporally symmetric networks, regardless of β_1.

Discussion

This paper extends our earlier simple oscillatory memory model to bias it to produce ordered recall of input sequences. This extension maintains the intrinsic oscillatory nature of the previous model through the use of changing threshold values, and accounts for the ordering of input sequences with weights that include both associated (simultaneous) and temporally asymmetric components. The same results as the previous model, such as the re-creation of our human

Table 2: Portion of peak-to-peak transitions in correct order.

		β_2				
		0.0	0.25	0.5	0.75	1.0
β_1	0.00	–	.81	.86	.93	.87
	0.25	.56	.71	.71	.83	.78
	0.50	.50	.70	.68	.79	.85
	0.75	.56	.65	.68	.75	.78
	1.00	.53	.61	.67	.74	.71

subjects' recency effect, were maintained. The combination of weight decay and temporally asymmetric weight matrices allowed the model to do a much better job of recalling stimuli in the order they were originally seen while simultaneously boosting the number of stimuli successfully stored.

In addition, it was possible to match the model's performance to that from human subjects in two separate tasks (6-back and 12-back), specifically the existence of a prominent recency effect, by tuning only the decay rate and the balance between temporally symmetric and asymmetric inluences. While our earlier model achieved success in matching the behavioral data, that model and human subjects were not being judged on the same scale, as the model did not have to recall stimuli in order while the human subjects did.

This model adds to the growing range of current models of short-term memory. It explains some of the richness of human memory behavior, for instance the recency effect in sequential recall tasks, but does so while remaining parsimonious in its design. There is no need in our model to explicitly specify lateral inhibition in order to provoke competition between stored patterns, such as in Haarman and Usher (2001). In contrast, competition is allowed to arise from the process of Hebbian learning and dynamic thresholds. Further, we do not use different structures for different phases of the memory process. There is no complex architecture of learning and recall units, or active gating structures to explicitly guide the recall process (Frank, Loughry, & O'Reilly, 2001; O'Reilly & Frank, 2006). Rather, a single substrate of identical nodes is all that is needed. The two weight matrices used in the model are also trained with nearly identical rules, and are treated identically during recall. There is also no need in our model to introduce extra layers or nodes to provide temporality of network activity, or to introduce recurrent connections or back-propagation between layers (Botvinick & Plaut, 2006). Multiple patterns, along with their order of appearance, can be stored on the same neural substrate simultaneously.

For the limited range of data considered here, our model did not need to maintain a unified record of the entire sequence of stimuli. Correlations between temporal events can be reconstructed by the network during recall in order to preserve the entire sequence, despite the network only being aware of the immediately preceding stimulus during training. The model's temporal "awareness," such as it is, only exists

in a thin temporal slice. Similarly, during the recall phase, each change of a node's activity is only dependent on the immediately preceding state of the network. Of course, with more complex data additional processing mechanisms would be needed.

While our model can be viewed as a variety of "chaining," it is important to recognize that it does not suffer from one of the principle weaknesses of chaining as a technique for storing sequences. Because of the inherently stochastic nature of the network's activity in the face of rapidly adjusting weight thresholds, there is little harm in being "knocked out" of sequence as the model is able to pick up the trail again. In fact, the initial state of the network is already out of the desired sequence: it is initialized to a random pattern, and not a noisy or partial version of the first pattern in the sequence like with many auto-associative networks. From this initially random state it is able to progress through the stimuli sequence, usually in the correct order, and only occasionally going astray but even then tending back towards the proper ordering. Note that other difficult conditions for chaining, such as duplicate stimuli and repetitions, were not present in the tasks that human subjects performed, and so were left out of our model's training as well.

An obvious direction for future research in this area is the introduction of additional sets of asymmetric weights. Just as we have one set of weights which refer back one time step into the past, it is possible to have a set of weights which refers to earlier activity, perhaps increasing effective sequencing of recall. Such an enhancement may help to deal with some of the difficulties of sequence learning mentioned above, such as repetitions, that were not addressed here. The previous version of the model also only needed a single parameter, k_d to be adjusted in order to match human behavioral data. By introducing β_1 and β_2 we have complicated this slightly. This could be ameliorated by using a single parameter to control the balance between symmetric and temporally asymmetric weights.

Acknowledgments: Supported, in whole or in part, with funding from the United States Government, including NSF award IIS-0753845.

References

Baddeley, A. (2000). Short-term and working memory. In E. Tulving & F. Craik (Eds.), *The oxford handbook of memory.* Oxford Univ. Press.

Bi, G., & Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, *24*, 139-166.

Botvinick, M., & Plaut, D. (2006). Short-term memory for serial order: A recurrent neural network model. *Psychological Review*, *113*(2), 201-233.

Brown, G., Preece, T., & Hulme, C. (2000). Oscillator-based memory for serial order. *Psych. Rev.*, *107*(1), 127-181.

Bunting, M., Cowan, N., & Saults, J. (2006). How does running span work? *Journal of Experimental Psychology: Human Perception and Performance*, *59*(10), 1691–1700.

Burgess, N., & Hitch, G. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review*, *106*(3), 551-581.

Chakravarthy, S., & Ghosh, J. (1996). A complex-valued associative memory for storing patterns as oscillatory states. *Biological Cybernetics*, *75*, 229-238.

Cowan, N. (2001). The magical number 4 in short-term memory. *Behavioral and Brain Sciences*, *24*, 87-185.

Cowan, N., Elliot, E., Saults, J., Morey, C., Mattox, S., Hismjatullina, A., et al. (2005). On the capacity of attention. *Cognitive Psychology*, *51*, 42-100.

Frank, M., Loughry, B., & O'Reilly, R. (2001). Interactions between frontal cortex and basal ganglia in working memory: A computational model. *Cognitive, Affective, and Behavioral Neuroscience*, *1*, 137-160.

Haarman, H., & Usher, M. (2001). Maintenance of semantic information in capacity-limited short-term memory. *Psychonomic Bulletin*, *8*(3), 568-578.

Hasselmo, M., Bodelon, C., & Wyble, B. (2002). Proposed function for hippocampal theta rhythm. *Neural Comp.*, *14*, 793-817.

Henson, R. N. A. (1999). Coding position in short-term memory. *Int'l Journal of Psychology*, *34*(5-6), 403-409.

Horn, D., Usher, M. (1991). Parallel activation of memories in an oscillatory neural network. *Neural Comp.*, *3*, 31–43.

Ingber, L. (1995). Statistical mechanics of neocortical interactions: Constraints on 40-hz models of short term memory based on persistent spiking. *Phys. Review E*, *52*, 4561-4563.

Lisman, J., & Idiart, M. (1995). Storage of 7 ± 2 short-term memories in oscillatory subcycles. *Science*, *267*, 1512-6.

Markram, H., Lubke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, *275*, 213-215.

O'Reilly, R., & Frank, M. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Comp.*, *18*, 283-328.

Pollack, I., Johnson, I., & Knaff, P. (1959). Running memory span. *Journal of Experimental Psychology*, *57*, 137-146.

Raffone, A., & Wolters, G. (2001). A cortical mechanism for binding in visual working memory. *Journal of Cognitive Neuroscience*, *13*, 766-785.

Schulz, R., & Reggia, J. (2004). Temporally asymmetric learning supports sequence processing in multi-winner self-organizing maps. *Neural Comp.*, *16*(3), 535-561.

Wang, D. (1995). Emergent synchrony in locally coupled neural oscillators. *IEEE Trans. Neural Netw.*, *6*, 941-7.

Winder, R., Reggia, J., Weems, S., & Bunting, M. (2009). An oscillatory hebbian network model of short-term memory. *Neural Comp.*, *21*, 741–761.

Zhang, L., Tao, H., Holt, C., Harris, W., & Poo, M. (1998). A critical window for cooperation and competition among developing retinotectal synapses. *Nature*, *395*, 37-44.

Nice Graphs, Good R^2, but Still a Poor Fit? How to be more Sure your Model Explains your Data

Niels Taatgen (n.a.taatgen@rug.nl)
Department of Artificial Intelligence, University of Groningen
Nijenborgh 9, 9747 AG Groningen, Netherlands

Hedderik van Rijn (hedderik@van-rijn.org)
Department of Psychology University of Groningen
Grote Kruisstraat 2/1, 9712 TS Groningen, Netherlands

Abstract

Although widely criticized, R^2 and RMSE are still the most popular measures to report the quality of fit between model and data. Here we present a different way to assess the quality of fit by comparing the fixed effect estimates of mixed-effects models of both the data and the model. We demonstrate the usefulness of this approach on the basis of a time estimation experiment for which two models were constructed. The model that at first seems to have a superior fit turns out to be based on an invalid characterization of the data when scrutinized more carefully, whereas the alternative model provides an accurate characterization.

Keywords: model fitting; time perception; declarative memory; mixed-effect models

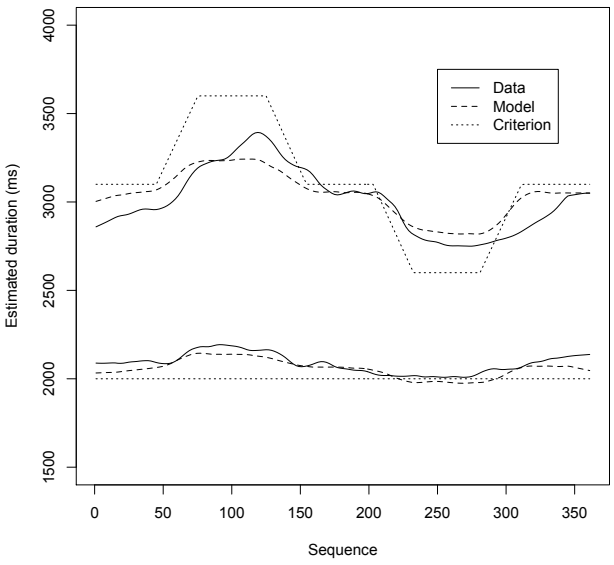
Introduction

One of the unsolved problems in cognitive modeling is how to judge whether a model produces a good fit of the experimental data. Most published papers in which a model is presented try to convince the reader that a fit is good by showing graphs that represent the empirical data along with the model fit. The fit is assumed to be convincing if both graphs are similar. In addition to eyeballing the graphs, statistical measures are often provided to quantify the fit. The most popular measure is R^2, which expresses the correlation between model and data, and some sort of distance measure, like RMSE.

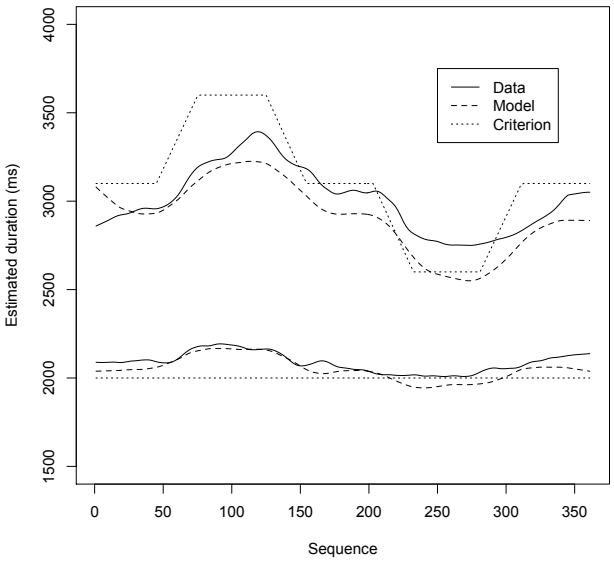
Figure 1 shows an example of two fits between model and data (ignore the "Criterion" curve for now, we will discuss that later). Which of these two models offers a better fit? Neither fit seems to be perfect, but both appear to be reasonable. The following table shows the measures of fit:

Table 1: Measures of fit for the two models in Figure 1

	model A	model B
R^2 upper graph	0.97	0.91
R^2 lower graph	0.81	0.82
RMSE upper graph	178	229
RMSE lower graph	35	46



(a) model A



(b) model B

Figure 1: Two fits between model and data

However, as Schunn and Wallach (2005) pointed out, there is no hard criterion for how high R^2 should be to consider a fit as "good". For RMSE the situation is even less clear, because the measure depends on the measure of the dependent variable. It should just be as low as possible, but there is no standard for what is low enough because the values are dependent on the experiment. Lacking any formal criteria, it is often assumed that the model with higher values for R^2 and lower values for RMSD should be preferred. On the basis of these criteria, Model A should be preferred over Model B, as it outscores it on three of the four measures, and tying it at the fourth. What we will show, though, is that model A is wrong, and model B is reasonably accurate.

Several researchers have criticized the enterprise of fitting models to data. Roberts and Pashler (2000), for example, have pointed out that an ill-constrained model can fit almost any data set. Pitt, Kim, Navarro and Myung (2006) have provided a method to assess the data-fitting capacity of models by examining the partitioning of the space produced by varying all model parameters. If this procedure yields a space with relatively few partitions it means the model makes strong predictions, but if there is a partition for almost any possible outcome, the model is worthless.

Exploration of the parameter space is not always a feasible option, because complex models can take substantial time to run for a single set of parameters, let alone for many combinations. A possible solution to this is to have no free parameters at all, or leave all free parameters at an architectural default, producing so-called "zero-parameter" fits. This is again not always possible, because sometimes parameters have no default value (like some of the parameters in ACT-R's declarative memory), in which case "zero-parameter fits" devolve into "fits with reasonable parameter values". Another issue hidden by the discussion about numerical parameters is the fact that there is considerable freedom in the structural parts of the model (either network topology in neural network models, of symbolic components in a symbolic model). Need another 50 ms to improve the fit? Add a production rule. Need another 200 ms? Add an extra perceptual action. The only way to prevent modelers from wiggling unreported free parameters into their models is to require them to make predictions first and collect data later. The model-data comparison may not always be pretty, but is at least honest (see Taatgen, van Rijn & Anderson, 2007, and Taatgen, Huss, Dickison & Anderson, 2008, for examples).

Apart from the discussion about how a model fit is achieved and how potential alternative fits can be explored, there is the question what kind of measure is a good assessment of a fit. To show that R^2 and RMSD comparisons can deceive, we will first explain our experiment and the goals of the experiment. We will then analyze the data using linear mixed-effect models, and use the same method on the two models. This analysis will provide a better way of comparing models to data, and, although it does not provide absolute criteria, shows convincingly that Model B should be preferred.

Experiment: Memory in Time Perception

To goal of the experiment was to study the role of memory in time perception. In many specialized theories of time perception it is assumed that people are able to represent and store intervals of time in the order of 1 to 60 seconds in memory, without offering any clear theory on the nature of this process. In ACT-R, time perception is modeled using a time estimation module that interacts with the rest of cognition in the same way as other ACT-R modules (Taatgen et al., 2007). The advantage is that ACT-R already has a module for memory, more specifically declarative memory, which can be used to explain memory effects in time perception. We encountered such memory effects in an experiment in which we explored how people estimate partially overlapping time intervals (van Rijn & Taatgen, 2008). In this experiment, subjects had to learn intervals of 2 and 3 seconds, but we noted that the representations of these intervals started to contaminate each other to the extent that some subjects merged both intervals together into a single representation of 2.5 seconds. To study this effect more carefully, we designed a new experiment, of which we will describe one of the conditions here.

Method

In the experiment, subjects learned two intervals, a short one of 2 seconds, and a long one of 3.1 seconds, which they had to reproduce repeatedly, always alternating between the short and the long. Subjects were presented with two circles of the screen, which were gray when they were not active. The circle on the right of the screen was associated with the 2 second interval, while the circle on the left was associated with the 3.1 second interval. During training, one of the circles would change color for a specific duration, and would then turn back to gray. Training consisted of 10 trials, 5 of each duration.

After training, grey circles would again change color to indicate the start of an interval, but now subjects had to press a key to indicate the end of the interval. Subjects received feedback on the accuracy of their produced intervals (we will refer to them as *estimates* from here on): "too short" if they responded earlier than 87.5% of the interval, "too long" if they responded later than 112.5% of the interval, or "correct" otherwise. After training, subjects received 15 warm-up trials of each duration, followed by the experiment proper.

The main manipulation in the experiment is that the criterion for the long interval shifts. For the first 25 estimates of the long interval, the criterion is 3.1 seconds. However, the criterion is then linearly increased to 3.6 seconds over 15 estimates. This means that at some point subjects are told they were too short where they were previously correct. After the shift to 3.6 seconds, the criterion stays at 3.6 seconds, then is decreased back to 3.1 seconds of 15 estimates, stays there for another 25

estimates, then decreases further to 2.6 seconds over 15 trials, stays at 2.6 seconds for 25 trials, increases back to 3.1 seconds over 15 trials and stays there for the remaining 25 estimates. Meanwhile, the criterion for the short interval (remember that short intervals and long intervals are alternated) remains constant at 2 seconds. The "criterion" line in Figure 1 indicates all these shifts. 16 subjects, all students of the University of Groningen, participated in the experiment.

Results

The solid line in Figure 1 shows the mean estimates subjects made for the two intervals. The lines have been smoothed by a Lowess filter (Cleveland, 1981). The results suggest that the two intervals indeed influence each other, given that the changes in criterion for the long interval also impact the estimate of the short interval.

There are (at least) four possible factors that can explain changes in the short interval. One is that the representations of the intervals affect each other directly, i.e., an increase in the internal representation of the longer interval carries over in the internal representation of the short interval. A second explanation is that feedback on the long interval also affects subsequent estimations of the short interval. For example, if we have just produced a long interval, and received the feedback that it was too short, we might unintentionally increase the duration of the short interval that has to be produced next. In addition to the impact of the other interval, previous estimations of the short interval and feedback on those might also impact the next estimate. In order to assess the impact of all these factors, we used mixed-effect models to analyze the data (Baayen, Davidson, & Bates, 2006).

What we did was start out with the most simple regression model to fit the data, and then started adding factors. Each factor adds degrees of freedom to the model, so with each added factor we checked whether improvement in the model was significant with respect to the added degrees of freedom.We started out with the following model, in which the produced short interval is just a constant plus an intercept for each subject:

$$\text{short}_{n,s} = \beta_0 + r_s + \varepsilon_{n,s}$$

So the estimate of short interval n for subject s is equal to constant β_0 plus a random effect for each subject s plus noise. We first start adding the estimates of the previous short intervals. It turns out that including both the previous short interval, and the one before that produce a significant improvement of the model:

$$\text{short}_{n,s} = \beta_0 + \beta_1\text{short}_{n-1,s} + \beta_2\text{short}_{n-2,s} + r_s + \varepsilon_{n,s}$$

Feedback on the previous short estimate also has a significant impact, but not feedback on earlier short estimates:

$$\text{short}_{n,s} = \beta_0 + \beta_1\text{short}_{n-1,s} + \beta_2\text{short}_{n-2,s} + \beta_3\text{short-fb-S}_{n-1,s}\, r_s + \beta_3\text{short-fb-L}_{n-1,s} + r_s + \varepsilon_{n,s}$$

The feedback has two components, because it can be "too short" (short-fb-S) or "too long" (long-fb-L). short-fb-S is equal to 1 if the feedback on the previous trial was "too short", and 0 otherwise. The same is true for long-fb-L and the "too long" feedback. We then added factors associated with the long interval. The estimate of the previous long interval did indeed have a significant impact, but earlier long intervals did not. Finally, we added in the feedback on the earlier long intervals. Here the feedback on the last long interval also led to a significant contribution. Table 2 lists the components and regression values of the final model.

Table 2. Fixed effects in the regression model for the short interval

Fixed Effect	Value of β	*t* value
Intercept	657 ms	4.6
short_{n-1}	0.385	8.3
short_{n-2}	0.085	3.3
short-fb-S_{n-1}	110 ms	3.1
short-fb-L_{n-1}	-208 ms	-6.5
long_{n-1}	0.16	5.1
long-fb-S_{n-1}	92.6 ms	3.2
long-fb-L_{n-1}	-163 ms	-4.2

From this analysis we can conclude that all potential factors contribute to the estimate of the short interval. We can now do the same analysis on the long interval, and determine what its duration depends on. Table 3 shows the final model that came out of that analysis. The general pattern is the same as for the short interval: previous estimates of the long interval and previous feedback on that interval affect the current estimate, even longer back than for the short interval. This is probably due to the fact that the long interval changes. But also the estimate of the previous short interval and the feedback on that interval impact the next long estimate.

Model

The two models of which the results are shown in Figure 1 are in fact instantiations of the same model with different parameter settings. The basis for the model is two modules from the ACT-R theory (Anderson, 2007), but implemented in statistical package R (http://www.r-project.org/). More specifically, we used the time estimation modules (Taatgen, et al., 2007), and the declarative memory module augmented with the blending mechanism (Lebiere, Gonzalez, & Martin, 2007).

Time Estimation

The temporal module of ACT-R measures time in units that start at 100ms, but become gradually longer, creating a nonlinear representation of time. For the purposes of the

Table 3. Fixed effects in the regression model for the long interval

Fixed Effect	Value of β	*t* value
Intercept	695 ms	3.8
$long_{n-1}$	0.34	8.5
$long_{n-2}$	0.16	4.0
$long_{n-3}$	0.12	4.6
$long_{n-4}$	0.05	1.9
long-fb-S_{n-1}	159 ms	4.8
long-fb-L_{n-1}	-118 ms	-2.5
long-fb-S_{n-2}	82.9 ms	2.5
long-fb-L_{n-2}	3.8 ms	0.1
$short_{n-1}$	0.15	2.9
short-fb-S_{n-1}	85 ms	2.1
short-fb-L_{n-1}	-107 ms	-6.5

present model, the nonlinearity is not very important. The temporal module can be given a start signal, which resets the clock, after which an accumulator starts collecting pulses. The short interval of 2 seconds corresponds to approximately 17 pulses, and the long interval of 3.1 seconds to approximately 26 pulses. Noise is added to each pulse, which means that estimates are always approximate. For the purposes of the model, the important aspect of the time estimation module is that it can estimate a particular time interval by translating it into number of pulses, and that it can reproduce a time interval by waiting until a particular number of pulses has been accumulated. The noise produces variability in the estimates that correspond to variability in human time estimation.

Declarative Memory

The assumption of the model is that when a particular time interval has to be produced, the number of pulses representing that interval is retrieved from memory. There is no single representation of a particular interval in memory, but rather a collection of past experiences. Each past experience is represented by a memory chunk, which contains the type of interval (long or short), and a number of pulses. When an interval is retrieved from memory, each chunk receives an activation value on the basis of its age (how old is the experience), and whether it matches the current request:

$$A(t) = \log(t - t_{creation})^{-d} + mismatchpenalty$$

In this equation, $t_{creation}$ is the time the chunk is created, so the activation of a chunk decreases with time. The mismatchpenalty of a chunk is 0 if the request matches the chunk (e.g., we are retrieving a short interval and the chunk represents the short interval), but a negative value in the case of a mismatch (e.g., we try to retrieve a short interval but the chunk represents a long interval).

In standard ACT-R, activation determines the probability of retrieval of a chunk. This means that more recent experiences that match the request have the highest probability to be retrieved. The following equation estimates these probabilities (where *t* is a noise parameter, and the summation is over all candidate chunks):

$$P_i = \frac{e^{A_i/t}}{\sum_j e^{A_j/t}}$$

With the blending mechanism (Lebiere et al., 2007), however, a weighed average of all candidate chunks is retrieved. If we try to retrieve the duration of the short interval, the results will be a blend of all intervals in memory, with the more recent intervals having a higher impact, and the intervals that match the request (short) having a higher impact than the mismatching long intervals. The resulting value can simply be calculated by multiplying the number of pulses in a chunk (V_i) by the probability of retrieval:

$$\text{Result value} = \sum_j P_j V_j$$

In order to determine how many pulses to wait for an interval, the model not only retrieves the representation of the interval, but also feedback received for that interval. For this we use exactly the same mechanism as for the retrieval of the interval. Whenever feedback is received, the model stores this in memory. If the feedback was "correct" it stores the value of 0, if it was "too long" it stores a negative value, and when it is "too short" it stores a positive value (this value is referred to as the feedbackshift, which is a free parameter in the model). Retrieval is done in the same way as the retrieval of the interval itself. This means that the feedback of previous trial for the same duration has the highest impact, but that earlier feedback and feedback for the other duration can also weigh in.

To summarize: if the model has to produce a certain interval, it determines the number of pulses by retrieving a blend of memory representations for that interval. It then retrieves previous feedback for that interval, which is also a blend of earlier feedback. It adds the two together, and waits for that many pulses to produce the interval.

Table 4. Free parameters in model A and B

Parameter	Model A	Model B
Noise parameter *t*	0.25	0.2
Mismatch penalty between short and long for interval retrieval	-1.3	both
Mismatch penalty between short and long for feedback retrieval	-0.8	-0.92
Feedbackshift: how many pulses to add or subtract on the basis of feedback	8	1.8

The free parameters for model A and B were set to the values in Table 4. All other parameters were set to their ACT-R or time estimation module defaults (d=0.5, t_0=100 ms, a=1.02, b = 0.015). The parameters in model A were determined using the procedure that many modelers follow: starting with some initial set of parameters try varying them in order to optimize the fit in terms of R^2 and RMSE. This is typically a satisficing procedure (unless the whole parameter space is explored): model fitting ends as soon as variation of parameters leads to little improvement, and the current fit is decent enough. For model B we used a different method that we will outline later.

So Which is the Better Model?

When we create a cognitive model, it is not our goal to fit a particular data graph, although this may be part of the process, but to explain the phenomena that we are interested in. The statistical analysis has revealed that both the representations of the two intervals and the feedback for the intervals play a role in producing the next interval. It does not tell us what cognitive mechanisms can produce this. The cognitive model does supply a possible answer: a single memory mechanism that has been validated in many other studies can incorporate all factors that play a role in producing the estimate. But is this really true? The graphs in Figure 1 show a good fit, and the R^2's and RMSE also look decent, so what else is there to say?

We can test the impact of the factors that turned up significantly in the data more directly by performing the same analysis on the model outcomes. Statistical significance is not very relevant here, because we can run the model as often as we like. But the model should produce β values that are comparable to the β's found in the data analysis. We therefore ran each model 100 times, and collected the model data in the same format as the human data. This allowed us to fit the same linear regression models. Table 5 shows the results for two models next to the data.

On the basis of this analysis a whole new picture emerges: Model A does not fit the data at all, while Model B provides a very decent fit. The table also reveals the problem of Model A: its representation of the interval is much too stable, as is shown by the estimates for the intercept. In Model A, the intercepts are approximately equal to the actual duration of the interval, and there is hardly any impact of previously produced intervals, either long or short (as evidenced by the low $long_{n-x}$ and $short_{n-x}$ effects). Moreover, Model A's responses to feedback are much stronger than in the data. For example, if Model A receives the "too short" feedback on the short interval, it will respond to this by increasing its next production of that interval by 487 ms, while subjects only increase it by 110 ms. It probably needs such strong values to produce the shifts in estimates of the long interval.

Table 5. Comparison between model and data for the two models

Short interval			
Fixed Effect	β data	β Model A	β Model B
Intercept	657 ms	2157 ms	789 ms
$short_{n-1}$	0.385	0.08	0.356
$short_{n-2}$	0.085	-0.03	0.048
short-fb-S_{n-1}	110 ms	487 ms	170 ms
short-fb-L_{n-1}	-208 ms	-521 ms	-153 ms
$long_{n-1}$	0.16	-0.06	0.15
long-fb-S_{n-1}	92.6 ms	432 ms	125 ms
long-fb-L_{n-1}	-163 ms	-534 ms	-211 ms
Long interval			
Fixed Effect	β data	β model A	β model B
Intercept	695 ms	3162 ms	493 ms
$long_{n-1}$	0.34	0.011	0.22
$long_{n-2}$	0.16	0.012	0.25
$long_{n-3}$	0.12	0.003	0.12
$long_{n-4}$	0.05	0.001	0.09
long-fb-S_{n-1}	159 ms	626 ms	198 ms
long-fb-L_{n-1}	-118 ms	-744 ms	-251 ms
long-fb-S_{n-2}	82.9 ms	60 ms	90 ms
long-fb-L_{n-2}	3.8 ms	-142 ms	-57 ms
$short_{n-1}$	0.15	-0.07	0.18
short-fb-S_{n-1}	85 ms	326 ms	20 ms
short-fb-L_{n-1}	-107 ms	-492 ms	-35 ms

To summarize, Model A might produce a good global model fit, but for the wrong reasons. Model B on the other hand has factor values that are quite similar to those in the data. This means that the same factors that play a significant role in subjects' performance also play approximately the same role in the model's performance. This also means that it is reasonably likely that the model will generalize to other situations in which time intervals have to be stored in memory (see Note at the end).

In fact, the parameter settings for model B were derived by using the factors in the statistical model as an optimization criterion instead of the R^2 and RMSE values. Starting with model A, it was clear the feedbackshift had to be adjusted to reduce the factors associated with feedback. After that, some smaller adjustments led to model B.

Conclusions

Although there are several proposals to improve the assessment of model fit (e.g., Pitt et al., 2006; Weaver, 2008), not all of them are applicable to all types of models, and some of them require intensive additional calculations. The method we showed here is relatively straightforward in comparison, because the same method that is used to analyze the data (which has to be done anyway) can also be used to analyze the model's fit. Although this comparison does not produce a nice and simple single value for the quality of the fit, such a value might be an illusionary

concept anyway. It is never possible to prove that a model has "a 95% probability of being correct". For this it is necessary to know the complete space of possible models/theories, something that is decidedly undecidable.

The nice thing about this analysis is that we can see whether the model produces the effects that we are interested in, and that it produces them in approximately the same order of magnitude. It was even helpful in data fitting itself, because it shows what particular factor is throwing the fit out of balance.

In conclusion, analyzing model fits with mixed-effect models is a promising tool in the modeler's toolbox.

Note

The experiment that we have discussed here had two additional conditions, one in which both intervals remained constant for the duration of the experiment, and one in which they long interval became shorter first and longer later. The model we presented here has not run for those conditions yet. We will do so before the conference and present the results there, and we will keep our fingers crossed that the fit will be good.

Acknowledgements

We would like to thank the participants of the Cognitive Modeling class in Groningen in participating in the discussion of these data, and Stefan Wierda for collecting the data.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford university press.

Cleveland, W. S. (1981) LOWESS: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician*, *35*, 54.

Lebiere, C., Gonzalez, C., & Martin, M. (2007). Instance-based decision making model of repeated binary choice. In *proceedings of the 8th International Conference on Cognitive Modeling*. Ann Arbor, Michigan, USA.

Pitt, M. A., Kim, W., Navarro, D. J., & Myung, J. I. (2006). Global model analysis by parameter space partitioning. *Psychological Review, 113*, 57–83.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review, 107*(2), 358-367.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded Cognition: An Integrated Theory of Concurrent Multitasking. *Psychological Review, 115*(1), 101-130.

Schunn, C. & Wallach, D. (2005). Evaluating goodness-of-fit in comparison of models to data. In: W. Tack (Ed.), *Psychologie der Kognition: Reden and Vorträge anlässlich der Emeritierung von Werner Tack* (pp. 115-154). University of Saarland Press, Saarbrücken, Germany.

Taatgen, N. A., Huss, D., Dickison, D. & Anderson, J. R. (2008). The acquisition of robust and flexible cognitive skills. *Journal of Experimental Psychology: General*, *137*(3), 548-565.

Taatgen, N. A., van Rijn, H., & Anderson, J. (2007). An Integrated Theory of Prospective Time Interval Estimation: The Role of Cognition, Attention, and Learning. *Psychological Review, 114*(3), 577-598.

van Rijn, H. & Taatgen, N.A. (2008). Timing of multiple overlapping time intervals: How many clocks do we have? *Acta Psychologica, 129*(3), 365-375.

Weaver, R. (2008). Parameters, predictions, and evidence in computational modeling: a statistical view informed by ACT-R. *Cognitive Science, 32,* 1349-1375.

The Evolution of a Goal-Directed Exploration Model: Effects of Information Scent and GoBack Utility on Successful Exploration

Leonghwee Teo (teo@cs.cmu.edu)
Bonnie E. John (bej@cs.cmu.edu)
Human-Computer Interaction Institute, Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213 USA

Abstract

We explore the match of a computational information foraging model to participant data on multi-page web search tasks and find its correlation on several important metrics to be too low to be used with confidence in the evaluation of user interface designs. We examine the points of mismatch to inspire changes to the model in how it calculates information scent scores and how it assesses the utility of backing up from a lower-level page to a higher-level page. The outcome is a new model that qualitatively matches participant behavior better than the original model, has utility equations more appealing to "common sense" than the original equations, and significantly improves the correlation between model and participant data on our metrics.

Keywords: ACT-R; CogTool-Explorer; Computational Model; Human-Computer Interaction; Information Foraging

Introduction

Predicting human performance to aid in the design of interactive systems is an important practical use of computational cognitive modeling. Models like SNIF-ACT 2.0 (Fu & Pirolli, 2007) and AutoCWW (Blackmon, Kitajima, & Polson, 2005) focus on predicting user exploration of websites. These models use the common concepts of label-following and information scent (*infoscent*). That is, they posit that the user's choice is partly determined by the semantic similarity between the user's goal and the options presented in the user-interface (UI). Budiu and Pirolli (2007) and Teo and John (2008) began to consider the 2-D spatial layout of the UI when predicting exploration behavior. Budiu and Pirolli (2007) reported a correlation between data and model of R^2 = 0.56 for the number of clicks to success and R^2 = 0.59 for search times in a Degree-Of-Interest (DOI) tree. Teo and John (2008) did not report correlations, but their model successfully predicted the effect of target position in 22 search tasks in a two-column format. This paper furthers this work by considering a multi-page layout of links in a website where previous information is hidden as exploration progresses.

We first describe our metrics and why they are important. We then present the tasks and the operation of a baseline model. After presenting the quantitative performance of the baseline model, we delve into some details of the model's performance to find inspiration as to how to improve the model. Finally, we present the best model found to date and discuss directions for future work.

Our Metrics

Ultimately, a UI designer would want a model to predict the range of human behavior that would be observed in the real world when using the interactive system, on metrics such as number of errors and where they occur, performance time, learning time and what was learned, effects of fatigue, environmental factors, or emotion on performance, and even levels of satisfaction or joy when using the system. No computational model is up to that task at this writing, and more modest metrics are used in current work.

For SNIF-ACT 2.0, Fu and Pirolli (2007) reported the correlation between model and participants on number of clicks on each link (R^2 = 0.69 and 0.91 for two different websites), the correlation for number of go-back actions for all tasks (R^2 = 0.73 and 0.80), and a table of percent of model runs that succeeded on each task juxtaposed with the percent of participants who succeeded on each task (R^2 = 0.98 and 0.94, calculated from Fu and Pirolli, 2007, Figure 13). The first two metrics were for models run under the model-tracing paradigm, that is, at each step the model was allowed to choose its action but was re-set to the participant's action if it did not choose what the participant chose; the last metric was for free-running models. For their free-running model, DOI-ACT, Budiu and Pirolli (2007) did not report percent success because their experiment participants completed all tasks (and the model could run to success on all but 2 of the 16 tasks), but instead reported the correlation between the model and participants for number of clicks to accomplish each task (R^2 = 0.56) and total time for each task (R^2 = 0.59).

We will report similar metrics that are both indicative of model goodness-of-fit and important to UI designers.

1. Correlation between model and participants on the percent of trials succeeding on each task (R^2*%Success*). Percent success is common in user testing to inform UI designers about how successful their users will be with their design, so a high correlation between model and data will allow modeling to provide similar information.
2. Correlation between model and participants on the number of clicks on links to accomplish each task (R^2*ClicksToSuccess*). We eliminated unsuccessful trials because some participants would click two or three links and then do nothing until time ran out whereas others continued to click (as did the model). Also, AutoCWW (Blackmon, et al., 2005) uses this metric.
3. Correlation between model and participants on the percent of trials succeeding without error on each trial (R^2*%ErrorFreeSuccess*). This measure indicates the

model's power to predict which tasks need no improvement and therefore no further design effort.

The Tasks

To test and improve our model, we chose a multi-page layout used in AutoCWW experiments (Toldy, 2009, Experiment 1), shown in Figure 1; Dr. Marilyn Blackmon generously provided the participant log files from 36 exploration tasks performed on this layout. The participants were given a search goal (at the top of each page) and had 130 seconds to complete each task. There were 44 to 46 valid participant trials recorded for each task.

CogTool-Explorer: Mechanisms & Parameters

We start our exploration with CogTool-Explorer (CT-E), developed in the ACT-R cognitive architecture (Anderson, et al., 2004) to account for the effects of 2-column layout on link choice in web search tasks (Teo and John, 2008). CT-E added ACT-R's simulated "eyes" and "hands" to SNIF-ACT 2.0 and interacts with a spatially accurate ACT-R "device model" generated by CogTool (John, Prevas, Salvucci, & Koedinger, 2004), including the position, dimension and text label of every link on a webpage.

Given a text description of a goal and a device model with at least one visible link, CT-E moves its visual attention to a link, visually encodes the text label of the link and evaluates its infoscent relative to the goal. Three ACT-R productions then compete, (1) clicking on the best link so far, (2) reading another link on this page, or (3) going back to the previous page. If CT-E decides to click on the best link it has seen so far, it looks back at that link, moves a virtual mouse pointer over it, and clicks, bringing the next webpage into the model's visual field. If it decides to go back, the previous page is brought into the model's visual field. If it decides to read another link, it moves its visual attention to the next closest link and continues. Of course, this simple see/decide/act cycle is controlled by mechanisms and parameters that can be manipulated to produce the best predictive model possible.

In more detail, CT-E uses ACT-R's "eye" as described in Anderson, et al. (2004) with Salvucci's EMMA model of visual preparation, execution and encoding (Salvucci, 2001), a long-standing implementation within CogTool. A visual search strategy adapted from the Minimal Model of Visual Search (Halverson & Hornof, 2007) guides where to move the eye. The strategy starts in the upper-left corner and proceeds to look at the link closest to the model's current point of visual attention, moderated by its noise function. This strategy will not look at a link more than once on each visit to the web page. Other noise parameters and strategies are possible (e.g., see Budiu and Pirolli, 2007), but as the strategy and noise setting from Halverson and Hornof (2007) produced good results in the two-column tasks in Teo and John (2008), the models in this paper will not vary any aspects of visual processing. Likewise, CT-E uses ACT-

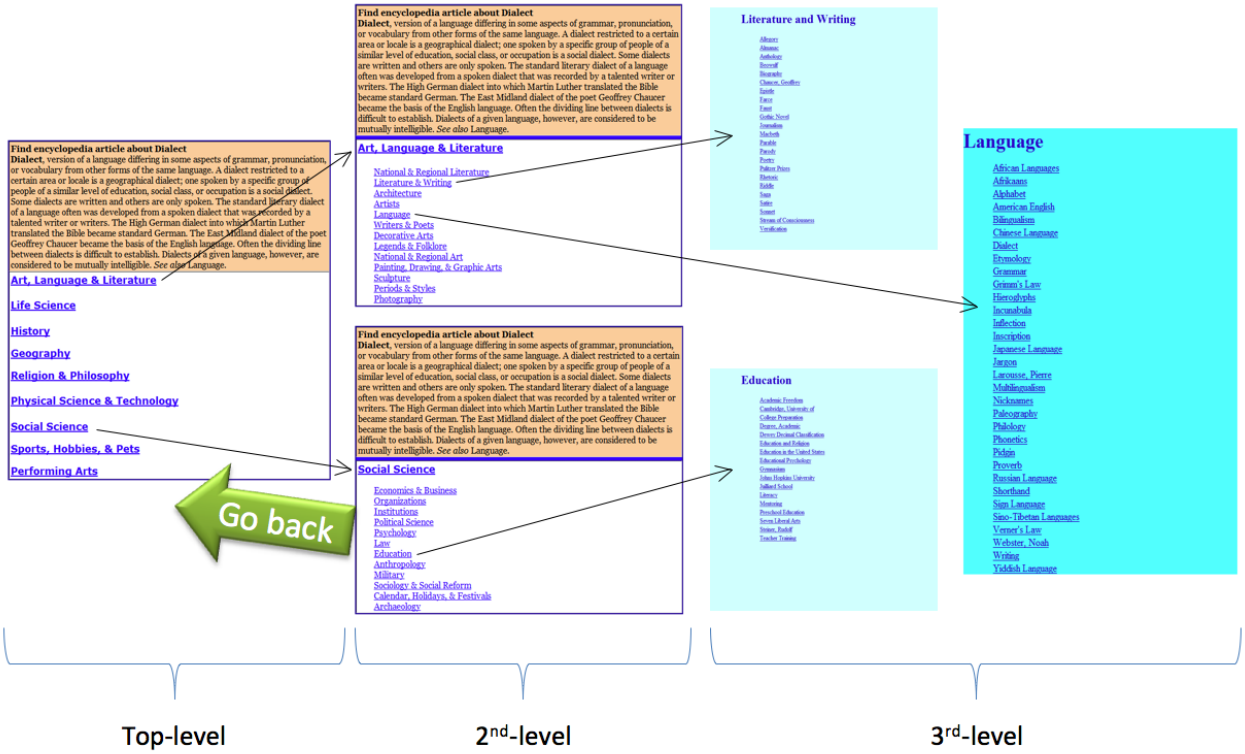


Figure 1: Multi-Page Layout from Toldy (2009). Participants start on the top-level page (leftmost) and on selecting a link, transition to 2nd-level pages. Participants may go back to the top-level page, or may select a link to go to a 3rd-level page. 3rd-level pages explicitly inform participants if they are on the correct path or not.

R's standard "hand," used in many CogTool models, and will retain that mechanism through this paper's exploration.

CT-E's estimation of information scent has used latent semantic analysis (LSA; Landauer, McNamara, Dennis, and Kintsch, 2007) to calculate the semantic relatedness of the search goal to links on the screen. We will continue using LSA throughout this paper, although other estimation procedures are possible (e.g., Fu and Pirolli (2007) and Budiu and Pirolli (2007) used pointwise mutual information). A noise function moderated the infoscent values to reflect the variability a person might display when assessing relatedness (baseline noise = ACT-R default = 1), and a scaling factor of 50 (set by Teo and John, 2008) transforms the infoscent values provided by LSA to the range of values expected by SNIF-ACT 2.0.

CT-E uses the same equations as SNIF-ACT 2.0 to decide which action to take based on what has been seen and evaluated so far, equations which also achieved good results in Teo and John (2008). These equations include two parameters, *k*, a "readiness to satisfice" factor, and the *GoBackCost*. Both of these were set to 5 in Fu and Pirolli (2007), but Teo and John's tasks required a *k* value of 600 to fit the data well, which we will continue to use here. The baseline *GoBackCost* parameter is set to Fu and Pirolli's value of 5.

Finally, when SNIF-ACT 2.0 went back to a page already seen, the link associated with the page backed-up from was marked as having been selected, and SNIF-ACT 2.0 would not select it again (not reported in Fu and Pirolli, 2007, but extracted from the SNIF-ACT 2.0 code). Presumably, since Fu and Pirolli's data come from naturalistic tasks, the link color changed when a link had been selected and thus this "perfect memory" was "in the world". This mechanism is also in CT-E's baseline model.

Performance of the Baseline CT-E Model

We ran the baseline CT-E model until the model runs converged. That is, we ran a set of 44-46 runs of each of the 36 tasks (equal to the number of valid participant trials on each task, for a total of 1649 runs in each set) and calculated the %Success for each task. We then ran an additional set, combined it with the previous set to form a new combined set and compared its values of %Success per task to the previous set's values. If all values were within 1% of each other, we considered the model converged and stopped. If any of the tasks had a %Success value greater than 1% from its counterpart in the previous set, we ran an additional set, combined it with the previous combined set to form a new combined set and compared its values of %Success per task to the previous combined set's values. The baseline model converged after 12 sets (~20,000 runs), with the following calculated values for our metrics and their 95% confidence intervals:

R^2*%Success* = 0.28 (0.21, 0.35)
R^2*ClicksToSuccess* = 0.36 (0.29, 0.43)
R^2*%ErrorFreeSuccess* = 0.44 (0.37, 0.51)

These values are disappointing for UI design because design practice requires far higher confidence in a model's predictions to be a useful alternative to user testing. These values are also substantially lower than the comparable values reported by other SNIF-ACT derivatives, SNIF-ACT 2.0's R^2*%Success* was 0.98 and 0.94 for the two websites modeled (Fu & Pirolli, 2007) and DOI-ACT's R^2*ClicksToSuccess* was 0.56 (Budiu & Pirolli, 2007).

Since the baseline CT-E model used the same utility equations and most of the same parameters as SNIF-ACT 2.0, it is necessary to understand why the R^2*%Success* results are so different. Our first hypothesis is that different data collection processes are to blame. Fu and Pirolli's (2007) data were from participants doing eight tasks on each of two websites, at their leisure, on their own computers. Their participants could abandon the task at will whereas the Toldy's tasks were collected in the lab and participants had 130s to complete each task (Toldy, 2009). Allowing the participants to abandon tasks probably eliminated the most difficult tasks with their higher variability. Not compelled to continue until success, not a single participant in Fu and Pirolli's data succeeded on 4 of their 16 tasks, in contrast to the range seen in Toldy's tasks (average *%Success*=71%, min=13%, max=100%). Since SNIF-ACT 2.0 also failed on these tasks, these four points provided a strong anchor at the origin for their R^2*%Success* value. Another major difference that might have led to better performance is that SNIF-ACT 2.0 used infoscent scores calculated with reference to only the website in the task (E. Chi, personal communication, June 18, 2010), whereas our infoscent scores were calculated with reference to the college-level TASA corpus (from Touchstone Applied Science Associates, Inc.). A corpus comprised of the task website might have produced infoscent scores with less noise (from word sense ambiguity, etc.) that the more general college-level corpus. Finally, simply switching tasks can illuminate deficiencies in any model, which will be the focus of the rest of this paper.

Inspirations for What to Change in the Model

Two glaring deficiencies in the behavior of the baseline model, relative to that of participants, inspired changes in the model. The first is that participants revisit links that they clicked before (13% of their actions) and the model never does. This means that the mechanism in SNIF-ACT 2.0 that perfectly remembers which links have been clicked on and never re-selects them must be changed to allow the possibility of matching the behavior in these data. We cannot tell from the data whether a revisit is a deliberate decision to click on the link a second time or that the participant forgot that link had been clicked (the links in this experiment did not change color when clicked); we chose to model the latter with the following mechanism in our baseline model. Each link is represented as a visual object that has a "status" attribute whose value is set to "chosen" when the link is clicked on by the model and then stored in declarative memory. ACT-R's decay mechanism governs

whether the fact that the link had been chosen will be retrieved when it is next seen and evaluated by this model. We set ACT-R's base level learning activation parameter, :bll, to 0.5 as recommended in the ACT-R tutorial (section 4.3), the retrieval activation threshold to -0.5 as shown in section 4.2, and both the permanent noise, :pas, and the instantaneous noise, :ans, to nil (section 4.5).

The second deficiency in the baseline model is that 22% of the participants' actions involve going back from a page and only 7% of the models' actions do. This behavior is comparable to Fu and Pirolli's 5% go-back actions, which, we believe matched their data because they allowed their participants to abandon tasks instead of going to completion. This calls into question the SNIF-ACT 2.0 mechanisms that govern go-back behavior, that is, both the GoBack utility equation and the *GoBackCost* parameter. We will lower the *GoBackCost* from 5 to 1 to get the exploration started and examine the GoBack utility equation with a more detailed examination of the model behavior.

After making the two fundamental changes motivated by global behavior of the baseline model (call this model *baseline++*), we guided our investigation by examining tasks where participants were least likely to be exploring in a random fashion, i.e., on tasks where participants were most successful. We sorted the 36 tasks by highest *%ErrorFreeSuccess* and then focused on the top four tasks.

The third task in this list, to search for information about pigeons (correct top-level link = "Life Sciences", correct 2nd-level link = "Birds") had infoscent scores that were all very low and not widely distributed for the top-level headings. Budiu and Pirolli (2007) discuss this problem as well; misleading and/or non-discriminating infoscent scores will plague any model and we did not consider this task further for inspiration about what to change. However, the other three tasks inspired three ways to change the baseline++ model.

Refinement of Infoscent Values for Top-level links

The topmost task was to search for information about ferns and its correct top-level link was "Life Sciences". The 46 participants only selected other top-level links 8% of the time and but went back from those 2nd-level pages to select "Life Science" and then "Plants" (in all but 2 cases) to complete the task. In contrast, the baseline++ model selected other top-level links about 70% of the time before selecting "Life Sciences", and on some model runs it never selected "Life Sciences" and failed the task.

One possible explanation for the model behavior was that it did not look at "Life Science" before deciding to select a link on the top-level page. When we examined the details of the model runs, this was not the case, as the model runs did see "Life Science" before selecting a link in over 95% of first-visits to the top-level page. A second possible explanation was that the model looked at too many links and saw other higher infoscent links before selecting a link on the top-level page. This also was not the case because, in all model runs up to the point where it finished looking at "Life Science", if we forced the model to choose the best link so far, it would have selected "Life Science" in over 60% of the runs. A third possible explanation lies in the infoscent values used by the model.

Given a particular goal, the baseline models followed AutoCWW (Blackmon, et al., 2005) by using LSA to compute an infoscent value for each link, based on the cosine value between two vectors, one representing the words in the goal description and the other the words in the link text. To approximate how a reader elaborates and comprehends the link text in relation to his or her background knowledge, AutoCWW adds all the terms from the LSA corpus that have a minimum cosine of 0.5 with the raw text and a minimum word frequency of 50 to the raw link text before using LSA. Kitajima, Blackmon and Polson (2005) explained that "elaborated link labels generally produce more accurate estimates of semantic similarity (LSA cosine values)." Our baseline model used the same method, thus, for the link "Life Science", the words *"science sciences biology scientific geology physics life biologist physicists"* were added and then submitted to LSA to compute the infoscent value.

AutoCWW uses a further elaboration method motivated by UI layouts with links grouped into regions labeled with a heading. Kitajima et al. (2005) explained that "readers scan headings and subheadings to grasp the top-level organization or general structure of the text". To represent a region, AutoCWW first elaborates the heading text as described in the previous paragraph, and then adds all the text and their elaborations from links in the same region. The baseline model did not use this elaboration method for top-level links because their subordinate links appeared on 2nd-level pages, different from Kitajima et al.'s assumption. However, participants did practice trials on the same multi-page layout as the actual trials, and perform all 36 test trials on the same layout. Therefore, we would expect that this experience would influence how participants assessed infoscent of the top-level link. This reasoning motivated our first refinement to the baseline++ model to better represent these participants: for the infoscent of a top-level link, we elaborate the top-level link and then add the text from all links in the corresponding 2nd-level page. While this refinement is similar to AutoCWW's procedure, the justifications are different. This refinement is also in line with Budiu and Pirolli's (2007) use of category-based scent, but approximates their human-generated categories with an automated process.

Refinement of Mean Infoscent of Previous Page

The second task on our list was to search for information about the Niagara River. The baseline++ model selected the correct link "Geography" on the top-level page, but went back from the 2nd-level "Geography" page over 60% of the time, while participants never did. To investigate, we looked at how the model decided to go back. Recall that like SNIF-ACT 2.0, after looking at and assessing the infoscent of a link, the baseline CT-E models choose between reading

another link, selecting the best link seen so far, or going back to the previous page using utility functions. The utility functions of reading another link and selecting the best link so far have both strong theoretical support (Fu & Pirolli, 2007) and empirical support from several studies that did not use or emphasize go-back behavior (Fu & Pirolli, 2007 and Teo & John, 2008). However, the utility function for going back has less support and was therefore a focus of our attention. From SNIF-ACT 2.0, the baseline CT-E models used the following GoBack utility equation.

$$\begin{aligned} \text{Utility}_{\text{GoBack}} &= \text{MIS(links assessed on previous page)} \\ &\quad - \text{MIS(links assessed on current page)} \\ &\quad - \text{GoBackCost} \\ &\textit{where MIS is Mean Information Scent} \end{aligned} \qquad \text{[Eq. 1]}$$

The infoscent values for the nine top-level links are sensible: the correct link, "Geography", has the highest LSA value by an order of magnitude. After selecting the top-level link with the highest infoscent and visiting the corresponding 2nd-level page, Eq. 1 includes "Geography's" high scent in its first operand, which attracted the model back to the top-level page. This behavior violates common sense; since the model had just selected the best top-level link to visit its 2nd-level page, it should not be pulled back to the previous page by the infoscent of the selected link. This reasoning inspired another refinement to the baseline++ model, changing Eq. 1 to Eq. 2:

$$\begin{aligned} \text{Utility}_{\text{GoBack}} &= \text{MIS(links assessed on previous page} \\ &\quad \textbf{excluding the selected link)} \\ &\quad - \text{MIS(links assessed on current page)} \\ &\quad - \text{GoBackCost} \\ &\textit{where MIS is Mean Information Scent} \end{aligned} \qquad \text{[Eq. 2]}$$

Refinement of Mean Infoscent of Current Page

The last task on our list of four was to find information about the Hubble Space Telescope. While both participants and model in this task selected the correct link "Physical Science & Technology" on the top-level page, the model went back from the corresponding 2nd-level page 50% of the time, but participants never did. Inspection of the model runs in the Hubble task revealed a different problem from that in the Niagara River task, however. After selecting the link with the highest infoscent and visiting the corresponding 2nd-level page, if the first link the model saw on that page had very low infoscent, the GoBack utility would be high because the value of the second operand would be low. This behavior also violates common sense; since the model had just selected the best link on the top-level page because it looked promising, the model should carry that confidence into the next page and should not immediately go back just because the first link it saw on the 2nd-level page did not relate to the task goal. This reasoning inspired our last refinement to the baseline++ model, changing Eq. 2 to Eq. 3:

$$\begin{aligned} \text{Utility}_{\text{GoBack}} &= \text{MIS(links assessed on previous page} \\ &\quad \textbf{excluding the selected link)} \\ &\quad - \text{MIS(links assessed on current page)} \\ &\quad \textbf{including the selected link)} \\ &\quad - \text{GoBackCost} \\ &\textit{where MIS is Mean Information Scent} \end{aligned} \qquad \text{[Eq. 3]}$$

This change has a nice symmetry with the previous change, carrying along the "confidence" inspired by the high infoscent top-level link. If the selected link's infoscent score is very high compared to the other top-level links, those other top-level links alone will not exert much pull to go back. If the selected link's infoscent score is high relative to the first few links it sees on the 2nd-level page the model will not go back until it "loses confidence" by seeing several low infoscent links, thereby diluting the effect of the high infoscent link that led the model to this page.

We ran one set of many preliminary models to get a feel for the contributions of these changes. The combination of all changes described here seemed to be the best model.

Performance of the Best Model So Far

With all the changes described above combined, we ran the model to convergence (10 sets, a total of 16490 runs), and attained the following calculated values for our metrics and

Table 1. Summary of Results. Gray shading indicates mechanism and parameters that did not change.

Mechanism, Parameter, or Metric	Baseline Model	Best Model So Far
Visual processes	ACT-R + Salvucci, 2001 + Halverson & Hornoff, 2007 [2]	No change
Manual processes	ACT-R [2]	No change
Information Scent Process		
Heading-level input	link labels	link labels + lower link labels
Link-level input	link labels	No change
Decision Process		
Click best link utility eq	SNIF-ACT2.0[1]	No change
k (readiness to satisfice)	600[2]	No change
Read next link utility eq	SNIF-ACT2.0[1]	No change
GoBack utility equation	SNIF-ACT2.0: Eq. 1[1]	Improved here Eq. 3
GoBackCost	5[1]	1
Memory of selected links	Perfect[1]	Imperfect :bll = 0.5 :rt = -0.5 :ans = nil :pas = nil
Metrics		
R^2%*Success*	0.28 (0.21, 0.35)	0.72 (0.66, 0.76)
R^2*ClicksToSuccess*	0.36 (0.29, 0.43)	0.66 (0.60, 0.71)
R^2%*ErrorFreeSuccess*	0.44 (0.37, 0.51)	0.82 (0.79, 0.85)

[1] from Fu & Pirolli, 2007
[2] from Teo & John, 2008

their 95% confidence intervals (Table 1):

R^2*%Success* = 0.72 (0.66, 0.76)

R^2*ClicksToSuccess* = 0.66 (0.60, 0.71)

R^2*%ErrorFreeSuccess* = 0.82 (0.79, 0.85)

Discussion and Future Work

The improved model presented above made large and significant improvements on all our metrics over the baseline model coming into this investigation. R^2*%Success* more than doubled and the other two metrics increased by more than 50%. Although there is room for improvement, these values are in the range where UI designers could use them to identify the tasks at the extremes. That is, this analysis identifies which tasks are sufficiently supported by the interface that effort can be diverted to other areas and which tasks are in most need of attention.

Future work will take at least two paths. First we must systematically explore the benefits of the model mechanisms and parameters described in this paper. We have presented only the conjunction of these elements, with a single set of parameters, but we will examine the mechanisms' individual and pairwise effects on model performance and explore the parameter space before moving on to other UI layouts and tasks.

Second, we should reconsider the metrics and how to use them. Although we believe the metrics presented here are both meaningful for goodness of fit and useful for UI design, other metrics should be considered. For example, Fu and Pirolli (2007) reported the correlation between the number of go-back actions by the model and participants; how might this help inform model improvements or design? As a second example, consider root mean square error (RMS error), a standard metric for quantifying the difference between the values estimated by a model and what is observed in empirical trials. UI designers often need to know absolute quantities when making decisions about design and development effort and cost trade-offs. Thus, a low RMS error would be as valuable as a high correlation (the RMS error did reduce for each metric with our improved model, but are not yet <20% which is desirable for UI design practice). In addition, we need to understand how to combine or trade-off metrics against one another, as it is unlikely that model exploration will produce the most desirable levels of all metrics at once.

In the meantime, AutoCWW has shown it could be used to improve the design of website links with only 54% of the variance explained for *ClicksToSuccess* (Blackmon, et al., 2005) and this improved version of CogTool-Explorer exceeds that level. If these results can be shown to extend beyond simple web search tasks, to other layouts, types of interfaces, and tasks, CogTool-Explorer will be well on its way to being a useful tool for design.

Acknowledgments

The authors thank the anonymous reviewers whose probing questions improved the science reported in this paper and Dr. Marilyn Blackmon for sharing the experiment data. This research was supported in part by funds from IBM, NASA, Boeing, NEC, PARC, DSO, ONR, N00014-03-1-0086. The views and conclusions in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of IBM, NASA, Boeing, NEC, PARC, DSO, ONR, or the U.S. Government.

References

ACT-R 6.0 Tutorial (June, 2010) Available for download at http://act-r.psy.cmu.edu/actr6/units.zip, June 13, 2010.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review. 111*, 4, 1036-1060.

Blackmon, M. H., Kitajima, M., & Polson, P. G. (2005). Tool for accurately predicting website navigation problems, non-problems, problem severity, and effectiveness of repairs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '05. ACM, New York, NY, 31-40.

Budiu, R. & Pirolli, P. (2007), Modeling navigation in degree-of-interest trees. In N.A. Taatgen & H. van Rijn (Eds.), *Proceedings of the 31th Annual Conference of the Cognitive Science Society* (pp. 845-850). Austin, TX: Cognitive Science Society.

Fu, W.-T., & Pirolli, P. (2007). SNIF-ACT: A cognitive model of user navigation on the World Wide Web. *Human-Computer* Interaction*, 22,* 355-412.

Halverson, T. & Hornof, A. J. (2007). A minimal model for predicting visual search in human-computer interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. ACM, New York, NY, 431-434.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '04. ACM, New York, NY, 455-462.

Kitajima, M., Blackmon, M. H. & Polson, P. G. (2005). Cognitive architecture for website design and usability evaluation: Comprehension and information scent in performing by exploration. *HCI International 2005*.

Landauer,T. K., McNamara, D. S., Dennis, S., & Kintsch W. (Eds). (2007). *Handbook of Latent Semantic Analysis*.

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, *1*, 201-220.

Teo, L., & John, B. E. (2008). Towards a tool for predicting goal-directed exploratory behavior, *Proceedings of the Human Factors and Ergonomics Society 52nd Annual Meeting* (pp. 950-954). Santa Monica, CA: Human Factors and Ergonomics Society.

Toldy, M. E. (2009) The Impact of Working Memory Limitations and Distributed Cognition on Solving Search Problems on Complex Imformational Websites. Unpublished Doctoral Dissertation, University of Colorado – Boulder, Department of Psychology.

A Computational Model of Second-Order Social Reasoning

Leendert van Maanen (leendert@ai.rug.nl) & Rineke Verbrugge (rineke@ai.rug.nl)
Department of Artificial Intelligence, University of Groningen
P.O. Box 407, 9700 AK Groningen, The Netherlands

Abstract

This paper presents the first computational cognitive model of second-order social reasoning. The model uses a decision tree strategy to reason about the opponent's behavior. We hypothesize that a decision tree strategy requires (1) declarative memory, and (2) working memory. Declarative memory is required to retrieve successive reasoning steps, while working memory is required to temporarily store these reasoning steps while the next step is retrieved from memory. The model fit on data from a social reasoning game supports the validity of the model. This initial result leads to an explicit prediction for an experiment in which the reasoning game is combined with another task that requires the same cognitive resources as hypothesized by the model. This work is a first step towards understanding higher-order social reasoning from a cognitive modeling perspective.

Keywords: reasoning; theory of mind**;** cognitive models; ACT-R

Introduction

What is social reasoning?

The ability to successfully interact with others requires knowledge on how your actions are going be interpreted by others. Additionally, successful interaction requires the ability to reason about the actions that other people might take to respond to, or even to anticipate, your own actions. (Verbrugge, 2009). A term that is often used in connection with this ability is *theory of mind* (Premack & Woodruff, 1978). In this paper we will present a computational cognitive model of second-order theory of mind, calling the process *second-order social reasoning*.

Contrary to the case of first-order mental state attributions such as "she *plans* to move her queen", second-order social reasoning requires the ability to attribute mental states *about* mental states to others, as in "she *believes* that I *intend* to sacrifice my horse" (Perner & Wimmer, 1985). In higher-order social reasoning, this ability is recursively applied for successful behavior. The cognitive model presented in this paper will be the first that explicitly addresses higher-order social reasoning. We will present a theory on how people reason in second-order social reasoning games, as well as explicit predictions on how behavior changes if the task is made more complex.

Second-order social reasoning has often been studied by use of simple strategic games in which success is only warranted if the players successfully anticipate each other's moves. A very simple example of such a game is tic-tac-toe (also known as noughts-and-crosses), in which each player has all information available on the playing board, and players have to take into account what the optimal move is for the opponent (that is, games of perfect information, Osborne & Rubinstein, 1994). A more complex example is Cluedo (Van Ditmarsch, 2002) in which not all information is known to each player, and players also have to reason about what information they will provide to their opponents by making a move, in addition to reasoning about optimal moves, for example, "I don't *want* Alice to *know* that I *know* that she has the ace of hearts". In this paper, we will focus on a simpler game called Marble Drop in which all information about the current game state is known. Marble Drop is equivalent to the well-known centipede game (Rosenthal, 1981) and will be discussed in detail in later sections.

What are important questions in social reasoning?

Two issues stand out in studying social reasoning. The first relates to human performance on games such as Marble Drop. Up to this point we have described behavior as "optimal" or "rational", but it turns out that humans perform significantly suboptimally on these games as the complexity increases (Flobbe, Verbrugge, Hendriks, & Krämer, 2008; Hedden & Zhang, 2002). Flobbe et al. for example found that participants in a centipede game only correctly perform 75.5% of second-order games, whereas they are near-perfect on the first-order games (97%).

The second issue relates to the role of memory in reasoning tasks. Taking the perspective of others about your own mental states and then incorporating that knowledge in your own reasoning must require some form of working memory. In this paper, we will present the first computational model that explicitly addresses both issues.

After a brief overview of other models of social reasoning, we will introduce our model. Then we will present the model fit on relevant data and we will discuss how this model can contribute new insights in the understanding of social reasoning.

Formal models of social cognition

Social reasoning has been formally studied from a number of perspectives. These perspectives differ in the amount of cognitive validity that is considered. One perspective is to study social cognition as an interactive game (Camerer, 2003). This game-theoretic perspective assumes that people are rational agents, optimizing their gain by applying strategic reasoning. However, many experiments have shown that people are not completely rational in this sense. For example, McKelvey and Palfrey (1992) have shown that in a traditional centipede game participants do not behave rationally. In this version of the game, the payoffs are distributed in such a way that the optimal strategy is to always end the game at the first move (i.e., Nash

equilibrium, Nash, 1951). However, in McKelvey and Palfrey's experiment participants continued the game for some rounds before ending it. One interpretation of this result is that the game-theoretic perspective fails to take into account the reasoning abilities of participants. That is, due to cognitive constraints such as working memory capacity, participants may be unable to perform optimal strategic reasoning, even if in principle they are willing to do so.

A different perspective, that focuses on cognitive validity in developing formal models, is that of a cognitive architecture (Anderson, 2007; Newell, 1990). Cognitive models developed within this framework aim to explain certain aspects of cognition by assuming only general cognitive principles. However, the current cognitive models that describe social interactions do not take second-order reasoning into account. For example, cognitive models of simple games exist in which it is important to know the opponent's behavior (e.g., Lebiere & West, 1999; West, Lebiere, & Bothell, 2006). These cognitive models demonstrate that declarative memory is important in playing strategically. In the current work however, we are less interested in how people adapt their strategy to an opposing strategy, but rather we are studying the cognitive limitations of explicit second-order reasoning. Related to this, Hendriks and colleagues (e.g., Hendriks, Van Rijn, & Valkenier, 2007; Van Rij, Van Rijn, & Hendriks, in press) have studied the development of first-order theory-of-mind in language using computational cognitive modeling.

An ACT-R model of social reasoning

To provide a full model of second-order social reasoning, we implemented our model in the cognitive architecture ACT-R (Anderson, 2007). ACT-R aspires to explain all of cognition using one theoretical framework. To achieve this, the heart of ACT-R consists of a procedural memory system, which contains condition-action pairs known as production rules. Besides the procedural module, ACT-R has designated modules for specific types of information. For example, the visual module processes visual information, whereas the declarative memory module processes declarative or factual information. Each module has a buffer that may contain one unit of information (a chunk). If the current contents of all buffers in the system matches the conditions of a particular production rule, that rule fires and its actions are executed. Each action may refer to an operation in one of the modules.

This general layout of the cognitive system enables the development of models in which different kinds of information can be processed at the same time, while each module can only process one unit of information at a time. Based on this feature, ACT-R predicts specific interference effects if different aspects of a task require the same cognitive resource at the same time (e.g., Borst, Taatgen, & Van Rijn, 2010; Van Maanen & Van Rijn, 2010; Van Maanen, Van Rijn, & Borst, 2009). In the discussion section of the current paper we will use this feature of the architecture to make explicit predictions for a particular social reasoning task.

Two modules of ACT-R deserve extra attention in the light of our model of second-order social reasoning: the declarative memory module and the problem state module. The declarative memory module retrieves information from long-term memory, called chunks. Each chunk in memory is represented by an activation value that represents the likelihood that that item can be retrieved. If the activation value drops below a certain minimal value (the retrieval threshold), the related information is no longer accessible. In that case, the system will report a retrieval failure after a constant time factor. If the activation value is above the retrieval threshold, the information is accessible. However, the time needed to retrieve it from memory depends on how active the item actually is. The more active, the faster the retrieval will be. Connected to the declarative memory module is a retrieval buffer, which may contain one (retrieved) item at a time. If another item is retrieved, it is stored in the retrieval buffer, with the previous item being pushed back to long-term memory.

The problem state module (sometimes referred to as the imaginal module) contains a buffer in which information can be temporarily stored. Typically, this information contains a subsolution to the problem at hand. In the case of a social reasoning task, this may be the outcome of a reasoning step that will be relevant in subsequent reasoning. Storing information in the problem state buffer is associated with a time cost (typically 200ms). The model that we present in this paper relies on the combination of the declarative module and the problem state buffer. That is, the model retrieves relevant information from memory and moves that information to the problem state buffer if new information is retrieved from memory that needs to be stored in the retrieval buffer.

Marble Drop game

To study the reasoning processes that are involved in social reasoning, we developed a cognitive model of a reasoning game in which in order to play optimally the players have to anticipate each other's moves. The particular game that was analyzed and modeled is a variant of the centipede game called Marble Drop (Meijering, Van Maanen, Van Rijn, & Verbrugge, 2010).

Marble Drop is a marble run game containing trapdoors (Figure 1). Players take turns in deciding whether to open one trapdoor or the other. In each turn, opening one trapdoor leads to the end of the game, whereas opening the other trapdoor means that the game continues to the next bin on the right and the opponent may choose which trapdoor to open. If a player decides to end the game, both players receive the credits that are associated with that stage of the game. If a player decides to continue the game, the players traverse to a new stage with which new credits are associated. Because all credits are known in advance, both players can reason about their opponent's possible moves further on in the game. The players can do this by applying

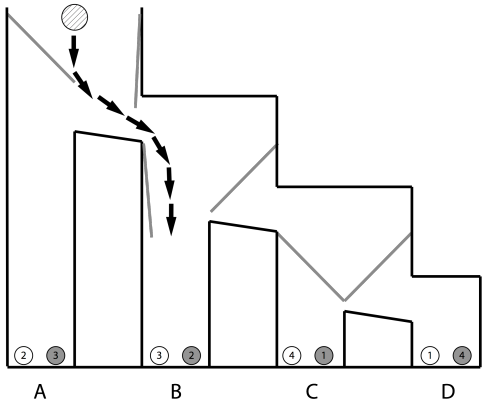


Figure 1. The interface of a second-order Marble Drop game. Color shades of the marbles in the experiment are represented by numbers.

backward induction (Van der Hoek & Verbrugge, 2002; Verbrugge & Mol, 2008). For example, a player can reason that his opponent wants the highest payoff in bins C and D. As a result the player knows the maximal payoff that he can get from bins C and D, and can then compare that information to his own payoff in bin B. If it is possible in a particular game for a player to behave optimally by directly predicting its opponent's actions, we refer to this game as being *first-order*. In a *second-order* game it is necessary to predict the opponent's predictions of ones own actions in order to behave optimally. In principle, Marble Drop games could be developed for third-order or even higher-order games.

The Model

The model follows a backward induction strategy to predict the opponent's moves further on in the game. Hedden and Zhang (2002) provide a decision tree analysis of this process for their matrix version of the game.[1] The model has knowledge on how to solve Marble Drop games for all possible distributions of payoffs over the bins of the marble run game. That is, the model stores chunks containing information on which payoffs to compare at each step. In addition, chunks representing the magnitudes of the payoff shades are stored in declarative memory, as well as chunks representing the location of the payoffs on the screen. Finally, chunks representing ordinal information are stored in declarative memory. This means that the model contains knowledge on the relative magnitudes of each combination of payoff values.

A model run starts with the initial comparison of two payoff values (Figure 2). For second-order games, that initial comparison is always a comparison between the player's own payoffs in Bins C and D. First, it retrieves from declarative memory where the first payoff is located on the screen (Bin D in Figure 1). If it retrieves that knowledge, the model attends Bin D and tries to retrieve the magnitude of the observed payoff. At the same time, the model stores the current comparison in the problem state buffer, to free the retrieval buffer for the upcoming payoff information.

Because in the experiment the payoffs are represented by shaded marbles, the model has to retrieve the value corresponding to the observed shade. Next, the model retrieves the location information for the other payoff value that is part of the current comparison. Again to free the retrieval buffer, the payoff value of the first payoff is stored in the problem state buffer. The payoff is attended and the corresponding value is retrieved from memory. Finally, the two values are compared by trying to retrieve a chunk with ordinal information from memory. Based on the outcome of this retrieval the model now retrieves a new payoff comparison. For example (Figure 1), if the value in bin D was smaller than the value in bin B, the model attends the payoff in bin B, and compares that with the payoff in bin A. If the value in bin D was larger than the value in bin B, then the model attends the opponent's payoff in bin D, and compares that with the opponent's payoff in bin C. The model continues to compare payoffs following the decision tree (Hedden & Zhang, 2002) until it reaches the bottom of the tree. There, it decides its action based on the final comparison.

Model fit The model was tested against data from a Marble Drop task (Meijering et al., 2010). In the experiment the participants were asked to solve zero-order, first-order, and second-order Marble Drop problems. In all these conditions, participants were instructed to indicate the optimal *first*

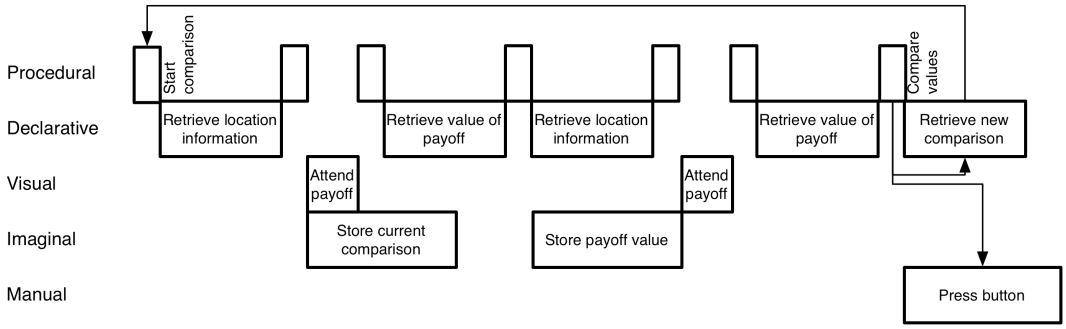


Figure 2. Flow chart of the model activity in ACT-R modules. The width of each box denotes the duration of each stage. Arrows indicate possible next actions.

[1] an analysis that shows the logical equivalence of these games can be found at http://www.ai.rug.nl/~leendert/Equivalence.pdf

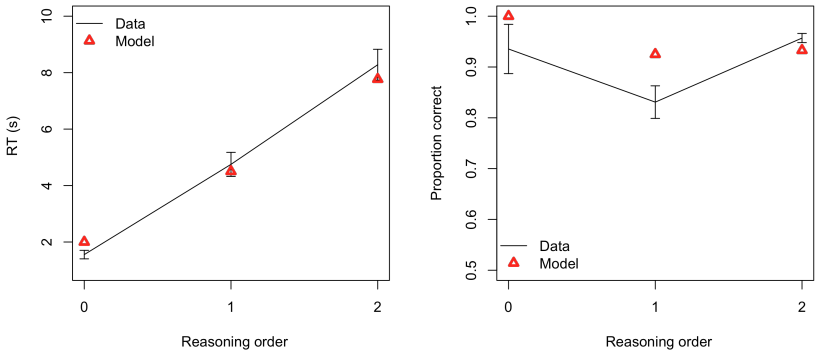


Figure 3. Model fit to data from Meijering et al. (2010). Left: Response time, Right: Accuracy.

move as quickly as possible. That is, even in second-order games participants had to make only one choice. However, because the opponent always played rationally (and the participants were informed of this), there was always only one optimal choice.

Figure 3 presents the model fit on both response times and accuracy of the first moves. The fit on the response times is very good ($R^2 = 1.0$; RMSE = 0.42 s). The fit on the accuracy data is slightly less (RMSE = 0.067, $R^2 = 0.2$), but this may be attributed to lack of data, making the estimated means less reliable.[2]

As the order of the Marble Drop reasoning problems increases, the model requires more time to respond. This is because more comparisons have to be made, and therefore more information has to be retrieved from declarative memory and stored in the problem state buffer. These steps take time, increasing the response time for higher-order reasoning problems. Because of the similar behavioral patterns between model and data, this study supports the view that participants in this task follow the same reasoning steps as the model does. That is, participants in a social reasoning game follow a decision tree to make the correct decision.

Discussion & Predictions

First model of second-order social reasoning

The ACT-R model of second-order social reasoning described in this paper is the first cognitive model to account for second-order social reasoning. Other cognitive models in the field of social reasoning have either not explicitly addressed orders of reasoning (e.g., Lebiere & West, 1999; West et al., 2006), or have focused on first-order reasoning only (e.g., Hendriks et al., 2007; Van Rij et al., in press).

Because the model is based on Hedden and Zhang's (2002) decision tree analysis of behavior in 2x2 matrix games, the model provides support for their theory of second-order social reasoning. The model can be considered as a cognitively plausible implementation of that analysis.

Model predictions

Our model can be used to provide explicit predictions regarding the use of memory in second-order social cognition (Verbrugge, 2009). In particular, the model relies on various declarative memory retrieval steps, in combination with storage of information in a problem state buffer. An explicit prediction would be that second-order theory of mind reasoning would be affected by performing another task at the same time that would require the same resources (Borst et al., 2010). To our knowledge, such an experiment has not been done yet. Therefore, in the remainder of this paper we would like to propose such an experiment, combined with explicit, quantitative predictions provided by the model. By providing the predictions of our model before actually doing the experiment, we counter the criticism that insufficiently constrained cognitive models can be made to fit any dataset (Roberts & Pashler, 2000).

A task that would require the same resources as hypothesized for social reasoning is a tone counting task. Participants are presented with tones of two different pitches and are requested to count the number of tones for each pitch. This task would tap into the same cognitive resources as hypothesized for the Marble Drop reasoning task, as maintaining two counters at the same time can be considered a heavy working memory load. A control condition in this task would be one in which participants would not need to maintain a counter, but rather just say "high" or "low" every time they heard a tone of a particular (higher or lower) pitch. Because the control task does not require maintaining a counter (a problem state), concurrent execution of this task and the social reasoning task does not pose a conflict, and the different stages of the tasks could be interleaved without much loss of time (Anderson, Taatgen, & Byrne, 2005).

A dual-task model of social reasoning A simple model of this task would involve maintaining the current counter in a problem state buffer. In addition, the model would – upon hearing a tone – check whether the pitch of the tone is the same as the pitch of the previous tone. Specifically, the model compares the pitch of the tone with the pitch

[2] As the data presented here are actually the practice block of the experiment performed by Meijering et al. (2010), the number of observations per participant was 4 for zero-order games, and 8 for first and second-order games.

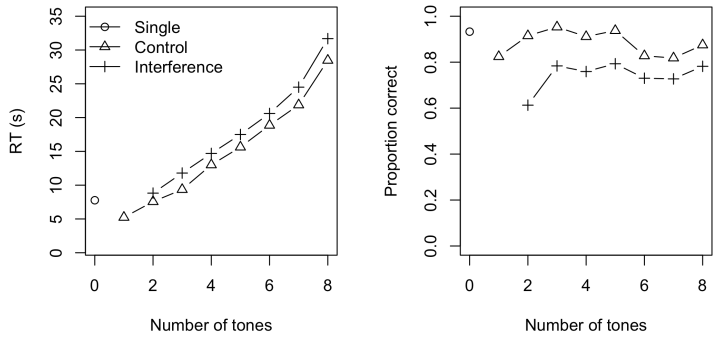


Figure 4. Model predictions for the dual-task social reasoning task. Left: Response time, Right: Accuracy.

associated with the counter in the problem state buffer. If this is the case, the model then retrieves the subsequent number of the stored counter from memory. If this is not the case, the model retrieves the other counter from memory, and based on that retrieves the subsequent number.

Such a model would require both the problem state buffer and the retrieval buffer, resulting in interference with performance on the Marble Drop game. For the control task, both the retrieval and the problem state resources are not required. The model of the control task consists of a simple stimulus response mechanism: When a tone of a particular pitch is heard, the model responds with a vocal response (either "high" or "low").

We adapted our model to also perform the tone counting task. The model was extended with a control mechanism that maintained which task was currently given preference (Salvucci & Taatgen, 2008). The model performs the Marble Drop task until a tone is presented. At that point a switch is made to the counting task. If necessary, the model tries to retrieve the current count and restore the problem state of the counting task. Then, it retrieves the subsequent number from declarative memory followed by a vocal response saying the number. After that, the model tries to restore the problem state of the Marble Drop task by retrieving a comparison from memory.

Model predictions We ran the second-order reasoning model in three conditions for a sufficient number of trials to obtain a stable estimate of the predicted response. In the first condition (Single) the model only performed the Marble Drop task. In the Control condition, the model performed the Marble Drop task in combination with the simple response task. The tones were presented with stimulus onset asynchronies (SOAs) of 2s, 5s, 8s, 11s, 14s, 17s, 20s, and 23s. Only those tones were presented that preceded the model response on the reasoning task. In the Interference condition, the model performed the Marble Drop task in combination with the tone counting task. The tones were presented similarly as in the Control condition.

Figure 4 presents the predicted reaction time and accuracy of the dual-task model as a function of the number of tones presented. The left-most data point in each graph (where the number of tones is zero) represents the behavior of the model under single-task conditions. This is the same as the model fit presented in Figure 3. For the Control condition the model predicts an increase in the response time, and no change in accuracy. This is because the single response task used as secondary task in the Control condition does not share any resources with the Marble Drop task. Thus, responding to the tones only adds time to the Marble Drop response, but does not change the difficulty of the task. In contrast, the tone counting task that the model performs in the Interference condition adds considerable time to the response. In addition, the accuracy of the model decreases as well. Moreover, the mean response time in the Interference condition increases dramatically to 27s (Figure 5), whereas the mean response time in the control condition is 8.3s, which is only slightly above the mean response time of the single response task (7.7s). Our interpretation of these results is that the tone counting task and the Marble Drop task share a cognitive resource. In particular, both tasks require a problem state buffer for maintaining intermediate results. Swapping these problem states takes extra time and is prone to errors, explaining the increased reaction times and the decreased accuracy.

Conclusion

This paper presents the first computational cognitive model of second-order social reasoning. The model uses a decision tree strategy to reason about the opponent's behavior in a social reasoning game. We hypothesize that a decision tree strategy requires (1) declarative memory, and (2) working memory. Declarative memory is required to retrieve successive reasoning steps, while working memory is required to temporarily store these reasoning steps while the

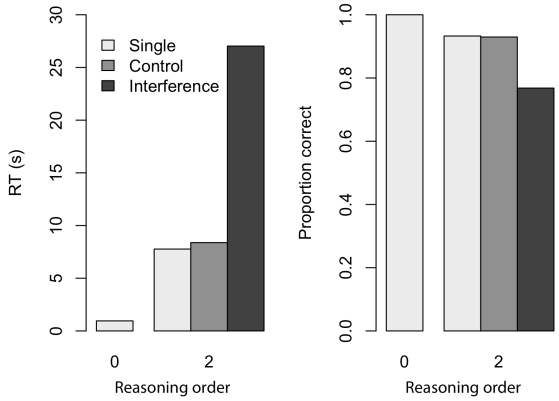


Figure 5. Predicted mean response time for the dual-task model. Left: Response time, Right: Accuracy.

next step is retrieved from memory. We implemented working memory as a problem state buffer using the ACT-R cognitive architecture (Borst et al., 2010). The model fit on data from a social reasoning game called Marble Drop (Meijering et al., 2010) supports the validity of the model. This initial result leads to an explicit prediction for an experiment in which the reasoning game is combined with another task that requires the same cognitive resources as hypothesized by the model. In particular, if the other task also requires the problem state resource, the interference of that task is substantial. On the other hand, a secondary task that is equivalent but does not require the problem state resource exhibits minimal interference. This work is a first step towards understanding higher-order social reasoning from a cognitive modeling perspective.

Acknowledgements

This research was supported by NWO Vici grant NWO-277-80-001 awarded to Rineke Verbrugge.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford UP.

Anderson, J.R., Taatgen, N.A. & Byrne, M.D. (2005). Learning to Achieve Perfect Time Sharing: Architectural Implications of Hazeltine, Teague, & Ivry (2002). *Journal of Experimental Psychology: Human Perception and Performance, 31*(4), 749-761.

Borst, J. P., Taatgen, N. A., & Van Rijn, H. (2010). The problem state: A cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 36*(2), 363-382.

Camerer, C. F. (2003). *Behavioral game theory: Experiments in strategic interaction*. Princeton: Princeton UP.

Flobbe, L., Verbrugge, R., Hendriks, P., & Krämer, I. (2008). Children's application of theory of mind in reasoning and language. *Journal of Logic, Language and Information, 17*(4), 417-442.

Hedden, T., & Zhang, J. (2002). What do you think I think you think?: Strategic reasoning in matrix games. *Cognition, 85*(1), 1-36.

Hendriks, P., Van Rijn, H., & Valkenier, B. (2007). Learning to reason about speaker's alternatives in sentence comprehension: A computational account. *Lingua, 117*(11), 1879-1896.

Lebiere, C., & West, R. L. (1999). A dynamic ACT-R model of simple games, *Proceedings of the Twenty-First Annual Conference of the Cognitive Science Society* (pp. 296-301): Erlbaum.

McKelvey, R. D., & Palfrey, T. R. (1992). An experimental study of the centipede game. *Econometrica, 60*(4), 803-836.

Meijering, B., Van Maanen, L., Van Rijn, H., & Verbrugge, R. (2010). The facilitative effect of context on second-order social reasoning. In R. Catrambone & S. Ohlsson (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Nash, J. (1951). Non-cooperative games. *The Annals of Mathematics, 54*(2), 286-295.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard UP.

Osborne, M., & Rubinstein, A. (1994). *A course in game theory*. Cambridge, MA: MIT Press.

Perner, J. & Wimmer, H. (1985). "John thinks that Mary thinks that...": Attribution of second-order beliefs by 5- to 10-year old children. *Journal of Experimental Child Psychology, 5,* 125-137.

Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences, 4*, 515-526.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review, 107*(2), 358-367.

Rosenthal, R. (1981). Games of perfect information, predatory pricing, and the chain store. *Journal of Economic Theory, 25*, 92-100.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review, 115*(1), 101-130.

Van der Hoek, W., & Verbrugge, R. (2002). Epistemic logic: A survey. In L. A. Petrosjan & V. V. Mazalov (Eds.), *Game theory and applications* (Vol. 8, pp. 53-94). New York: Nova Science.

Van Ditmarsch, H. P. (2002). The description of game actions in cluedo. In L. A. Petrosian & V. V. Mazalov (Eds.), *Game theory and applications* (Vol. 8, pp. 1-28). Hauppage, NY: Nova Science Publishers.

Van Maanen, L., & Van Rijn, H. (2010). The locus of the Gratton effect in picture-word interference. *Topics in Cognitive Science, 2*(1), 168-180.

Van Maanen, L., Van Rijn, H., & Borst, J. P. (2009). Stroop and picture-word interference are two sides of the same coin. *Psychonomic Bulletin & Review, 16*(6), 987-999.

Van Rij, J., Van Rijn, H., & Hendriks, P. (in press). Cognitive architectures and language acquisition: A case study in pronoun comprehension. *Journal of Child Language*.

Verbrugge, R. (2009). Logic and social cognition: The facts matter, and so do computational models. *Journal of Philosophical Logic, 38*(6), 649-680.

Verbrugge, R., & Mol, L. (2008). Learning to apply theory of mind. *Journal of Logic, Language and Information, 17*(4), 489-511.

West, R. L., Lebiere, C., & Bothell, D. (2006). Cognitive architectures, game playing, and human evolution. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation* (pp. 103-123). New York, NY: Cambridge UP.

Wickelgren, W. A. (1977). Speed-accuracy tradeoff and information-processing dynamics. *Acta Psychologica, 41*(1), 67-85.

Neural Correlates of Temporal Credit Assignment

Matthew M. Walsh (mmw187@andrew.cmu.edu)
Department of Psychology, Carnegie Mellon University, 342C Baker Hall
Pittsburgh, PA 15213

John R. Anderson (ja@cmu.edu)
Department of Psychology, Carnegie Mellon University, 345D Baker Hall
Pittsburgh, PA 15213

Abstract

When feedback follows a sequence of decisions, how do people assign credit to intermediate actions within the sequence? To explore this temporal credit assignment problem, we recorded event-related potentials (ERPs) as participants performed a sequential decision task. Our ERP analyses focused on feedback-related negativity (FRN), a component thought to reflect neural reward prediction error. The experiment showed that FRN followed negative feedback and negative intermediate states. This outcome suggests that participants evaluated intermediate states in terms of expected future reward, and that these evaluations guided acquisition of earlier actions within sequences. We compared these results to the predictions of three reinforcement learning models that address temporal credit assignment: Actor-critic, Q-Learning, and SARSA.

Keywords: Actor-critic; ERP; Q-Learning; SARSA; Temporal credit assignment; Temporal difference learning.

Introduction

To behave adaptively, humans and animals must learn to predict the outcomes of their actions. Reinforcement learning (RL) provides a mechanism for acquiring this knowledge through trial-and-error interactions with an environment (Sutton & Barto, 1998). According to many RL models, the difference between expected and actual outcomes, or "reward prediction error", provides a learning signal. By revising estimates based on prediction error, humans and animals learn to anticipate outcomes, and consequently, to select actions that maximize reward and minimize punishment.

RL methods have influenced contemporary neuroscientific theories. For example, one popular RL method, temporal difference (TD) learning, has been used to characterize the phasic response of midbrain dopamine neurons to rewarding and punishing events (Schultz, Dayan, & Montague, 1997). Several studies have confirmed that the response of these neurons depends on reward magnitude and reward likelihood (Tobler, Fiorillo, & Schultz, 2005). Rather than responding directly to experienced outcomes, however, these neurons respond to the *difference* between expected and actual rewards. Thus, midbrain dopamine neurons convey information about TD prediction error.

Recent ERP research with humans has revealed a frontocentral negative component that appears 200-300 ms after the display of error feedback (Gehring & Willoughby, 2002; Miltner, Braun, & Coles, 1997). Three features of this feedback-related negativity (FRN) indicate that it too reflects neural reward prediction error. First, FRN is larger after unexpected than expected outcomes (Holroyd et al., 2009). Second, FRN correlates with behavioral adjustment (Cohen & Ranganath, 2007). Third, neuroimaging experiments, source localization studies, and single cell recordings suggest that FRN originates from the anterior cingulate cortex (ACC), a region implicated in goal-direct behavioral selection (Holroyd et al., 2009). These ideas have been synthesized in the reinforcement learning theory of the error-related negativity (RL-ERN), which proposes that midbrain dopamine neurons transmit a prediction error signal to the ACC, and that this signal strengthens or weakens the actions that precipitated outcomes (Holroyd & Coles, 2002).

Although the RL-ERN theory has stimulated a great deal of research (for review, see Nieuwenhuis et al., 2004), feedback immediately follows actions in most studies of FRN. Similarly, although RL methods have stimulated a great deal of psychological research (for review, see Fu & Anderson, 2006), most studies of RL in humans also involve relatively simple tasks. These scenarios contrast with complex control problems we face in daily life. One such problem is temporal credit assignment. When feedback follows a sequence of decisions, how should credit be assigned to intermediate actions within the sequence?

Here, we consider three TD learning methods that address the temporal credit assignment problem: Actor-critic, Q-Learning, and SARSA. These methods evaluate actions in terms of immediate and future reward. For example, an action may bring an individual into direct contact with reward. Alternatively, an action may bring an individual into a state associated with a high probability of future reward. How should future reward be calculated? In the actor-critic model, future reward is treated as the value of potential options, weighted according to the probability of selecting each (Sutton & Barto, 1998). In Q-Learning, future reward is treated as the value of the best potential option (Watkins & Dayan, 1992). Finally, in SARSA, future reward is treated as the value of the future option that is actually selected (Rummery & Niranjan, 1994).

In the current experiment, we recorded ERPs as participants performed a sequential decision task. The initial decision in each sequence brought participants to an intermediate state associated with a high or a low probability of receiving positive feedback, and the final decision was followed by positive or negative feedback. Based on the idea that FRN reflects neural prediction error

(Holroyd & Coles, 2002), we tested two main hypotheses. First, FRN should be greater for unexpected than for expected outcomes. This follows from the fact that RL models anticipate probable outcomes. Consequently, model prediction error is greater for unexpected than for expected outcomes. Second, if credit assignment occurs "on the fly", as predicted by the TD model, negative feedback *and* negative intermediate states will evoke FRN. Alternate methods exist for performing temporal credit assignment (e.g. model-based RL, eligibility traces). If credit is only assigned at the end of the decision episode, as predicted by these alternate models, only negative feedback will evoke FRN. In addition to testing these two hypotheses, we compared predictions of three TD models, actor-critic, Q-Learning, and SARSA, to the behavioral and neural results of the experiment.

Experiment

Task

A pair of letters appeared at the start of each trial. A cue appeared after participants selected a letter. A second pair of letters followed the cue. Feedback appeared after participants selected a second letter. Participants completed 2 experiment blocks of 400 trials. 13 graduate and undergraduate students participated in the experiment.

Within each block, one pair of letters appeared at the start of all trials (Figure 1). When participants chose the correct letter in the first pair ("J" in this example), a positive and a negative cue appeared equally often. When they chose the incorrect letter in the first pair ("R"), a negative cue always appeared. A second pair of letters followed the cue. The correct letter in the second pair depended on the cue identity. The correct letter for the positive cue ("V" in this example) was rewarded with 80% probability, and the correct letter for the negative cue ("T") was rewarded with 20% probability. Incorrect letters were never rewarded. Consequently, optimal selections yielded positive feedback for 80% of trials involving the positive cue (0.8 Cue) and for 20% of trials involving the negative cue (0.2 Cue). The symbols "#" and "*" denoted positive and negative feedback.

Recording

The EEG was recorded from 32 Ag–AgCl sintered electrodes (10–20 system), and recordings were algebraically re-referenced offline to the average of the right and left mastoids. The vertical EOG was recorded as the potential between electrodes placed above and below the left eye, and the horizontal EOG was recorded as the potential between electrodes placed at the external canthi. The EEG and EOG signals were amplified by a Neuroscan bioamplification system with a bandpass of 0.1-70 Hz and digitized at 250 Hz. Eye blinks were corrected using ICA. 800 ms epochs were extracted from the continuous recording and these epochs were baseline corrected relative to the 200 ms prestimulus interval.

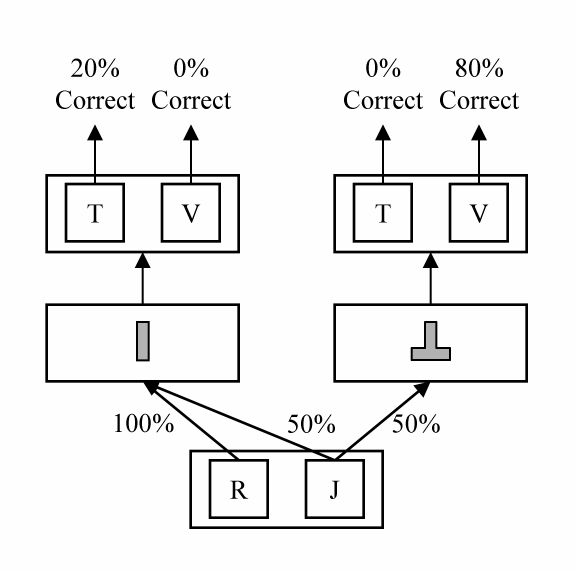


Figure 1. Experiment states, transition probabilities, and outcome likelihoods.

Feedback-locked ERPs were analyzed for trials where participants selected the correct letter for the cue, and FRN was calculated as the difference between ERP waveforms after losses and wins. FRN amplitude is often confounded by changes in P300 amplitude, a component that is also sensitive to event likelihoods. Consequently, we compared losses and wins that were equally likely by creating an "expected outcome" difference wave (0.2 Cue losses – 0.8 Cue wins), and an "unexpected outcome" difference wave (0.8 Cue losses – 0.2 Cue wins). FRN was measured as mean voltage of the difference waves from 200-300 ms after feedback onset, relative to the 200 ms prestimulus baseline. Cue-locked ERPs were analyzed for trials where participants selected the correct starting letter (after which the probability of receiving the 0.2 or the 0.8 Cue was equal). Cue FRN was measured as mean voltage of the cue difference wave (0.2 Cue – 0.8 Cue) from 200-300 ms after cue onset.

Models

Actor-critic (Sutton & Barto, 1998)

The actor-critic (AC) model computed a state-action value function, $Q(s,a)$, and a state value function, $V(s)$. The state-action value function, which corresponded to the actor, enabled action selection. The state-value function, which corresponded to the critic, enabled evaluation of action consequences. Actions affected the transition from state s_t to s_{t+1}, and actions affected the presentation of reward, r_{t+1}. Following the selection of an action, a_t, the critic issued an evaluation in the form of prediction error, δ,

$$\delta = \left[r_{t+1} + \gamma \bullet V\left(s_{t+1}\right)\right] - V\left(s_t\right). \qquad (1)$$

The AC model maximized the combined immediate, r_{t+1}, and future reward, $V(s_{t+1})$, and future reward was discounted by γ (γ < 1.0). The value of the previous state, $V(s_t)$, was updated according to

$$V(s_t) \leftarrow V(s_t) + \alpha \bullet \delta, \qquad (2)$$

where α controlled the learning rate (0.0 < α < 1.0). The value of the previous state-action pair, $Q(s_t,a_t)$, was updated according to

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \bullet \delta. \qquad (3)$$

Q-Learning (Watkins & Dayan, 1992)

The AC and Q-Learning models differed in two ways. First, the Q-Learning model used an action-state value function, $Q(s,a)$, to select actions and to evaluate outcomes. Second, the Q-Learning model treated future reward as the value of the optimal selection policy in state *t + 1*,

$$\delta = [r_{t+1} + \gamma \bullet \max_a Q(s_{t+1}, a)] - Q(s_t, a_t). \qquad (4)$$

As in the AC model, future reward was discounted by γ, and the state-action value function was updated according to Equation 3.

SARSA (Rummery & Niranjan, 1994)

Like the Q-Learning model, the SARSA model only required an action-state value function, $Q(s,a)$. Unlike the Q-Learning model, however, the SARSA model treated future reward as the value of the actual state-action pair selected in state *t + 1*,

$$\delta = [r_{t+1} + \gamma \bullet Q(s_{t+1}, a_{t+1})] - Q(s_t, a_t). \qquad (5)$$

As with the AC and Q-Learning models, future reward was discounted by γ, and the state-action value function was updated according to Equation 3.

To summarize, all models used δ to learn the values of the state-action pairs that comprised the experiment task (Figure 1), and all models sought to select actions that maximized immediate and future reward. Although the initial selection in each trial was not followed by immediate reward (i.e. r_{t+1} = 0), the initial selection was followed by future reward associated with a subsequent state (AC model), or a subsequent state-action pair (Q-Learning and SARSA models). As such, prediction error for the initial selection was calculated as the difference between discounted future reward and the value of the first state (AC model), or the value of the first state-action pair (Q-Learning and SARSA models). Prediction error for the final selection was calculated as the difference between immediate reward and the value of the second state (AC model), or the value of the second state-action pair (Q-Learning and SARSA models). Positive feedback had a value of 1.0 and negative feedback had a value of 0.0[1].

Model predictions were based on 500 simulations. All state and state-action pairs began with values of 0.5 and values were updated according to prediction error. In each trial, logistically distributed noise was added to state-action values, and the state-action pair with the greatest value was selected. Two model parameters, learning rate (α = .05) and the temporal discounting factor (γ = 0.8), were fixed according to values reported in Fu & Anderson (2006). Interestingly, when α and λ were treated as free parameters, mean squared error (MSE) for each model was minimized at values of α and γ within ±0.02 of their fixed values. Selection noise (*t*, defined as the standard deviation of the logistically distributed noise added to state-action pairs) remained as a free parameter. We compared model selections to participant performance. Additionally, we computed the difference in δ for expected feedback (0.2 Cue losses – 0.8 Cue wins), unexpected feedback (0.8 Cue losses – 0.2 Cue wins), and cues (0.2 Cue – 0.8 Cue) to derive model FRN. We then fit model FRN to observed FRN using a slope term (*m*) and a zero intercept.

Results

Behavioral Results

Selection accuracy varied by choice, $F(2,24) = 10.33$, $p < .001$, and selection accuracy increased by block half, $F(1,12) = 102.54$, $p < .0001$ (Figure 2). Selection times for correct responses did not vary by choice, $F(2,24) = 2.47$, $p > .1$, or block half, $F(1,12) = 2.55$, $p > .1$.

ERP Results

We first analyzed feedback-locked ERPs. Waveforms showed a pronounced negativity from 200-300 ms after loss feedback (Figure 3). This FRN (loss – win) appeared to be greater for unexpected than for expected outcomes. A 3 (site: Fz, Cz, Pz) by 2 (outcome likelihood: expected, unexpected) ANOVA on FRN amplitude revealed effects of site, $F(2,24) = 10.91$, $p = .005$, and outcome likelihood, $F(1,12) = 13.26$, $p = .003$. FRN was greater for unexpected than for expected outcomes at site Fz, $t(12) = 3.69$, $p = .003$. We also considered FRN over the first and the second halves of blocks (Figure 5). A 2 (outcome likelihood) by 2 (block half) ANOVA at Fz showed an effect of outcome likelihood, $F(1,12) = 11.68$, $p = .005$, but not block half, $F(1,12) = 0.10$, $p > .1$. Although the interaction was not significant, $F(1,12) = 3.13$, $p > .1$, experience caused FRN to increase for unexpected outcomes and to decrease for expected outcomes[2].

We then analyzed cue-locked ERPs. A 3 (site) by 2 (cue) ANOVA revealed a nonsignificant effect of site, $F(2,24)$ =

[1] Because the model used a soft-max decision policy, choice proportions depended only on the absolute differences between Q-values. Consequently, changes to the noise parameter, *t*, can accommodate a wide range of positive and negative reward values.

[2] In a subsequent experiment with a larger sample size, this interaction reached significance.

0.24, $p > .1$, a marginal effect of cue, $F(1,12) = 3.05$, $p = .1$, and a nonsignificant interaction, $F(2,24) = 1.78$, $p > .1$. ERPs were relatively more negative for 0.2 than for 0.8 Cues at site Fz, but the effect failed to reach significance, $t(12) = 1.77$, $p = .1$. When we considered the first and the second halves of blocks separately, however, a different picture emerged (Figure 4). A 2 (cue) by 2 (block half) ANOVA at Fz revealed a significant interaction between cue and block half, $F(1,12) = 6.56$, $p = .025$. In the first half of blocks, ERPs did not vary by cue, $t(12) = .46$, $p > .1$, but in the second half of blocks, ERPs were relatively more negative for 0.2 than for 0.8 Cues, $t(12) = 2.76$, $p = .017$. The discovery of cue FRN indicates that participants evaluated intermediate outcomes in terms of future reward, as predicted by the temporal difference models.

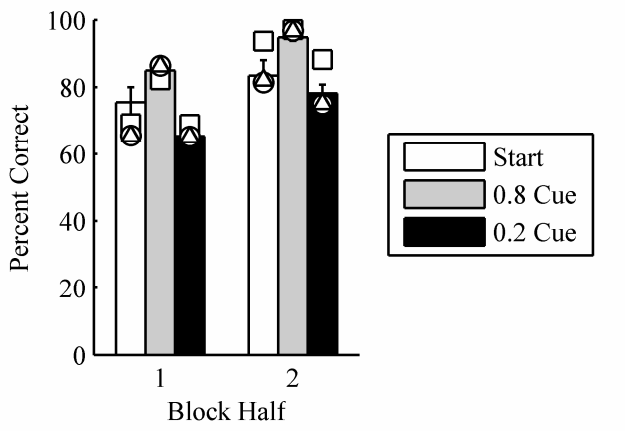


Figure 2. Selection accuracy for start pair, 0.8 Cues, and 0.2 Cues by block half and for participants (bars), AC (squares), Q-Learning (circles), and SARSA (triangles).

Model Performance

For each model, we estimated the value of noise, t, that best accounted for selection accuracy over the first and second halves of experiment blocks. For the Q-Learning and SARSA models, MSE was minimized at $t = 0.1$ (Q-Learning: MSE = 0.002, $r^2 = 0.90$; SARSA: MSE = 0.002, $r^2 = 0.90$). For the AC model, MSE was minimized at $t = 0.2$ (MSE = 0.004, $r^2 = 0.71$). As seen in Figure 2, all models displayed effects of choice and block half like those seen for participants. Additionally, the Q-Learning and SARSA models, which were structurally most similar, yielded nearly identical predictions to one another ($r^2 = 0.99$). Finally, the AC model outperformed participants and the other two models over the second half of blocks.

Next, we examined whether FRN related to model δ. To do so, we computed model FRN as the difference in δ for expected feedback, unexpected feedback, and cues. For each model, we estimated the value of the slope parameter, m, that best accounted for FRN over the first and second halves of experiment blocks. For the Q-Learning and SARSA models, MSE was minimized at $m = 2.6$ (Q-Learning: MSE = 0.295, $r^2 = 0.85$; SARSA: MSE = 0.294, $r^2 = 0.85$). For the AC model, MSE was also minimized at $m = 2.6$ (MSE = 0.262, $r^2 = 0.86$). As seen in Figure 5, all models predicted that cue FRN would increase with experience, and that FRN for unexpected outcomes would increase with experience while FRN for expected outcomes would decrease with experience. These trends were observed.

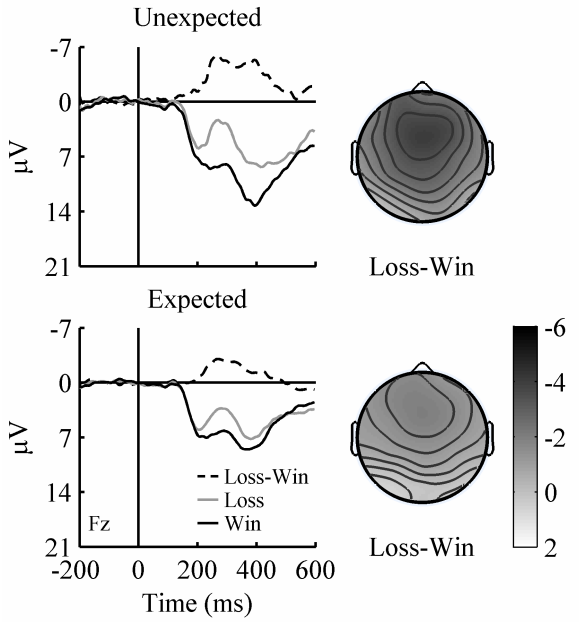


Figure 3. ERPs evoked by unexpected and expected losses and wins at site Fz (left panels). Scalp voltage topography for loss – win comparison from 200-300 ms (right panels).

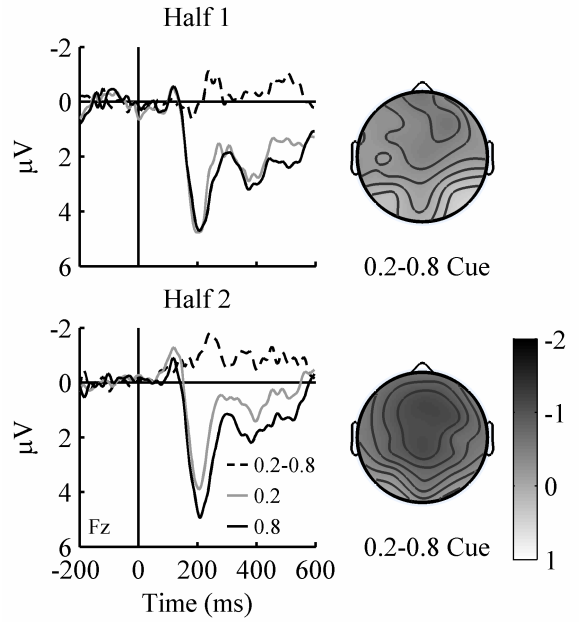


Figure 4. ERPs evoked by 0.2 and 0.8 Cues for the first and the second halves of blocks at site Fz (left panels). Scalp voltage topography for 0.2 Cue – 0.8 Cue comparison from 200-300 ms (right panels).

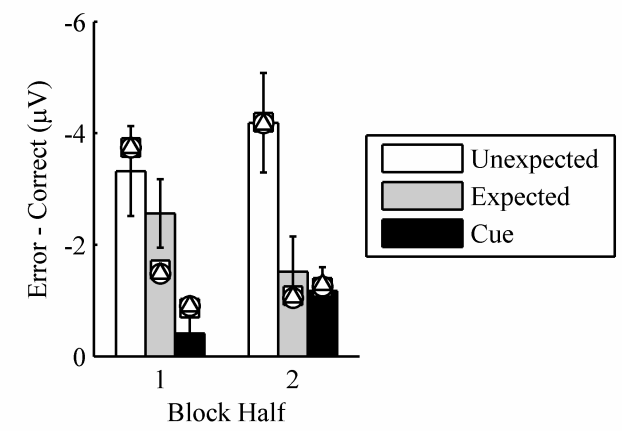


Figure 5. FRN for unexpected outcomes, expected outcomes, and cues by block half and for participants (bars), AC (squares), Q-Learning (circles), and SARSA (triangles).

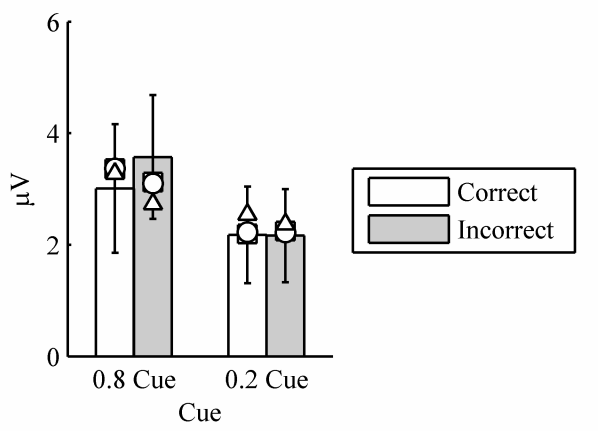


Figure 6**.** Cue-locked voltages preceding correct and incorrect responses by cue and for participants (bars), AC (squares), Q-Learning (circles), and SARSA (triangles).

The behavioral results favored the Q-Learning and SARSA models. The AC model outperformed participants and the other two models over the second half of blocks. Performance differences between models related to the nuanced meaning of state-action pairs, $Q(s,a)$, for each. In the Q-Learning and SARSA models, Q-values approximate values of state-action pairs. In the AC model, Q-values approximate selection preferences that maximize the state-value function, $V(s)$. Because a deterministic selection policy maximized the state-value function, $V(s)$, in our task, Q-values in the AC model became increasingly polarized until near-deterministic selections emerged. The same effect could be achieved in the Q-Learning and SARSA models by annealing the noise parameter.

To further distinguish between the Q-Learning and SARSA models, we re-analyzed cue-locked waveforms based on cue identity (0.2 Cue, 0.8 Cue) and the response that followed the cue. If prediction error depended on the value of future actions, as predicted by SARSA, we expected that cue-locked waveforms would be more negative before participants chose the incorrect response than before they chose the correct response. From 200-300 ms after cue presentation, average area under the 0.2 Cue waveform was less than area under the 0.8 Cue waveform at site Fz, $F(1,12) = 8.40$, $p = .013$ (Figure 6). Waveforms did not depend on the accuracy of the forthcoming response, however, $F(1,12) = 1.71$, $p > .1$.

We computed model δ for the same combination of factors[3]. Q-Learning and AC predictions were consistent with observations (Q-Learning: MSE = 0.237, $r^2 = .0.77$; AC: MSE = 0.225, $r^2 = .0.76$) in that they predicted an effect of cue but not response accuracy. In contrast, the SARSA model predicted a more negative signal before incorrect than correct responses (MSE = 0.419, $r^2 = .0.32$), owing to how the algorithm computed future reward (Eq. 5).

[3] This analysis was based on the area under individual waveforms rather then FRN. Consequently, we computed new slope and intercept terms to compare model δ to observations.

General Discussion

Although the RL-ERN theory has stimulated a great deal of research, feedback immediately follows actions in most studies of FRN. Similarly, although RL methods have stimulated a great deal of psychological research, most studies of RL in humans involve simple environments. In the current experiment, we examined learning in a more complex problem space. We asked how people assign credit to intermediate actions when making sequences of decisions.

The experiment yielded two clear results. First, FRN was greater for unexpected than for expected outcomes. Although some studies have reported a relationship between FRN and prediction error (Holroyd et al., 2009), others have not (Hajcak et al., 2005). This discrepancy has led to the proposal that FRN relates most strongly to prediction error when outcomes are contingent on behavior (Holroyd et al., 2009). In our experiments, feedback was contingent on behavior, and consistent with the proposal of Holroyd et al. (2009), we did observe a relationship between prediction error and FRN. Second, FRN also followed negative intermediate outcomes even though these outcomes did not directly signal reward. This result shows that people evaluated intermediate outcomes in terms of expected future reward. Although many theories propose that such evaluations underlie temporal credit assignment (Fu & Anderson, 2006; Holroyd & Coles, 2002; Schultz, Dayan, & Montague, 1997; Sutton & Barto, 1998), these results provide one of the clearest demonstrations of TD learning to our knowledge.

We also examined three TD methods: Actor-critic, Q-Learning, and SARSA. A recent neuroimaging study provided support for the AC model by showing that activity in the dorsal and ventral striatum of the basal ganglia corresponded to the behavior of the actor and the critic in the AC model (O'Doherty et al., 2004). Alternatively, single-cell recordings from midbrain dopamine neurons in monkeys have supported SARSA (Morris et al., 2006), and

recordings from dopamine neurons in rats have supported Q-Learning (Roesch, Calu, & Schoenbaum, 2007). An integrative account of these findings is hindered by the between species comparison. Consequently, it is unclear, as of yet, which form of TD control is most applicable to humans. The behavioral and neural results of the current experiment were consistent with Q-Learning. This considerations not withstanding, the current data do not definitively distinguish between TD variants. The more valuable contribution of this work is the demonstration that intermediate states inherit value, a feature central to each TD model. Future studies should aim to elucidate the precise TD algorithms that underlie neurological computations.

Our simulations demonstrated that the core Q-Learning model could account for the behavioral and neural data. Additionally, our computational instantiation clarified two nuanced features of the experiment results. First, FRN decreased for expected outcomes and increased for unexpected outcomes. Model FRN changed in the same manner. Because utility estimates began at 0.5, δ was initially -0.5 (0.0 – 0.5) for all losses, and δ was initially 0.5 (1.0 – 0.5) for all wins. As the model learned, the utility of the correct response for the 0.2 Cue approached 0.2 and the utility of the correct response for the 0.8 Cue approached 0.8. Consequently, δ magnitude decreased for expected wins and losses, and δ magnitude increased for unexpected wins and losses, giving rise to the observed changes in FRN.

Second, cue FRN increased with experience. The Q-Learning model (and in fact all TD models) also showed an experience-dependent increase in cue FRN. The models only distinguished between positive and negative cues after the values of the states and actions that followed those cues (e.g. future reward) became polarized. As this result demonstrates, the TD models learn the utility of actions that are near to rewards before learning the utility of actions that are far from rewards. Humans and animals also exhibit this learning gradient (Fu & Anderson, 2006).

Do the results of this experiment indicate that TD methods alone are sufficient for coping with temporal credit assignment? We think not. Although participants faced a discrete Markov decision process (MDP) in our experiment, people must sometimes identify current states *and* recall past transitions. Violations of the Markov property may be problematic for TD methods. Additionally, although TD learning reduces the delay between action selection and credit assignment, TD learning does not typically eliminate delays in continuous time domain tasks. An important question for future research is how people integrate TD learning with other RL methods, like eligibility traces and model-based RL, to behave proficiently in complex environments.

Acknowledgments

This work was supported by training grant T32GM081760 and NIH training grant MH 19983 to the first author and NIMH grant MH068243 to the second author.

References

Cohen, M.X., & Ranganath, C. (2007). Reinforcement learning signals predict future decisions. *The Journal of Neuroscience, 27,* 371-378.

Fu, W.T., & Anderson, J.R. (2006). From recurrent choice to skill learning: a reinforcement-learning model. *Journal of Experimental Psychology: General, 135,* 184-206.

Gehring, W.J., & Willoughby, A.R. (2002). The medial frontal cortex and the rapid processing of monetary gains and losses. *Science, 295,* 2279-2282.

Hajcak, G., Holroyd, C.B., Moser, J.S., & Simons, R.F. (2005). Brain potentials associated with expected and unexpected good and bad outcomes. *Psychophysiology, 42,* 161-170.

Holyroyd, C.B., & Coles, M.G.H. (2002). The neural basis of human error processing: reinforcement learning, dopamine, and the error-related negativity. *Psychological Review, 109,* 679-709.

Holroyd, C.B., Krigolson, O.E., Baker, R., Lee, S., & Gibson, J. (2009). When is an error not a prediction error? An electrophysiological investigation. *Cognitive, Affective, & Behavioral Neuroscience, 9,* 59-70.

Miltner, W.H.R., Braun, C.H., & Coles, M.G.H. (1997). Event-related brain potentials following incorrect feedback in a time-estimation task: evidence for a "generic" neural system for error detection. *Journal of Cognitive Neuroscience, 9,* 788-798.

Morris, G., Nevet, A., Arkadir, D., Vaadia, E., & Bergman, H. (2006). Midbrain dopamine neurons encode decisions for future action. *Nature Neuroscience, 9,* 1057-1063.

Nieuwenhuis, S., Holroyd, C.B., Mol, N., & Coles, M.G.H. (2004). Reinforcement-related brain potentials from medial frontal cortex: origins and functional significance. *Neuroscience and Biobehavioral Reviews, 28,* 441-448.

O'Doherty, J., Dayan, P., Schultz, J., Deichmann, R., Friston, K., & Dolan, R.J. (2004). Dissociable roles of ventral and dorsal striatum in instrumental conditioning. *Science, 304,* 452-454.

Roesch, M.R., Calu, D.J., & Schoenbaum, G. (2007). Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature Neuroscience, 10,* 1615-1624.

Rummery, G.A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. (Tech. Rep. CUED/F-INFENG/TR166). Cambridge University.

Schultz, W., Dayan, P., & Montague, P.R. (1997). A neural substrate of prediction and reward. *Science, 275,* 1593-1599.

Sutton, R.S., & Barto, A.G. (1998). *Reinforcement learning: an introduction.* Cambridge, MA: MIT Press.

Tobler, P.N., Fiorillo, C.D., & Schultz, W. (2005). Adaptive coding of reward value by dopamine neurons. *Science, 307,* 1642-1645.

Watkins, C.J., & Dayan, P. (1992). Q-learning. *Machine Learning, 8,* 279-292.

A Computational Model of Functional Category Learning in a Cognitive Architecture

Yongjia Wang (yongjiaw@umich.edu)
John E. Laird (laird@umich.edu)
Computer Science and Engineering, EECS Department
University of Michigan, Ann Arbor, MI 48109 USA

Abstract

Categorization of objects is an important cognitive capability for human and higher animals. Phenomena related to category learning have been investigated both in human subjects and in animal behavior studies. However, it is less well understood in the computational processes that are responsible for the emergence of functionally meaningful categorizations from specific learning contexts. Here we present a unique computational model integrating object categorization and reinforcement learning (RL) in the Soar cognitive architecture. Our model simultaneously captures how object categorization affects behavior adaptation, and how behavioral adaptation influences object categorization over time in a specific functional context. Results from synthetic data demonstrate that our model successfully improves the speed of RL via categorization. The qualitative predictions from our model are consistent with existing theories of category learning.

Keywords: cognitive architecture; category learning; reinforcement learning; behavioral adaptation

Introduction

Category learning has been actively studied in higher animals including human (Ashby & Maddox 2005) and primates (Smith 2010). Categorization enables an individual to response to a novel stimulus, which resembles some other stimuli with known responses.

In this paper, we model several related phenomena in human category learning. The most important one is related to the notion of basic-level category as described by Rosch (1978). Consider the following two examples of abstraction hierarchies: furniture-chair-rocker and vehicle-car-sedan. The middle categories, chair and car, are basic categories, because they dominate both their subordinate and superordinate categories in terms of how fast they can be retrieved when a person is asked to describe the object without being put in a specific context. The original theory about basic-level categories was mainly concerned with this 'uniformity' aspect of category recognition across different individuals. On the other hand, there are also variations. First, non-basic level categories are frequently chosen in specific task contexts. Second, basic-levels are dependent on long term learning experience and can be significantly different across individuals in specific domains. All of these are characteristics of category learning. However, there has been a lack of computational models that coherently explain the combination of basic-level effects, context effects, and long-term learning effects in a specific functional setting, where a cognitive agent has to interact with the world to achieve some goals.

We present a unique computational model of category learning that integrates a hierarchical perceptual category learning component and a reinforcement learning component in the Soar cognitive architecture (Laird 2008). In our model, the underlying computation mechanism improves the agent's behavioral adaptation through category learning and at the same time results in the emergence of functionally meaningful categorizations as a result of feedback from reinforcement learning. We term our model a *functional category learning model*.

Our functional category learning model relies on perceptual category learning, and has the following features. First, functional categorization requires additional functional properties as input that are non-perceptual. For example, a venomous snake is in a different functional category from a harmless snake, but they may look very similar and fall under the same perceptual category of snakes. Second, functional categories are by definition specific to a particular functional context. For example, categorizations of animals as sources of food versus as pets are very different. Third, functional categories are directly related to decision making and are adaptive relative to the agent's experience. For example, a domain expert develops more detailed categorizations than novices do. Our hypothesis is that basic-level categories are rooted in people's experience, and depending on how objects are used, the categories can be significantly different across cultures, even individuals within the same culture.

Our functional category learning model involves two components. One is a perceptual category learning system, which can automatically learn hierarchical category structures based on innate perceptual features. The other is a reinforcement learning system, which uses the perceptual categories as the representational basis and incrementally forms functionally meaningful categories based on their utility values.

Hierarchical Categorization

There is a long history of hierarchical models of category learning. Quillian (1968) proposed the semantic network model, which can represent categorical relationships among objects in a hierarchical structure. However, the semantic network model does not include a learning mechanism to build the structure. COBWEB (Fisher 1987) is an algorithm that can incrementally learn a hierarchical organization of categories. A previous version of the ICARUS cognitive architecture used a COBWEB-based system, called LABYRINTH for its declarative learning and memory (Langley *et al.* 1991). Ambros-Ingerson *et al.* (1990)

described a neurologically inspired hierarchical clustering algorithm, which operates in a way very different from COBWEB and Granger (2005) has demonstrated the plausibility of using such hierarchical clustering algorithm as a principled computational instruction for human cognition.

Reinforcement Learning

Hierarchical category learning provides the necessary representational basis, however the representation itself is insufficient for functional category learning because it has no direct connection to how the learned knowledge can be used. Another learning process is required to connect the category representations with the agent's intrinsic functional meanings. We consider reinforcement learning (RL, Sutton & Barto 1998) as a candidate mechanism to establish such connections via incremental trial-and-error learning with feedback.

RL has been successfully applied in adaptively learning optimal control policies in the field of machine learning. The general model of RL has also been considered as a mechanism for human skill learning (Fu & Anderson 2006). Cognitive architectures such as Soar (Laird 2008) and ACT-R (Anderson *et al.* 2004) both have a reinforcement learning mechanism. However, there has not been a computational model integrating category learning and RL in these cognitive architectures.

Demonstration Task

We briefly describe our demonstration task before describing the implementation of our model, so that we can illustrate how the model works using a concrete example.

The demonstration task models a hunting scenario where the agent is presented with pairs of prey and hunting tools. There are diverse types of prey and tools, and different tools have different effectiveness on different prey. For example, a slingshot is good for small birds, but it will not work for larger prey. We assume that the agent does not have prior knowledge to predict the outcomes based on perceptual features of the objects. The agent must incrementally acquire such connections based on its experience through RL. During interaction with the world, the agent receives a positive reward if hunting is successful and a negative reward if it is unsuccessful. In order to learn faster, the agent needs to generalize its predictions based on perceptual similarity. For example, if the agent has learned that a bow is good for hunting rabbits, then it is likely to work against a woodchuck as well. Meanwhile, to improve generality, the agent must adapt its learning to the right level of abstraction through the course of using RL.

Model Implementation

Overview

Our model is implemented by combining a hierarchical category learning (HCL) system with Soar-RL (Nason & Laird 2005), which has been shown to successfully model animal behavioral data (Wang & Laird 2007). Our model uses the HCL component to perform perceptual learning.

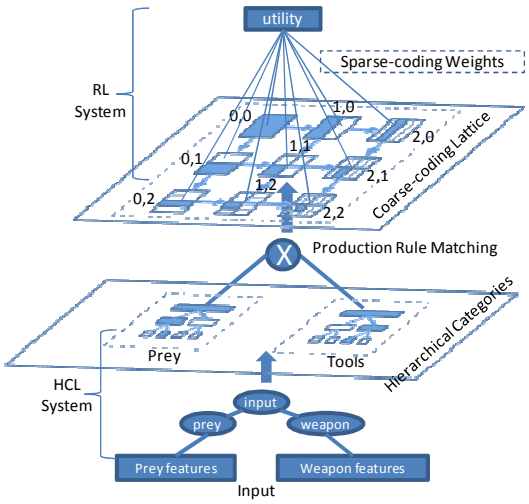


Figure 1: Overall structure of the system viewed as a two-layer network

The output of the HCL system is the input to the RL system. We have experimented with both COBWEB and a biologically inspired hierarchical clustering algorithm (Ambros-Ingerson *et al.* 1990). In general, any incremental hierarchical clustering system will be compatible with our model.

Learning Algorithm

In a functional category learning model, the functional utilities of objects are associated with specific actions, and can be naturally represented as value functions in the RL system. Soar-RL encodes the value function as a set of production rules, with an expressive syntax equivalent to first-order logic. The left-hand side of a rule tests state and action features, while the right-hand side generates the expected value for the matching state action pair. The expected value of an action is the sum of the values of all rules matching the current state and that action. The Soar-RL model is a special instance of the sparse-coarse coding approach to value function approximation (Sutton 1996).

In our functional category learning model, instead of using raw perceptual features of the objects in the state representation, the RL system uses the symbolic category representations from the HCL system. The entire structure of our model can be viewed as a two-layer network as shown in Figure 1. The bottom layer represents the HCL system. In this paper, we assume such hierarchical structure has been learned by the agent through regular perceptual category learning before the hunting task. And we investigate the emerging properties of doing reinforcement learning with such hierarchical categorization. The dark colored nodes in the hierarchies represent symbolic categories matching with the input objects. These symbolic categories are used in the state representation and are matched by rules in the RL system. Rules are represented as cells in the coarse-coding layer. A rule testing general category symbols will be coarser than a rule testing more specific category symbols. Dark colored cells represent rules that match the current state. The numbers on each grid indicates the hierarchy levels for component hierarchies,

which will be explained later. The grids form an emerging lattice structure, with the transitive relationship *coarser-than*, represented by the arrows.

We formally describe the general algorithm below. To learn the target value function of a state action pair, the system first maps the input objects into a vector of functional roles R, which represents the argument types of the target function. The vector O represents the input objects binding with R:

$$R = (r_1, r_2, \dots, r_n)$$
$$O = (o_1, o_2, \dots, o_n)$$

In the example, the function is to predict the utility of hunting some prey with some tool, and for a particular instance, the inputs are two objects: rabbit and bow. According to our notation, input to the system will look like R=(prey, tool), O=(rabbit, bow). After matching objects with functional roles, the HCL system incrementally builds a set of hierarchies H correspondingly:

$$H = (h_1, h_2, \dots, h_n)$$

Let *height*(h_i) denote in the height of the hierarchy h_i, and k_i denote a cluster/category/node within the hierarchy. Let *level*(k_i) denote the level of cluster k_i in hierarchy h_i, with the root level being 0. Cells, grids and their relations, shown in Figure 1, are defined as following:

$$Cells = \{C_K, K = (k_1, k_2, \dots, k_n) | k_i \in clusters\ in\ h_i\}$$
$$Grids = \{G_L, L = (l_1, l_2, \dots, l_n) | 0 \le l_i \le height(h_i)\}$$
$$C_K\ belongs\ to\ G_L \equiv \forall i \in [1,n], level(k_i) = l_i$$
$$G_L \prec G_M \equiv \forall i \in [1,n], l_i \le m_i$$

More intuitively, each cell represents a rule in our RL system. A set of cells are composed into a grid that partitions the state space at a specific level of resolution. There is an emerging lattice structure among the grids with the transitive relation *coarser-than* ($\prec$). For a given object o_i, the activation of a cluster k_i is denoted as $a(k_i)$:

$$a(k_i) = \begin{cases} 1\ if\ o_i \in k_i \\ 0\ if\ o_i \notin k_i \end{cases}$$

The mapping from o_i to k_i is achieved via category recognition in the HCL system, and only a single path of clusters are activated for a particular input as shown in Figure 1. Details of the COBWEB algorithm can be found in Fisher (1987). $a(k_i)$=1 means object o_i in the current state, bound to the corresponding functional role r_i, is an instance of the category represented by the cluster k_i. The activation of a cell, $a(C_K)$, is defined as:

$$a(C_K) = \prod_{i=1}^{n} a(k_i)$$

$a(C_K)$=1 only when all the objects match with the rule, which will fire to participate in predicting and learning the target value. The weight, $w(C_K)$, from the cell to the output unit is represented as a numeric value associated with the rule in the RL system. The learning algorithm updates the weights according to the *delta rule* for the identity activation function used in our RL system, where y is the predicted value and o is the target output value (current reward + discounted future rewards). The learning rate α for a specific rule C_K is chosen to decay over time t, where t is represented by the times the rule has been trained:

$$y = \sum_{C_K} (w(C_K) \times a(C_K))$$
$$\Delta w(C_K) = \alpha(C_K, t) \times (o - y) \times a(C_K)$$

The connection between the coarse-coding layer and the output unit is always sparse, since, for any input, only one cell from each grid in the lattice has non-zero activation. This is due to the competitive learning nature of the hierarchical clustering layer – only one cluster is activated at each level.

Simulation and Results

We use a hunting task as described earlier with synthetic data to evaluate our model. The data used in the task is shown in Figure 2. The hierarchies represent natural perceptual categories based on unsupervised learning with perceptual features, which are outputs from the HCL system as shown in Figure 1. We assume the agent has innate feature detectors that result in such perceptual categorization purely based on observing the objects without any hunting experience with the objects.

The functional interaction structure in this domain is represented in the two-dimensional table in Figure 2. A dark cell means the corresponding tool is good for hunting the prey and the agent will receive a reward of +1 if it chooses the action 'hunt'. The white cell means the corresponding tool is bad for hunting the prey and the agent will receive a reward of -1 if it chooses the action 'hunt'. The agent can alternatively choose the default action 'avoid', which will always give a 0 reward. We expect that the hierarchical categorizations will help the agent generalize its experience from a specific instance to similar combinations of objects. For example, the experience of hunting a rabbit with a longbow can be successfully generalized to hunting all four-legged animals with bow expect for one situation (longbow is not strong enough for hunting deer), so that both the category of Four-leg and Bow are useful abstractions. On the other hand, since there are variations within the group,

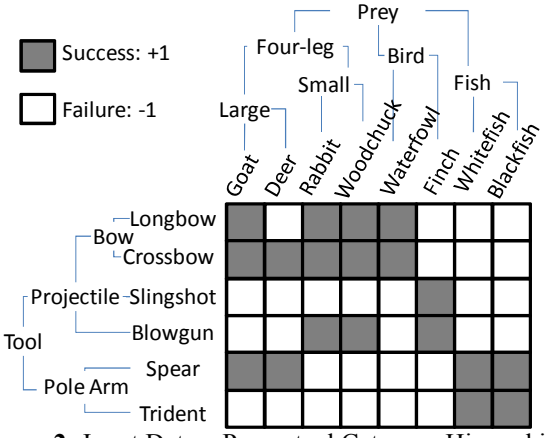


Figure 2: Input Data – Perceptual Category Hierarchies and Interaction Outcome Table

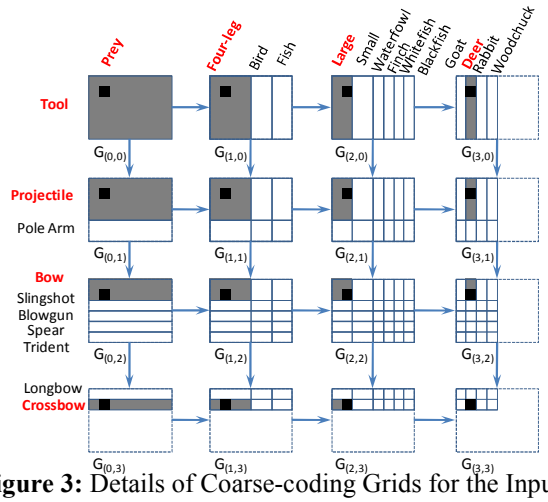


Figure 3: Details of Coarse-coding Grids for the Input (Deer, Crossbow)

we expect both concepts will be dominated by their subordinate categories in certain situations. In addition to trying to be close to reality, we designed the data so that it is complex, while at the same time, it has structure that tests specific aspects of the system, and it is simple enough to interpret the results.

To emphasize that the initial categorizations are based on innate perceptual features as opposed to taxonomic features, we use the labels such as Four-leg, Large, and Small, instead of Mammal, Ungulate, and Rodent to indicate they are perceptual categories. Birds have feathers, sharp beaks, and can fly. Fish all have similar shape, scales and swim in the water. A hierarchical clustering algorithm such as COBWB can automatically discover such statistical correlations among high dimensional perceptual features and incrementally build up a hierarchical structure as shown in Figure 2. Since we focus on the interaction between hierarchical categorization with RL, we did not include a detailed perceptual learning step in our simulation.

The effectiveness of tools with regard to prey may appear obvious to the reader. We make the assumption that the agent has no relevant prior knowledge to derive the effectiveness of a tool based on perceptual features. It has to incrementally learn the effectiveness of a tool for a prey through experience and build up the connections from perceptual similarities to functional outcomes piece by piece via the RL mechanism.

Figure 3 shows the details in the layer of coarse-coding rules for a specific input: hunting a deer with a crossbow. The black dots spatially represent the specific input in different grids. The gray areas represent the generalization effects when the more general rules fire. In this case, the agent receives a reward of +1 and each of the 16 rules participates in prediction and updating. Since a general rule (a larger cell) receives more training samples than a more specific rule (a smaller cell), it converges to the target value faster. On the other hand, the smaller cell will tend to compensate for the value in the context of the larger cell. The region with the dotted border in 7 of the grids on the lower and right borders means there are no more specific rules generated for those regions because it has already reached the leaf level of the categorization hierarchies.

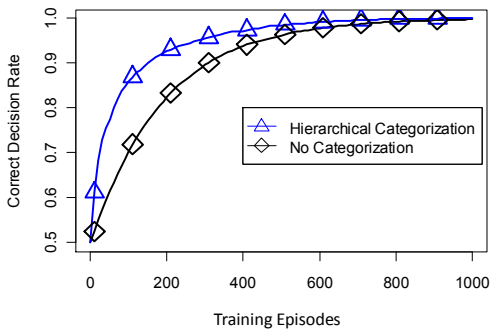


Figure 4: Learning with and without Hierarchical Categorization

Result 1: Category Learning to RL

Figure 4 compares the learning performance of hierarchical categorization with a baseline that uses the leaf level nodes without generalization. In the training data, there are two instances under each of the leaf nodes shown in Figure 2. For example, there are two instances of *Goat* that look different but have the same functional properties. Therefore, the size of the input space is: 16 (prey) times 12 (tools) equals 192. We evaluate the performance improvement during the course of learning. The agent is trained with random samples from the input space with replacement. The learning rate is set at 0.1. For a given amount of training episodes, we evaluate the rates of correct decisions it makes if it follows the policy derived from the current value function. The result shows that the model successfully integrates hierarchical categorization to speed RL.

Result 2: RL to Category Learning

Next, we analyze how functionally meaningful categorizations emerge from the process of RL. For a given input, there are multiple rules firing simultaneously, each coming from a different grid as shown in Figure 3. We define the *dominant rule* as the rule with the highest absolute value, or equivalently the *winning cell* with largest magnitude in its weight:

$$WinningCell = ArgMax_C\{\|w(C)\|\}$$

Correspondingly, we define the *dominant categories* as the categories associated with the dominant rule. In the hunting task for a specific input, there will be a dominant category for prey and a dominant category for tool. For example, the rule testing *Fish* and *Pole Arm* (the lower-right dark square consisting of 4 cells) dominates all the more specific rules that involve subtypes of *Fish* or subtypes of *Pole Arms* because it receives more training samples. It also dominates more general rules because there are inconsistent updates for those rules that cancel out each other. Consequently, the categories for *Fish* and *Pole Arm* are the dominant categories in these particular situations. The general principle is that a rule simultaneously maximizing both generality and consistency will dominate other rules. Intuitively, the associated dominant categories are more

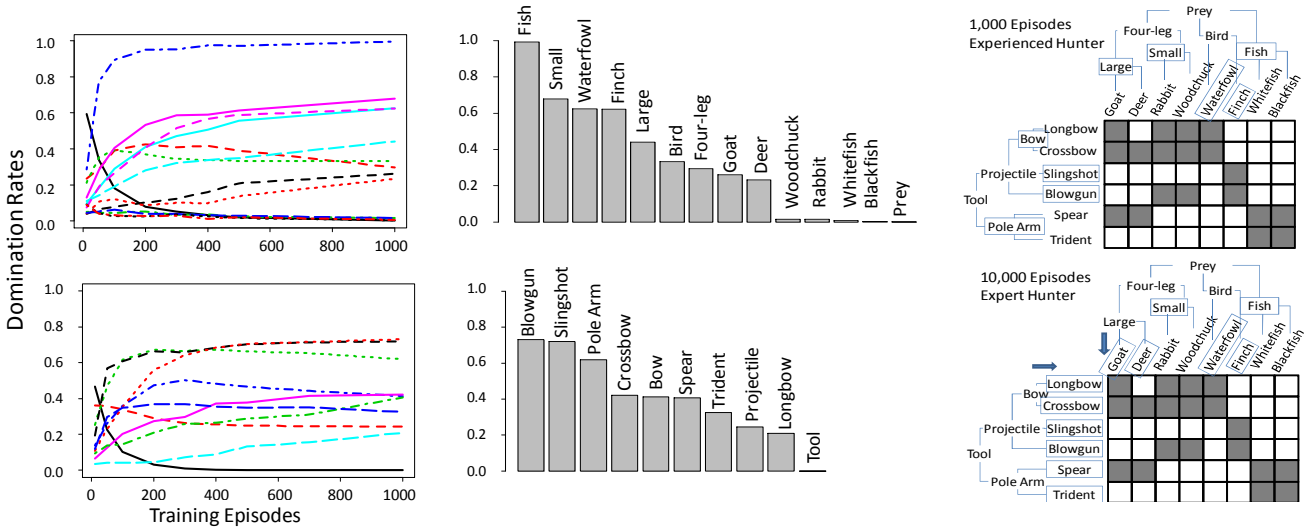


Figure 5: Dynamics of Domination Rates up-to 1,000 Episodes (on the left),
Domination Rates at 1,000 Episodes (in the middle),
Push-down of Basic-level Categories (boxed) with more Training Episodes – 1,000 vs. 10,000 (on the right)

functionally salient than their superordinate and subordinate categories, since they are the sources contributing to most of the decisions made by the voting mechanism. We use the overall *domination rates* across all possible inputs to measure the functional saliency of a category in a more context-free manner, which indicate how likely a category will become a basic-level category when there is no context effect.

The left side of Figure 5 shows the dynamics of domination rates up to 1,000 training episodes for all the categories of prey and tools. The trend is that the more general categories initially have higher domination rates because they cover more inputs and are trained with higher frequencies. As more experience is gained, consistent categories under a less consistent parent category have increasing domination rates (such as the two subtypes of birds), while less consistent superordinate categories become less dominant (such as the general category *Prey*, *Four-legged* animal, and *Bird*). On the other hand, a perceptual category that does not have any functional differences from other members under the same superordinate category does not arise as a functionally salient category (such as *Rabbit*, *Woodchuck* and the two subtypes of *Fish*). The middle of Figure 5 shows the domination rates after 1,000 training episodes. Since the ordering of inputs causes variations in the value of rules, we measure the mean domination rates across 300 independent learning trials, and the estimated standard errors for the means (not shown in the figure) are all less than 0.01. For example, the category for *Small Four-legged* animal dominates its superordinate and subordinate categories (including *Prey*, *Four-legged* animal, *Rabbit* and *Woodchuck*) in about 68% of all possible inputs. The category of *Rabbit* rarely dominates because its superordinate category completely captures the decision boundaries.

The right side of Figure 5 shows the context-free basic-level categories in boxes, which are the dominating categories along a path. The top figure shows the situation at 1,000 training episodes (for an experienced hunter) and the bottom figure at 10,000 episodes (for an expert hunter). The additional training experiences can "pull down" the basic-level towards more specific categories (indicated by the arrows). This effect arises naturally in our model and corresponds to the fact that a human domain expert possesses more specific basic-level vocabularies than a less experienced person.

Discussion

The general definition of category learning is the process that groups similar stimuli together so that similar responses can be made. Traditional cognitive theories of category learning include two competing views: the prototype view **(**Rosch 1973**)** and the exemplar view (Medin & Schaffer 1978). The prototype view is based on the principle of cognitive economy (Rosch 1978) and is supported by the existence of linguistic representations of abstract categories. However, there has been a shift of favor from the prototype towards the exemplar view because exemplar models provide superior empirical results in a variety of experimental settings (Nosofsky & Zaki 2002). A practical concern about the prototype view is that a prototype may fail to retain sufficient discriminative information. More recent models reconcile the two extreme forms and rely on representations at multiple abstraction levels (Vanpaemel & Storm 2008, Love *et al.* 2004).

Our model is consistent with both the prototype and exemplar views. In addition, it explicitly models the learning process and can deal with the more challenging situations where the input states involve multiple objects (such as the interaction between prey and tools). In terms of decision making, our model is more like exemplar based models, where the agent acquires information about specific inputs, and then makes generalizations to novel inputs based on perceptual similarity. In terms of category abstraction, our model agrees with prototype models. In particular, it

predicts a similar trend as in the phenomenon of basic-level category (Rosch 1978) where the most prominent categories (basic-level categories) reside in the middle of a categorization hierarchy.

Furthermore, our model predicts that category domination is context specific. For example, in the hunting context used as our demonstration task, *Pole Arm* is the dominant category if the sub-context is hunting *Fish* (all subtypes of *Pole Arms* are good for fishing). In a different context, however, *Spear* and *Trident* will dominate if the sub-context is hunting *Deer*. Our model explicitly supports the hypothesis that the "context-free" basic level categories, as described by Rosch, are the overall effects acquired across multiple functional contexts. Since the everyday activities related to common objects are largely the same across individuals, the context-free basic-level categories appear to be consistent as manifested in natural language.

Our model does not involve a dedicated process of selecting functional meaningful categories. Selection is achieved as an emerging by-product of the RL process. As a consequence, our model cannot explain certain types of category learning that rely on deliberate reasoning or higher degrees of abstractions, where the agent generalizes across instances that are perceptually distinctive but functionally similar. Such deliberate categorization is better described by rule based category learning model (Rouder & Ratcliff, 2006), or analogical reasoning processes such as in the structure-mapping engine (SME, Falkenhainer *et al.* 1989).

Conclusion

In this paper, we have presented the first computational model that integrates hierarchical category learning and RL in a general cognitive architecture, which can be used to coherently model basic-level effects, context effects and long-term learning effects in category learning. The unique feature of this model is that it simultaneously captures how categorization affects behavior adaptation, and how behavior adaptation influences categorization in a functional context. The general trends predicted by our model are consistent with existing category learning theories. Although the Soar-RL model has been successfully applied to match animal behavior data (Wang & Laird 2007), further empirical experiments are required to confirm its validity in our category learning model.

Acknowledgement

This research was supported in part by the Ground Robotics Reliability Center (GRRC) at the University of Michigan, with funding from government contract DoD-DoA W56H2V-04-2-0001 through the Joint Center for Robotics.

References

Ambros-Ingerson, J., Granger, R., & Lynch, G. (1990). Simulation of paleocortex performs hierarchical clustering. *Science*, *247*(4948), 1344-1348.

Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychol. Rev.*, *111*(4), 1036-1060.

Ashby, E. G., & Maddox, W. T. (2005). Human category learning. *Annu. Rev. Psychol.*, *56*, 149-178.

Falkenhainer, B., Forbus, K., & Gentner, D. (1989). The structure-mapping engine - algorithm and examples. *Artificial Intelligence*, *41*(1), 1-63.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*(2), 139-172.

Fu, W., & Anderson, J. R. (2006). From recurrent choice to skill learning: A reinforcement-learning model. *J. Exp. Psychol. Gen., 135*, 184-206.

Granger, R. (2006). Engines of the brain: the computational instruction set of human cognition. *AI Mag.*, *27*(2), 15-32.

Laird, J. E. (2008). Extending the Soar cognitive architecture. In *Proceeding of the 2008 Conference on Artificial General Intelligence*.

Langley, P., McKusick, K. B., Allen, J. A., Iba, W. F., & Thompson, K. (1991). A design for the ICARUS architecture. *SIGART Bull.*, *2*(4), 104-109.

Love, B. C., Medin, D. L., & Gureckis, T. M. (2004). SUSTAIN: A network model of category learning. *Psychol. Rev.*, *111*(2), 309-332.

Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychol. Rev.*, *85*(3), 207-238.

Nason, S., & Laird, J. E. (2005). Soar-RL: integrating reinforcement learning with Soar. *Cognitive Systems Research*, *6*(1), 51-59.

Nosofsky, R. A., & Zaki, S. R. (2002). Exemplar and prototype models revisited: response strategies, selective attention, and stimulus generalization. *J. Exp. Psychol. Learning*, 28(5), 924-940.

Quillian, M. R. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, *12*(5), 410-430.

Rosch, E. (1978). Principles of categorization. In *Cognition and Categorization* (pp. 27-48). John Wiley & Sons Inc.

Rosch, E. (1973). Natural categories. *Cognitive Psychology*, *4*(3), 328-350.

Rouder, J. N., & Ratcliff, R. (2006). Comparing exemplar- and rule-based theories of categorization. *Current Directions in Psychological Science*, *15*(1), 9-13.

Smith, J. D., Chapman, W. P., & Redford, J. S. (2010). Stages of category learning in monkeys (Macaca mulatta) and humans (Homo sapiens). *J. Exp. Psychol. Anim. B*, *36*(1), 39-53.

Sutton, R., & Barto, A. (1998). *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press.

Sutton, R. S. (1996). Generalization in reinforcement learning: successful examples using sparse coarse coding. *NIPS 8*, 1038-1044.

Vanpaemel, W., & Storms, G. (2008). In search of abstraction: The varying abstraction model of categorization. *Psychon. B. Rev.*, *15*(4), 732-749.

Wang, Y., & Laird, J. E. (2007). The importance of action history in decision making and reinforcement learning. In *Proceedings of the Eighth International Conference on Cognitive Modeling*. Ann Arbor, MI.

Interference and ACT-R: New evidence from the fan effect

Robert L. West (robert_west@carleton.ca)
Institute of Cognitive Science, Department of Psychology, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Aryn A. Pyke (apyke@ccs.carleton.ca)
Department of Psychology, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Matthew F. Rutledge-Taylor (mrtaylo2@connect.carleton.ca)
Institute of Cognitive Science, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Hana Lang (hlang@connect.carleton.ca)
Institute of Cognitive Science, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada

Abstract

We present data demonstrating that interference plays a role in the fan effect. We also show that this cannot be accounted for using ACT-R. An ACT-R model is fit to the data and we discuss options for altering the model to account for the data.

Keywords: interference; fan; spreading activation; ACT-R; memory; cognitive model.

Introduction

The fan effect refers to the fact that cues that are associated with more facts result in slower recall than cues that are associated with less facts. For example, in the study that established the fan effect, Anderson (1974) asked subjects to memorize facts about where various different characters had been seen. Subjects were then shown a cue with a character and a place and asked if it was true (i.e., if they occurred together in the set of facts subjects had memorized). Overall, the more places a character had been, the slower subjects were to confirm that it was either true or false. Also, subjects were slower to say false than they were to say true.

In the ACT-R architecture (Anderson & Lebiere 1998) the cue is held in a buffer as a chunk and each slot value of the cue spreads activation to chunks in declarative memory that have the same slot values. For example, if the cue chunk is person:hippy location:park, then hippy will spread activation to all chunks that have hippy as a slot value and park will spread activation to all chunks that have park as a slot value (note, the slot names do not play a role). The number of lines of activation leaving from a slot value in the cue is the fan of that slot value, and the fan of the cue is the sum of the fans of its slot values.

In ACT-R, the amount of activation spread from a cue to a chunk is theorized to be proportional to the number of past associations between slot values of the cue and the chunk. In the ACT-R architecture, the way of calculating this is based on an assumption that exposure to different facts has been counterbalanced, as in a psychology experiment (Anderson & Reder, 1999). If it is assumed that everything has been counterbalanced and the number of exposures per chunk is equal then the activation can just be divided evenly among the chunks. So, the higher the fan the lower the amount of activation delivered to each individual chunk (see Anderson & Reder, 1999, for how to use ACT-R when exposures have not been counterbalanced). Anderson and Reder (1999) modeled the fan effect in ACT-R by assuming that people retrieve the most active chunk and respond *true* (i.e., they have seen it before) if the retrieved chunk matches the cue, and *false* (i.e., they haven't seen it before) if it does not.

One consequence of this model is that only the spreading activation received by the chunk that is chosen affects the reaction time (RT). In other words, there is no interference from the activation of other chunks. However, as fan goes up so do the number of other chunks that receive activation. As part of a fan experiment we tested the effect of these "other" chunks to see if interference plays a role in the fan effect and how that might alter the ACT-R fan model.

Experimental Design

In our experiment we created false cues by taking a true fact and replacing one element with a false element. For example, if subjects had studied the fact that the *red hat is in the kitchen*, we could create a false cue by replacing *hat* with *pen*. Under these conditions the ACT-R model predicts that the fan of the false element of the cue will have no effect on retrieval time, unless the original fact is not retrieved. However, we performed simulations with the ACT-R fan model and found that in our experimental design, the chunk representing the original version of the fact always received the most activation, and therefore was always chosen (as far as we can see this is also true for other fan experiment designs, but it is possible to create more

extreme differences in fan where this would not be true). Related to this, the fan of the false element should also have no effect on the error rates. Although the ACT-R fan model does not explicitly model errors, errors would be due to noise and the retrieval threshold. This could conceivably interact with fan for the chunk that is being retrieved but the fan of the false element does not affect this chunk.

Method

Subjects

Twenty eight participants (11 males and 16 females: mean age 19.9 years, SD = 2.2) were recruited from introductory psychology courses a Carleton University to take part in the experiment. Participants received course credit as compensation for their time.

Procedure

The experiment was divided into three main phases: A study phase, a recall phase and a recognition phase. In the study phase each participant was assigned one of three unique sets of study sentences and was instructed to memorize the sentences in the list. Once the participant indicated that they were prepared to proceed, the recall portion of the experiment began.

The study set consisted of sixteen sentences of the form, "The *color thing* is in the *place*". The color term was one of ten colors; the thing was one of ten house-hold items; and the place was one of ten locations in/around a typical home. Very typical item/locations combinations, such as 'comb'/'bathroom', were not used in generating the study set sentences. Each term could have a fan of either 1 or 4. Thus, the four possible sentence fans were: 3, 6 9, and 12.

In the recall phase each participant was tested three times. Each test began with the participant trading the study set with the experimenter for a new list of sentences identical to the study set, but with one term from each sentence replaced with a blank, and the order of the sentences randomized. The participant's task was to correctly fill-in each of the blanks with the missing word. The participant was given as much time as he or she needed. Once finished, the experimenter recorded the number of correct responses and for each error, provided the correct missing word to the participant. The participant was then given the opportunity to review the study set before being tested again. The three tests were balanced such that each term from each sentence in the study set was replaced with a blank exactly once. After the third iteration the recognition phase began.

The recognition phase of the experiment was conducted on a computer using Experiment Builder (by SR Research). The participant's task was to correctly judge whether sentences presented in the middle of a 17" CRT display were members of the study set, or not. Accuracy and reaction time data were recorded for each trial.

Each participant was presented with 96 test sentences, which consisted of three exposures to each of the study set sentences, and 48 sentences that were not from the study set. The participants were told that they should consider sentences from the study set to be *true*, while all others should be considered *false*. Each false sentence was generated by swapping one of the three terms from a true sentence with another term from the same category (e.g., color, thing, or place) and with the same fan. Each true sentence was used to generate three different false sentences. Thus, for each exposure to a true sentence there was a false sentence with the identical fan. Once the test sentence appeared the participant would indicate if the sentence was in the study set by hitting the 'z' key, or if it was not by hitting the '/' key.

Results

The data from one of the participants was excluded from the results presented below. This was due to the fact that this participant's performance was significantly poorer than all other participants by a large margin ($P < 0.001$). The results below reflect the data collected from the remaining 27 participants. By the end of the third iteration of the recall phase the participants were correctly completing the sentences 91.4 percent of the time. The results of the recognition phase replicated the fan effect. These results are reported in Rutledge-Taylor, Pyke, West, & Lang (2010). However, in this paper we will focus on the results related to the predictions described above and fitting an ACT-R model to the data.

The hallmark of a good scientific theory is that it makes precise, falsifiable predictions. Many theories in Psychology fail to meet this criterion. However, because ACT-R is precisely specified, models built in ACT-R are more readily falsifiable. To test the predictions of the ACT-R fan model (Anderson & Reder, 1999) concerning the fan of the false items we ran an ANOVA testing for the effect of the fan of the false items on RT and error rate. RT was significantly higher when the fan of the false item was equal to 4 than when it was equal to 1 ($P<0.001$). The error rate was also significantly higher when the fan of the false item was equal to 4 than when it was equal to 1 ($P<0.001$). We also ran a correlation between the fan of the false items and RT, with the fans of the true items partialled out. We found a significant correlation of $r=0.156$ ($P<0.001$, one tailed). Similarly, we ran a correlation between the fan of the false items and % errors, with the fans of the true items partialled out. Here we also found a significant correlation of $r=0.193$ ($P<0.001$, one tailed). The size of these correlations was roughly similar to the same correlations done with the fan of the true items.

Contrary to the predictions of the ACT-R fan model, we found that the fan of the false items significantly affected RT such that a larger fan led to slower responses (see Figure 1). Consistent with this and also contrary to the predictions of the model, we found that the fan of the false element significantly affected the error rate such that a larger fan led to more errors (see Figure 2). These results indicate that interference from the false item plays a role in the fan effect.

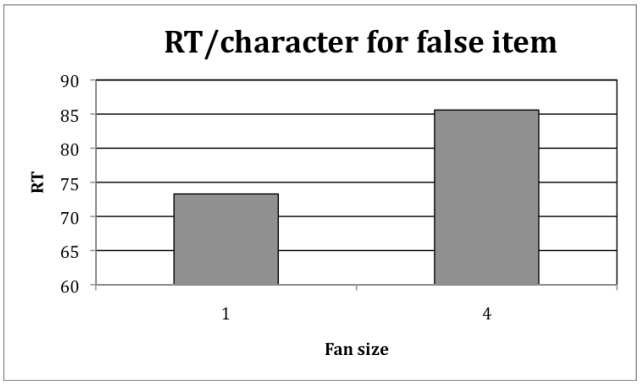


Figure 1: Reaction time in msec/character for responding false to a false cue as a function of the fan of the false item in the cue.

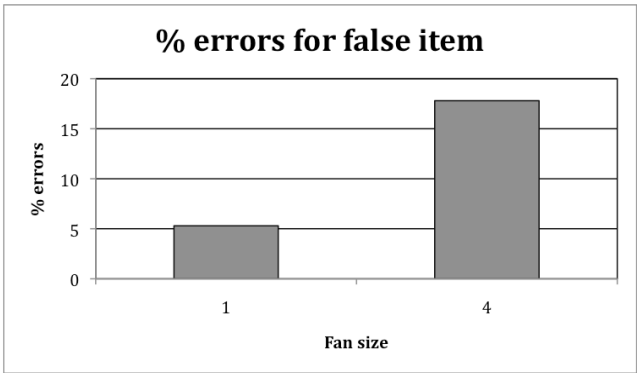


Figure 2: Percent errors for responding false to a false cue as a function of the fan of the false item in the cue.

.

Model Evaluation

Although falsification of a model is sometimes viewed as a bad thing, falsification actually shows that a model was well specified. Falsification also creates an opportunity to move toward a better model. To this end we fit the ACT-R fan model to our data. Anderson and Reder (1999) used the following function to calculate RT:

$$RT=I+Fe^{-Ai}$$

Where *F* is a scaling constant for time, *I* is a constant representing how long it takes subjects to make their response after they know it, *e* is the base for natural logarithms and *A* is the activation of the chunk (which includes spreading activation). Activation was calculated as:

$$A=B+S$$

Where *B* is base level activation and *S* is spreading activation. We fitted the Anderson and Reder (1999) ACT-R fan model to our data using identical parameter values, except that we had to increase *S* slightly from 1.45 to 2 to avoid getting negative activation values. As in Anderson and Reder (1999), *B* was set to zero because it trades off with *S*.

We eventually figured out that the slight difference in *S* was because we used the current method of calculating fan size in ACT-R 6, which is to add 1 to the fan of each item in the cue to represent the match between that item and a chunk in memory representing that item. For example, 1 would be added to the fan of cup because it is assumed that we all have a chunk in declarative memory representing cup. In contrast, Anderson and Reder (1999) calculated the results without adding 1 to fans of the items in the cue. Whether or not to do this is an interesting issue. However, we recalculated our results without adding 1 and found it made very little difference to our results or conclusions.

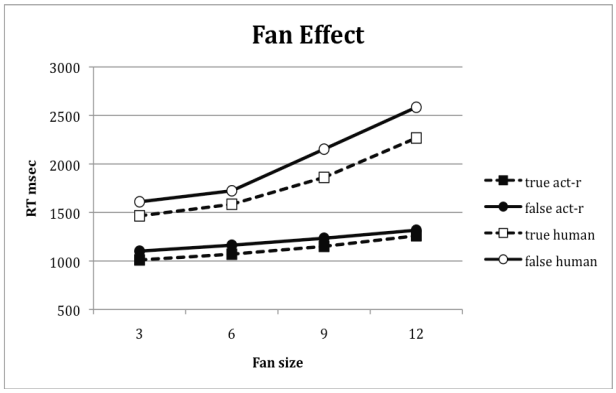


Figure 3: The original Anderson & Reder (1999) ACT-R fan model fit to our data.

Figure 3 shows the fit of the original ACT-R fan model to our data. The fact that the model predicts an overall lower RT is not significant as it can be accounted for by assuming our subjects took longer to press the true/false keys, which can be modeled by increasing the *I* parameter. However, the shape of the functions and the relationship between the functions is clearly different. The human data shows a clear upward curve that the model does not and the model RTs converge as fan increases wile the human data diverges.

Figure 4 shows the original fan effect data from Anderson and Reder (1999) re-plotted. Note that it shows the same divergence and upward curve. In fact, the original ACT-R model for this data (faithfully recreated and shown in Figure 5) also shows a slight upward curve for the *true* cues, but not for the *false* cues. Also, as with our data, the *false* function diverges from the *true* function as fan goes up. However, it is important to keep in mind the scale of the graphs and realize that these effects are much smaller in the Anderson and Reder (1999) data and may not even be real, although, the consistency of this result across conditions and studies indicates that we should take it seriously.

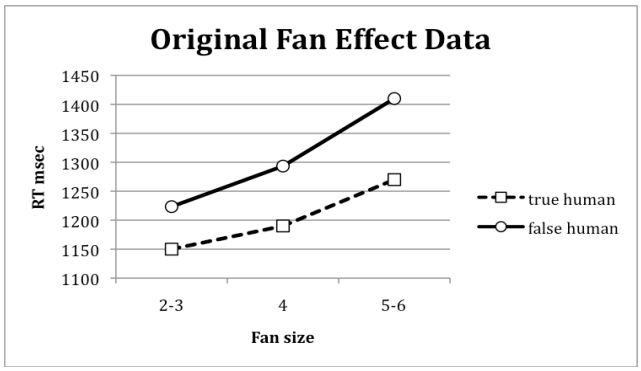


Figure 4. Re-plotted data from Anderson and Reder (1999)

An Alternative Model

Next we addressed the issue of the parameter values. Specifically, we wanted to know if the ACT-R model could be made to fit the data. The only way that we could find to fit the data was to use the latency exponent parameter (*f*) that is available in ACT-R 6. This parameter, which has rarely been used, changes the RT function to:

$$RT=I+Fe^{-(f*A)}$$

By setting *f=3* and increasing *F* from *613* to *2000* we obtained a good fit to the data (see Figure 6 - note, that the *I* parameter could be increased to overlap the functions but it is easier to see this way). Increasing *f* lowers overall RT, so increasing *F* can be viewed as a way of compensating for this. The other effect of raising *f* was to increase the acceleration of the rate at which lowering activation raised RT. We will refer to this as the ACT-R(*f*) model (see Figure 6). However, please note that this model violates the ACT-R modeling convention of using establish parameter values unless you have a justification (Anderson & Lebiere, 1998).

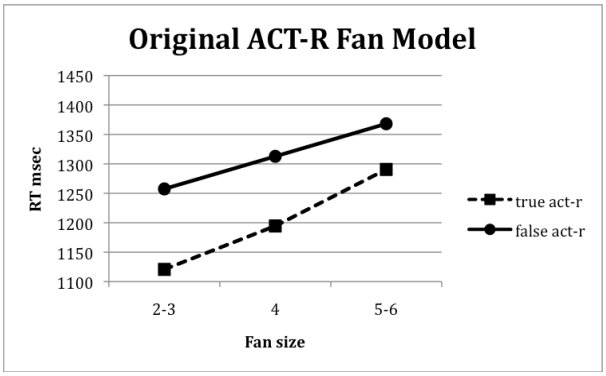


Figure 5. A re-creation of the ACT-R fan model from Anderson and Reder (1999)

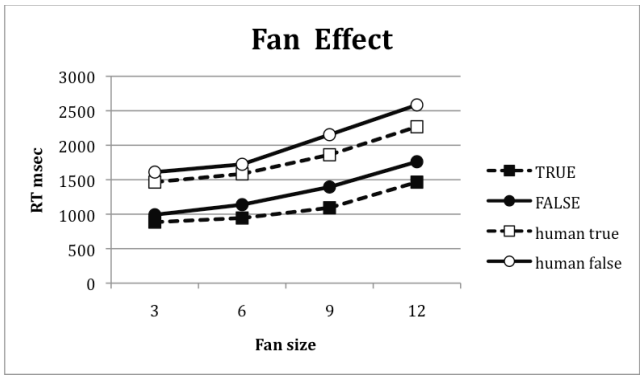


Figure 6: The ACT-R(f) model fit to our data (note, that the *I* parameter could be increased to overlap the functions but it is easier to see this way).

Rationalizing the alternative model

There are three ways we can interpret the ACT-R(*f*) fan model. We know that it cannot account for our finding that the fan of the false item in the cue affects RT and % error any better than the normal ACT-R fan model. However, it is possible to interpret the manipulation of *f* as representing the aggregate effect of interference. In this case, *f* would be related to the total effect of interference in the task. If we assume that our use of more cue items and higher fans produced greater overall levels of interference, then the fact that our results show a more pronounced nonlinear effect than the Anderson and Reder (1999) results could be modeled by increasing *f* to represent higher levels of interference. In this sense, ACT-R could be adjusted to account for the presence of interference but could not be said to include a (process) model of interference. More studies would be required to see if *f* actually does function this way.

A less charitable approach to understanding the ACT-R(*f*) model would be to point out that adding *f* to a model that already has a lot of parameters creates a system capable of fitting a lot of different functions. We had no principled reason to adjust *f* and found that it worked as part of a parameter tweaking process that involved all of the available parameters. So possibly the fit of this model is merely fortuitous.

A third, more constructive way of viewing it is to see the manipulation of *f* as a proxy for an additional mechanism or process - in this case, interference. Although ACT-R does not include an interference mechanism, modifications have been introduced to do this. For example, the spacing effect modification of Pavlik and Anderson (2005) assumes that interference plays a role in order to account for the spacing effect in memory. Similarly, the semantic interference modification proposed by Van Maanen & Van Rijn (2007) assumes a form of interference to account for the Stroop effect. Likewise, our findings indicate the need for an explicit model of interference in ACT-R. A simple way of doing this that is consistent with our manipulation of *f* is to introduce a penalty that reduces activation based on the total

fan of the information in the cue – the higher the overall fan, the greater the penalty for all chunks receiving spreading activation. We could create such a function but it would not be meaningful at this point since it would be custom made to fit our data. Essentially, this would have the same effect as raising *f*, but the effect would be tied to the overall fan and therefore would account for our finding that the fan of the false item in the cue affects RT and % error.

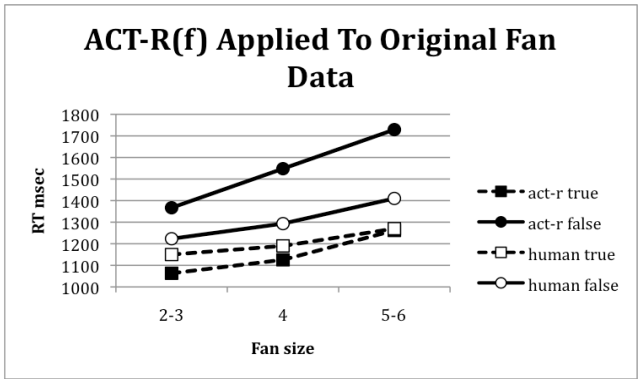


Figure 7. The ACT-R(f) model applied to the data from Anderson and Reder (1999).

Model Re-Evaluation

To gain further insight into the ACT-R(*f*) model we applied those parameter settings to our recreation of the Anderson and Reder (1999) fan model. The results are illustrated in Figure 7. The *true* results actually produce a reasonable fit to the data but the *false* results clearly do not fit. This could be because the fit of the ACT-R(*f*) model to our data was merely fortuitous, or it could be because higher *f* values are only appropriate when interference is higher, as suggested above.

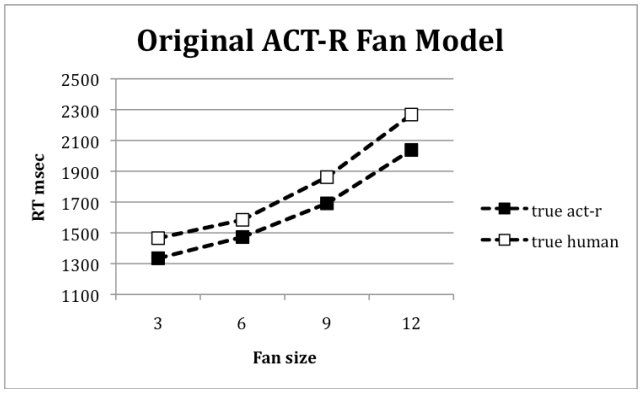


Figure 8. The Anderson and Reder (1999) ACT-R fan model fit to our data for correctly identifying true cues only.

Based on our experimental findings showing that the ACT-R fan model for correctly identifying *false* cues cannot be correct, we also tried fitting Anderson and Reder's (1999) fan model to our data for the *true* results only (see Figure 8). Without having to fit the *false* data we were able to get a good fit by adjusting only *F* and *I* (*F*=1000; *I*=1100; *S*=1.45; similar to Anderson and Reder we did not add 1 when calculating the fan). This is much less problematic because it avoids adjusting *f*, which is almost never altered in ACT-R modeling. Also, it is important to remember that there is variability associated with the human data so it is likely that a single intermediate value of *F* could be used to obtain a reasonable fit to our data and Anderson and Reder's (1999) data.

Conclusions

Our results show that the ACT-R fan model for correctly identifying false cues cannot be completely correct. Also, although fitting ACT-R to our data was possible, it was also problematic because it required unprecedented alterations to the parameter values as well as assumptions about the meaning of those alterations that remain untested. However, when we did *not* try to fit the ACT-R fan model for correctly identifying false cues, the ACT-R fan model for correctly identifying true cues fit our data well, without any problematic parameter alterations. Based on this, it appears most likely that the problem lies with the assumptions and processes behind the ACT-R fan model for correctly identifying false cues.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6*, 451-474.

Anderson, J. R. & Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: Erlbaum.

Anderson, J. R. & Reder, L. M. (1999). The fan effect: New results and new theories. *Journal of Experimental Psychology: General*, 128, 186-197.

Pavlik, P. I., Jr., & Anderson, J. R. (2005). Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science, 29*, 559-586.

Rutledge-Taylor, Pyke, West, & Lang (2010). Modeling a three term fan effect. *Proceedings of the International Conference on Cognitive Modeling*.

Van Maanen, L., & Van Rijn, H. (2007). An accumulator model of semantic interference. Cognitive Systems Research, 8(3), 174-181.

An Online Database of ACT-R Parameters: Towards a Transparent Community-based Approach to Model Development

Tsunhin John Wong[1] (jwong@mpib-berlin.mpg.de)

Edward T. Cokely[1,2] (cokely@mpib-berlin.mpg.de)

Lael J. Schooler[1] (schooler@mpib-berlin.mpg.de)

Max Planck Institute for Human Development, Lentzeallee 94 14195 Berlin, Germany[1]
Michigan Technological University, Cognitive & Learning Sciences, Townsend Drive, Houghton, MI 49931[2]

Abstract

We present a database that provides an interface for the ACT-R modeling community to interact with each other (http://www-abc.mpib-berlin.mpg.de/actrdb/). The database includes estimated values of ACT-R parameters from a wide range of ACT-R modeling studies, selected from the studies available on the ACT-R website. It serves as a tool to query studies and estimated values for ACT-R parameters, providing the exact range of values for each of the available free numerical parameters. In short, the database supports an alternative community-based approach to manage the challenges associated with parameter estimation for complex cognitive architectures like ACT-R.

Keywords: ACT-R; modeling; parameter.

Managing Parameters for ACT-R Models

Unified theories of cognition allow us to approach mechanisms of human cognition in a holistic, cumulative manner (Simon & Newell, 1973). Among the existing unified theories of cognition, ACT-R is one of the most widely used architectures, producing the largest body of sustained research and application. In order to study a wide range of cognitive mechanisms, ACT-R includes a variety of modifiable parameters. While these parameters enable flexibility they also result in fundamental challenges.

Wexler (1978) criticized the early framework of the ACT research program (Anderson, 1976), stating that "There is no explanatory power in ACT because there are no restrictions on human abilities". He also posited that "the general problem with ACT is (its flexibility), it is simply so weak that there is no way to find evidence for or against it". About twenty years later, Pashler and Roberts (2000, 2002) again brought these concerns to the fore, arguing that the practice of using good fits as major evidence for complex theories is "rotten to core". Indeed, goodness-of-fit metrics remain a very common means of model validation. These concerns not only hold when criticizing ACT-R and some other unified models, but also address a wide-spread misuse of goodness-of-fits as key evidence in psychology. Sound scientific theory requires that models not only fit but also predict data (Gigerenzer, 1998; Gigerenzer & Brighton, 2009). How can modelers of the ACT-R architecture deal with these concerns about parameter estimation and model fitting?

There have been some attempts to understand the relation among ACT-R parameters that result from parameter fitting. For example, Anderson, Bothell, Lebiere, and Matessa (1998) suggested that there is a systematic linear relationship between the estimated values of activation thresholds and the logarithm of estimated latency factors. Their data also implied that estimated values of these parameters are exceedingly regular. To date, however, there has been no meta-analytic assessment to evaluate whether there is any sustained regularity of these estimated parameters for ACT-R models across other published studies.

Computational cognitive models are often evaluated by their fit and generalizability. These properties of a model are related to two aspects of model complexity: (1) number of parameters and (2) the functional forms of computation. In part, such evaluations seek to evaluate the extent to which noise is unnecessarily captured (Pitt, Myung, & Zhang, 2002; Oaksford, 2002). Using cross-validation, Taatgen, van Rijn, and Anderson (2007) estimated parameters of a base-model once and then made use of these estimated values throughout subsequent models. This study exemplifies a strict practice that allows minimal parameter estimation; however, like many ACT-R studies, the work of Taatgen et al. still relied on superior goodness-of-fits as the major support for their proposed models.

The latest ACT-R architecture version 6.0 has 62 free parameters with numerical values, together with the flexibility of mapping these parameters to tailor-made handlers and tens of other non-numerical parameters. Different instantiations of specific ACT-R models do not typically require setting and optimizing all these numerical parameters, as default values are provided. However, our analyses of a large and representative sample of ACT-R studies indicates that on average each ACT-R model modifies nearly six free numerical parameters for better model fitting. Moreover, many of these studies added task-specific parameters.

Parameter Estimations by "Wisdom of the crowd"

To provide another path to parameter estimation-free modeling, we developed a database to collect estimated and modified ACT-R parameters from the ACT-R modeling community. With this database, we hope to facilitate comparisons of ACT-R parameters by drawing on the wisdom of the crowd. Accordingly, we catalog previous studies that have provided estimates together with corresponding parameters. The database makes it relatively easy to determine whether a particular newly estimated value falls within a reasonable range according to previous related studies. Moreover, this database also serves to provide some meta-analytical data on the variety and ranges of selected parameters across a large representative set of studies.

Taatgen, et al. (2007) have argued that the ideal goal for an ACT-R modeler is to fix all parameters: A modeler should not estimate any parameter during modeling. One key goal of this current project is to collect and compile data from a representative range of published ACT-R models (with exact values for estimated parameters) in a sustainable database to assist ongoing modeling projects. The online database provides several potentially useful functions including updated information about the means and the medians of the existing free numerical parameters. Moreover, the database has been designed to be scalable so as to be readily extended to other models and tasks. In what follows, we describe the database and briefly review some functions and findings. We close with a discussion of potential applications and implications.

Method

We started with the studies and models that made use of the ACT-R architecture listed on the ACT-R website (http://act-r.psy.cmu.edu/). From this online repository of ACT-R studies, we selected all studies that have made both their ACT-R models and manuscripts available; a total of 44 studies were included at the time of data collection. From these models, we collected the information about the version of ACT-R architecture that was used as well as the particular ACT-R parameters that were modified. We also collected information about the deprecated parameters from previous versions of ACT-R and other task-specific parameters that these models made use of.

Overview of functions of the database

The database can be accessed through an Internet-interface at the URL:

http://www-abc.mpib-berlin.mpg.de/actrdb/

The Internet-interface was tested and works with most of the popular website browsers, such as Firefox 3+, Safari 4, Internet Explorer 8+, and Opera. Along with the information about ACT-R parameters, our database serves at least four main functions:

1. Monitoring how frequently parameters are modified.
2. Obtaining parameter means, medians and distributions.
3. Searching the keyword descriptions of ACT-R studies in the database.
4. Collecting fields of study and other information related to ACT-R parameter estimations.

Below we describe the basic functions of the database.

Frequency graph In the middle of the frontpage, there is a frequency graph listing all the numerical parameters that have been modified by at least one study in the database. Layout of the frequency graph is arranged so that the modified parameters are listed in descending order of frequency from the bottom to the top (Figure 2).

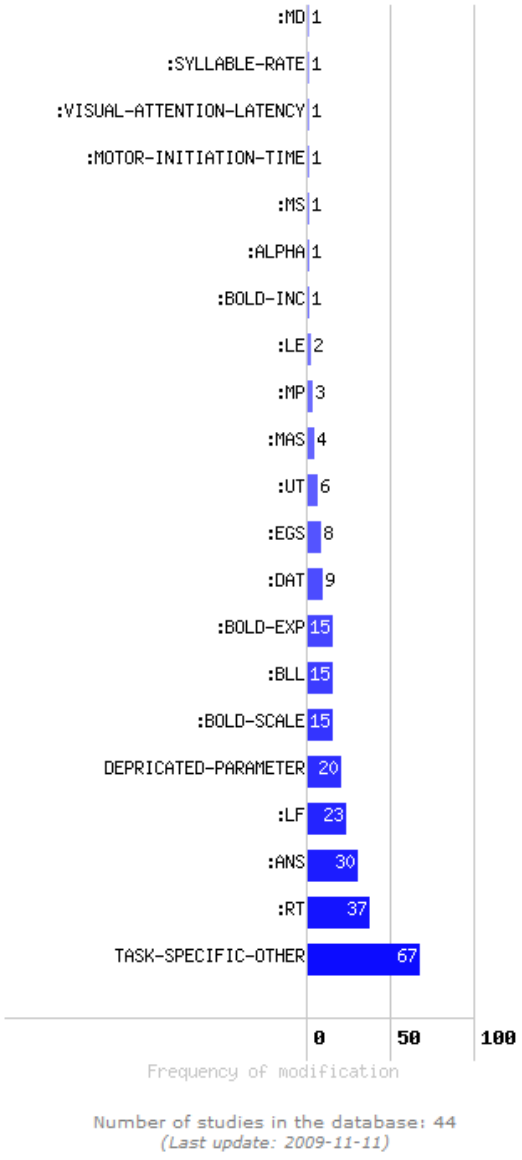


Figure 1: Frequency of modifications of ACT-R parameters.

Frontpage function buttons At the top of the frontpage of the database there are functional buttons labeled "Query parameters", "list studies", and "Enter your ACT-R study". These buttons provide access to the major ways to interact with the database (See Figure 2). A 'Home' button returns the user to the portal frontpage.

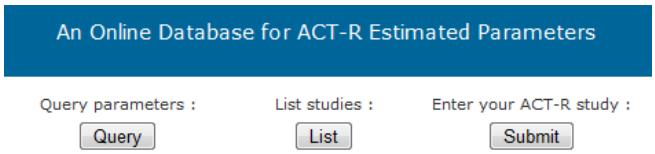


Figure 2: Major functional buttons at the top of frontpage.

Performing keyword search in the database At the bottom of the frontpage there is a search box where users can perform keyword searches or exact title searches (See Figure 3).

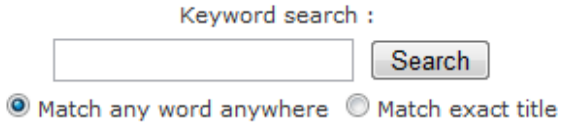


Figure 3: Keyword search or title search in the database.

Query parameters By pressing the query button a user can query any ACT-R parameter for any particular version of ACT-R in the database, using a drop-down menu (See Figure 4).

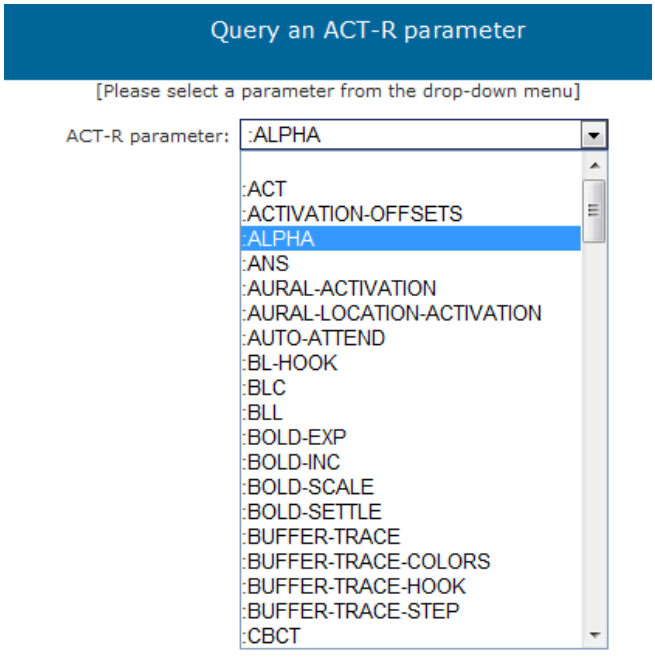


Figure 4: A drop-down menu to query ACT-R parameters.

After a parameter is chosen, the studies in the database that modified the particular parameter are displayed together with the mean and the median. The database also provides a graph describing the distribution of its modification among studies listed as well as the default values and the equation(s), if any (See Figure 5).

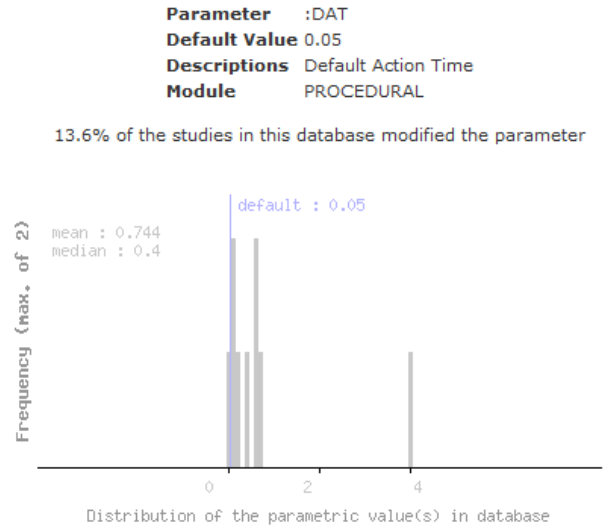


Figure 5: Information about the ACT-R parameter :DAT.

Results of listing or searching the database By pressing the 'List' button, or by performing a search on the frontpage, a user will reach a list of studies (See Figure 6).

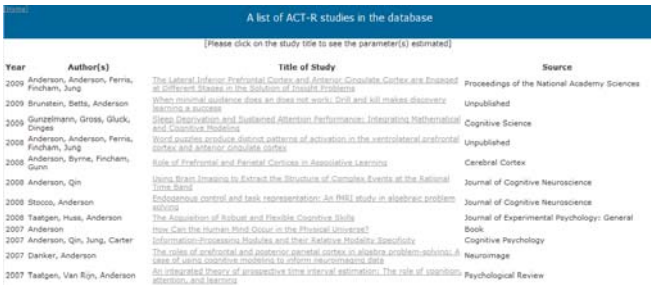


Figure 6: A list of ACT-R studies displayed after pressing the 'List' button or performing a search.

By further clicking on the title of a study specific information about parameter modifications of that study will be displayed (See Figure 7).

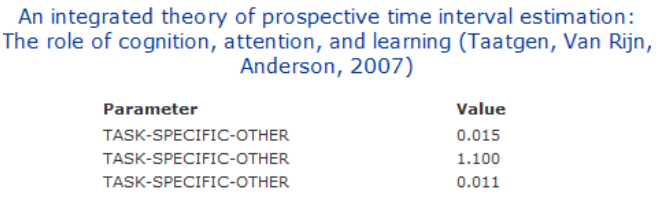


Figure 7: Information about parameter modifications.

Submit your model By pressing the 'Submit' button a user will reach the interface for entering information about parameter modifications of an ACT-R study (See Figure 8).

The interface is designed so as to guide the user through reporting their study in a step-by-step manner. By allowing ACT-R modelers to interact through our online database (providing their own estimated parametric values, modified values, and comments on their entries and models) we provide a more sustainable 'living' archive that benefits from the 'wisdom of crowds'. Readers are welcome to try out the database and provide feedback.

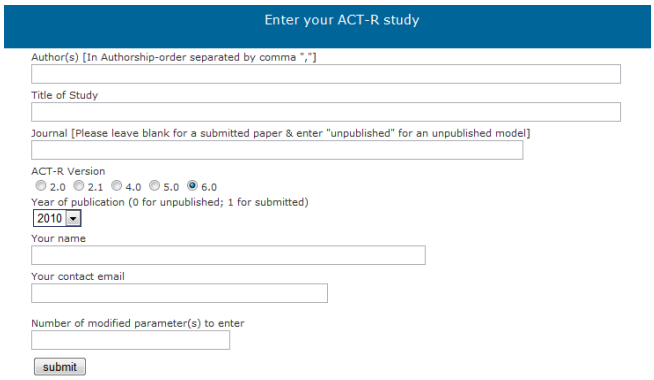


Figure 8: An interface to enter information about an ACT-R modeling study.

Results and Discussion

A brief report of some notable ACT-R parameters

At the time of publication, 44 studies were included in the database Together with a total of 261 instances of parameter modifications or estimations. On average each study modified 5.93 parameters. Among the ACT-R parameters that were modified in these studies, the three most frequently modified were :RT, :ANS, and :LF. The two ACT-R parameters :BOLD-EXP and :RT have the widest ranges of modified values among all parameters (See Table 1).

Table 1: The Most Frequently Modified ACT-R Parameters and Parameters with The Widest Range of Values

ACT-R Parameter	Default value	Description	Frequency modified
:ANS	Nil	Activation noise of chunks	30
:LF	1	Latency factor of chunks retrieval	23
:RT	0	Retrieval threshold of chunks	37
:BOLD-EXP	6	Exponential parameter for computing the BOLD response.	15

Note: For detailed descriptions of all the ACT-R parameters, we refer the interested reader to the ACT-R website (http://act-r.psy.cmu.edu/), Anderson (2007), and Anderson & Lebiere (1998).

Applications

Anderson et al. (1998) demonstrated that there are systematic variations between τ (:RT) and F (:LF) across studies. Unfortunately, not all the parameters in ACT-R have received this level of attention. As ACT-R continues to develop, it will acquire even more parameters. To help manage obstacles and challenges associated with such growth, our online database may provide a useful and convenient way for the ACT-R community to interact with each other and monitor these parameters. In the long run, by flagging frequently monitored parameters the database may point to weaknesses in the theory. In the short run, the database provides an overview of the parameter space.

To illustrate, when a modeler wants to study a phenomenon that requires estimation of an ACT-R parameter, this database serves as a portal to get an overview of the parameter in question with just a few mouse-clicks. With a keyword search about the phenomenon one can get a list of related modeling studies. When directly querying the parameter, the database provides studies that have modified the parameter from its default value, alongside with the means, medians, and default value (if any) on a distribution graph. This provides the modeler with a transparent window onto what was previously opaque information about what parameter values other ACT-R modelers were using.

Beyond fixing exact parameters, we also expect that the database can simplify much of the procedure used to estimate ranges of ACT-R parameters. The database can provide information about ways and approaches for capturing individual differences (e.g. age, abilities), environmental differences, and task differences (e.g. vigilance). For example, the default action time (:DAT), which is set at 0.05 second, dictates the basic firing speed of a procedure in an ACT-R model. While it is standard to use to default values when possible, there are indications that age (Mata, Schooler, & Rieskamp, 2007) and environmental factors (Gunzelmann, Gross, Gluck, & Dinges, 2009) may alter this basic firing speed. Another example is the retrieval threshold parameter (:RT), which is normally set to zero but can be expressed instead as a logistic function with a range of possible values, reflecting forgetting (Schooler & Hertwig, 2005). In these instances, using the interactive database to gather information provides a way to better monitor and estimate the most reasonable (or common) parameters of variation for human speed of processing.

Implications for parameter estimations in ACT-R

We setup the database in response to some important concerns stemming from the general problems of parameter estimation associated with a framework as complex as ACT-R. By setting up this database, we appeal to the 'wisdom of the crowd' among ACT-R modelers. In ongoing work we are testing the *median parameter hypothesis:* The parameterization of ACT-R based on the median estimated

ACT-R parameter values across all studies will fare better in predicting performance when compared to the parameterization that was used for each particular study. We could also imagine searching for a set of parameters that gives the best fit to all the studies in the database. It is our hope that these efforts may bring ACT-R modelers closer to true "zero-parameter fits".

Setting parameter values *a priori* to plausible values constrains overly flexible models by restricting the range of a model's predictions. This should lead to more accurate and perhaps more useful predictions of human performance patterns. A possible further development is to estimate a recommended range of values for every ACT-R parameter that correspond to human cognitive limitations. Finding such a correspondence would be in line with practices used in the human factor community, where limits of human performance are essential inputs for system design.

Conclusions

The major aim of this database is to provide a collaborative interface for ACT-R modelers to document and monitor values of ACT-R parameters in an efficient and sustainable way. By making use of the "wisdom of the crowd", ACT-R modelers can minimize model flexibility and increase the generalizability of their models. This can also be seen as a natural experiment concerning how best to estimate parameters in a social manner. By using a database of parameters that encourages generalizability and penalizes flexibility the ACT-R community might move closer to answering Newell's beautiful call for a truly unified theory of cognition.

References

Anderson, J. R. (1976). *Language, Memory, and thought*. New Jersey: Lawrence Erlbaum.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press, USA.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Lawrence Erlbaum.Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language, 38*, 341–380.

Gigerenzer, G. (1998). We need statistical thinking, not statistical rituals. *Behavorial and Brain Sciences, 21*, 199-200.

Gigerenzer, G., & Brighton, H. J. (2009). Homo heuristicus: Why biased minds make better inferences. *Topics in Cognitive Science, 1*, 107-143.

Gunzelmann, G., Gross, J. B., Gluck, K. A., & Dinges, D. F. (2009). Sleep Deprivation and Sustained Attention Performance: Integrating Mathematical and Cognitive Modeling. *Cognitive Science, 33(5)*, 880-910.

Mata, R., Schooler, L., & Rieskamp, J. (2007). The aging decision maker: cognitive aging and the adaptive selection of decision strategies. *Psychology and Aging, 22(4)*, 796-810.

Oaksford, M. (2002). How does it fit? *Trends in Cognitive Sciences, 6*(10), 412-413.

Pitt, M. A., Myung, I. J., & Zhang, S. (2002). Toward a method of selecting among computational models of cognition. *Psychological Review*, *109*(3), 472–491.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review, 107(2)*, 358–367.

Roberts, S., & Pashler, H. (2002). Reply to Rodgers and Rowe (2002). *Psychological Review, 109(3)*, 605–607.

Schooler, L. J., & Hertwig, R. (2005). How Forgetting Aids Heuristic Inference. *Psychological Review*, 112(3), 610–628.

Taatgen, N. A., Van Rijn, H., & Anderson, J. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention, and learning. *Psychological Review, 114(3)*, 577–598.

Wexler, K. (1978). A review of John R. Anderson's Language, Memory, and Thought. *Cognition, 6*, 327-351.

Locating the Neural Correlates of the Problem State Resource: Analyzing fMRI Data on the Basis of a Computational Model

Jelmer P. Borst (jpborst@ai.rug.nl), Niels A. Taatgen (niels@ai.rug.nl)
Department of Artificial Intelligence, University of Groningen, the Netherlands

Hedderik van Rijn (d.h.van.rijn@rug.nl)
Department of Psychology, University of Groningen, the Netherlands

Keywords: fMRI analysis; cognitive modeling; problem state.

Introduction

Multitasking often has to be investigated with experiments using complex tasks. An example is our research on the 'bottleneck' role of the problem state resource (Borst, Taatgen, & Van Rijn, 2010). The problem state resource is the part of working memory that is used to store intermediate results. Previously, we have shown that its capacity is limited to one element. Because we were interested in finding the neural correlates of the problem state resource, and fMRI data of complex tasks are difficult to analyze with classical analysis methods, we developed a novel, computational-model-based fMRI analysis method. We show that this method can be used to analyze complex tasks by locating the brain area responsible for maintaining problem states: the inferior parietal lobule.

Methods

Our participants were asked to perform a 'triple-task' in the fMRI scanner: They solved multi-column subtraction problems, entered text, and performed a listening comprehension task concurrently. Both the subtraction task and the text entry task had two versions: an easy version without problem state usage and a hard version with problem state usage. Due to the problem state bottleneck, problem states had to be replaced constantly in the hard subtraction – hard text entry condition (Borst et al., 2010). This should lead to considerably more activity in brain areas associated to the problem state in the hard – hard condition than in the other conditions. That is, we predicted an over-additive interaction effect.

This type of complex task is difficult to analyze with classical fMRI analysis methods that assume 'pure insertion'. In such a complex task cognitive resources are used at different time points in each trial, while pure insertion methods assume that a resource is active in one condition but not in the other conditions. As an alternative analysis method, we fit a computational model developed using ACT-R (Anderson, 2007) and Threaded Cognition (Salvucci & Taatgen, 2008) to the behavioral data, and subsequently regressed the model's problem state activity against the fMRI data to find regions that are sensitive to problem state activity. This gives a much finer-grained stimulus function than classical methods, as we use model behavior within a single trial. Figure 1a and 1b give a concrete example of what this means over the course of four trials in our experiment. Figure 1a shows the single stimulus function of the problem state resource that was used for the new model-based analysis and Figure 1b shows the four stimulus functions that are needed for the classical fMRI analysis of an interaction effect.

Results & Discussion

The results of the analyses are displayed in Figure 1c (model-based method) and 1d (classical method). First, the results show that the model-based analysis method outperformed the classical method: it enabled us to find the neural correlates of the problem state resource, while the classical method did not yield any significant results. Secondly, the results show that the problem state resource is located in the posterior parietal cortex, with the peak activity in the inferior parietal lobule.

These findings illustrate the applicability of a new analysis method for fMRI, which not only allows for using complex tasks in the fMRI scanner, but also for locating multiple cognitive resources in one experiment. For example, while we have shown the results for the problem state resource, the same methodology can be used for the visual resource, yielding an area in the occipital cortex. Furthermore, this model-based fMRI analysis method can be applied to every data set when there is a model available that is more detailed than the global trial structure of the experiment, showing which constructs of a model are linked to which brain areas.

Acknowledgments

The research was supported by Office of Naval Research grant N00014-08-10541 awarded to Niels A. Taatgen.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Borst, J. P., Taatgen, N. A., & Van Rijn, H. (2010). The problem state: A cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 36*(2), 363-382.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review, 115*(1), 101-130.

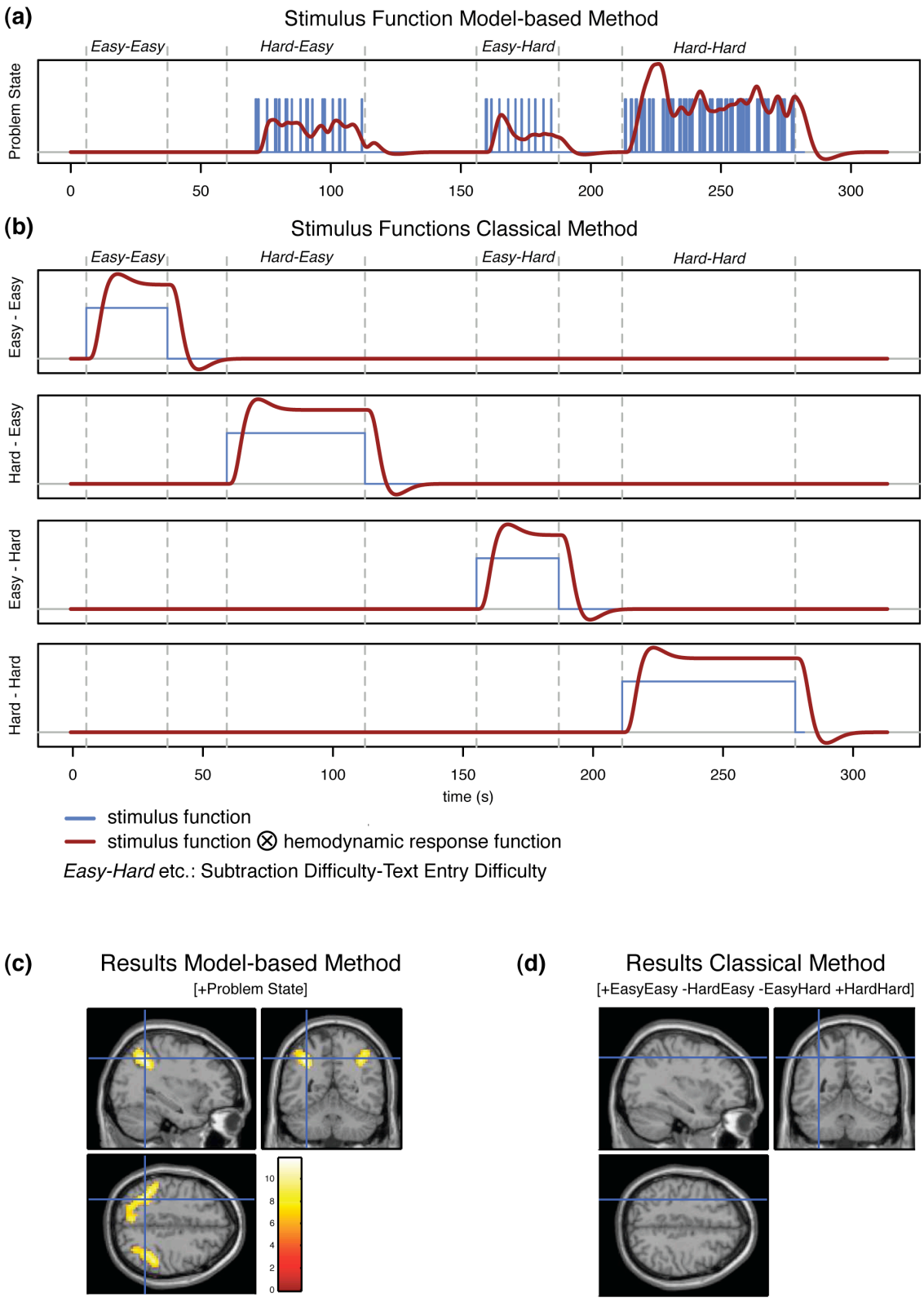


Figure 1. Comparison of the model-based analysis method and the classical method. Panel (a) shows the demand function of the problem state resource in the model in blue, and its convolution with a hemodynamic response function in red. Panel (b) shows the four stimulus functions that are necessary to analyze an interaction using the classical method. *Easy-Easy* etc. above the diagrams indicate the experimental condition. Panel (c) shows the results of the model-based method (contrast is displayed above the results), and (d) the results of the classical method.

"Hello Java!" Linking ACT-R 6 with a Java simulation

Philippe Büttner
Technische Universität Berlin
Centre of Human Machine Systems
Franklinstr. 28-29, FR 2-6
10587 Berlin, Germany
pbu@zmms.tu-berlin.de

Abstract

Many more applications and simulations are developed in the Java programming language than in the Lisp programming language. This can be attributed to a number of reasons, including platform independence, object-oriented programming, etc. Due to the fact that the ACT-R software was programmed in Lisp, incompatibility issues between it and Java arose. These issues necessitated the establishment of a tool capable of preventing a Lisp reimplementation of existing Java applications. Consequently, "Hello Java!" was developed to link cognitive models written in ACT-R with Java applications, namely simulations. In order to achieve this, a package must be added to the Java simulation so that it can observe and perform actions on the frame, as well as communicate with the ACT-R software. The line of communication between Java and Lisp is established through a TCP/IP connection. As a result, the simulation and cognitive models can be run on different computers. Since the release of ACT-R 6, the methods for perception and action have been externalized. These externalized methods can be utilized as devices for the ACT-R software, making it possible to, consequently, use a Java simulation as a device for ACT-R.

Keywords: ACT-R; device; Java; simulation; network

Introduction

ACT-R (Anderson & Lebiere, 1998) is a computational theory of human cognition with two separate, but interacting, knowledge stores developed in Lisp. Both declarative knowledge and procedural knowledge are unified into a production system where procedural rules act on declarative chunks. The ACT-R system includes the capability to create simulated environments, such as screen interfaces. Production rules have the ability to interact with this environment by perceiving objects and making motor movements through perceptual and motor buffers.

The externalization of all necessary methods involved in the perception of objects and the facilitation of motor movements makes it possible to extend the environment to a world outside of ACT-R, without requiring a modification of its architecture. An interaction with the production rules can be enabled with any device which meets the specifications of these externalized methods. Due to the fact that Lisp supports communication via the TCP/IP network protocol, it becomes possible for ACT-R to interact with any environment also possessing the capability to communicate via TCP/IP.

I was able to take advantage of this trait and developed "Hello Java!" as an open source tool for linking cognitive models written in ACT-R to applications and simulations written in Java. This tool and additional examples, including the source code, are available on the following website: http://www.zmms.tu-berlin.de/kogmod/tools/hello-java.html

Design

To link a cognitive model written in ACT-R to a frame-based Java simulation two elements that coordinate the interaction between the model and the simulation are required. The first of these elements is a Java package that must be added to the simulation. The second element is a device for the ACT-R software that bridges the incompatibility between Java and Lisp with the TCP/IP network protocol. I will now proceed to describe the communication and synchronization of the elements in detail:

Communication

The line of communication between Java and Lisp is established through a TCP/IP connection, which is a protocol commonly used to connect computers to the internet. All information sent with this protocol can be transmitted as a string representation. This string representation carries data pertaining to perception and action that is exchanged between the cognitive model and the simulation. In order to be able to communicate via TCP/IP, each side must encode and decode information using a common vocabulary and unified grammar. ACT-R receives data pertaining to perception and sends back data pertaining to the execution of motor movements. Java, meanwhile, receives data instructing it to perform the cognitive model's desired action and provides a visual representation of all objects visible in the frame.

By using a network structure it is possible to run a cognitive model and a simulation on different computers. This increases computing power for each, cognitive model and simulation.

Synchronization

In order to keep the cognitive model and the simulation synchronized, the simulation must adjust its cycle speed to that of the cognitive model. The opportunity to do this applies to all clock-controlled simulations. A clock-controlled simulation updates all elements visible on the screen after one cycle. The length of a cycle depends on the time resolution of the simulation. The shorter the cycle, the smoother a simulation appears to be.

ACT-R is responsible for adjusting the cycle speed of the simulation. It accomplishes this by synchronizing the execution of the simulation's cycles with the computed time interval of the cognitive model. ACT-R's scheduler is used to trigger the cycles in a calculated time interval.

As an example, let us assume that the cycle length of a clock-controlled simulation is one second. The scheduler will also be adjusted to one second and it assumes control of the cycles of the simulation. Due to the fact that the time interval of the scheduler is synchronized with the computed time interval of the cognitive model, the same time will elapse for the simulation and the cognitive model.

Extending the Java simulation

Java has established itself as a widely used, universal, platform-independent programming language. In order for ACT-R to be able to access a Java simulation, the simulation itself must be extended by a package. In general, a Java package contains classes, methods and functions that extend an application. This package coordinates the functions of observing all objects from the simulation, performing actions on the simulation, and exchanging information with ACT-R. In order to run the package, one must first add it to the simulation and initialize it. The package consists of three sub-packages. Their descriptions and roles are listed below:

Robot

This sub-package triggers actions like clicking a mouse button, moving the mouse, stroking a key, moving the attention pointer and speaking, if the cognitive model decides to perform one of these actions.

GUI

GUI contains the information of all objects visible in the frame. By recursively accessing objects in the frame, data pertaining to the following objects becomes accessible: labels, text fields, buttons, radio buttons and toggle buttons. In principle, every Java object can be accessed. Therefore, information pertaining to the kind, value, colour, size and relative position of an object must be encoded to a string representation. This information is necessary for the visual icon of ACT-R. An object can be one of the following types: text, line or oval. Because ACT-R interprets an oval as a button, every type of button is assigned as oval.

Net

This sub-package provides all methods responsible for encoding and decoding information and handles the network connection with ACT-R. A socket process is started that waits for a connection from ACT-R on a predefined port. If the socket process receives information, it proceeds to parse it, thereby enabling it to perform actions based on methods obtained from the robot sub-package. Furthermore, it can send back visual information from the GUI package.

ACT-R device

As described above, ACT-R 6 provides externalized, accessible methods responsible for the perception of objects and the execution of motor movements. These methods can be implemented and utilized as a device, resulting in an interaction between this particular device and the production rules. In order to be able to link ACT-R to a Java simulation, it was necessary to implement the following methods:

- device-move-cursor-to: ACT-R sends an action to the Java simulation to move the mouse pointer to a given position.
- device-handle-click: ACT-R sends an action to the Java simulation to perform a mouse click.
- device-handle-keypress: ACT-R sends an action to the Java simulation to perform a keystroke.
- device-speak-string: ACT-R sends an action to the Java simulation to speak a string.
- get-mouse-coordinates: ACT-R sends a request to the Java simulation to gather data pertaining to the position of the mouse.
- build-vis-locs-for: This method updates the visual icon of ACT-R's vision module with all visible objects of the device. It will be invoked after the proc-display command is called.
- device-update: This method is called after ACT-R computes one cycle. At this point it is optional to update the visual icon of ACT-R's vision module or to perform other tasks.

Updating the visual icon of the vision module

To update the visual icon, it is necessary to call an update-method. This method regulates the gathering of visual information, but is not one of ACT-R's device-methods. The update-method sends a request to the Java application instructing it to collect data pertaining to all visible objects present in the simulation. Once this visual data is transmitted to ACT-R, it is then written into the visual icon of ACT-R via the "build-vis-locs-for" method. The visual icon provides information about the kind, value, colour, size and relative position of visible objects in the environment. This method responsible for updating the visual icon can be triggered in two different ways:

- By utilizing a scheduler corresponding to a regular time interval, resulting in reduced network traffic for longer time intervals. A scheduled update is especially practical on clock-controlled simulations in which the simulation updates the screen after a certain interval.
- By allowing every instance in which ACT-R calls the "device-update" method to serve as a trigger. Although this method ensures that the visual icon is always up-to-date when being accessed by ACT-R, it incurs higher network traffic costs.

Discussion

"Hello Java!" was developed as a tool to directly interface ACT-R with an external system. Its defining trait is its ease of use resulting from the fact that no modification of the simulation is required, that no restrictions are imposed upon the model and that it is possible to synchronize the simulation with the model.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: Erlbaum.

Answer Set Programming for Computational Psychological Models

Sara Girotto (sara.girotto@ttu.edu)
Department of Psychology, Texas Tech University
Lubbock, TX 79404

Marcello Balduccini (marcello.balduccini@gmail.com)
Intelligent Systems, KRL, Eastman Kodak Company
Rochester, NY 14650

Keywords: formalization of psychological knowledge; answer set programming; short term memory.
Abstract: Our work explores the use of Answer Set Programming (ASP) to formalize and reason about psychological knowledge. To demonstrate the viability of ASP for this task, in this paper we discuss an ASP-based formalization of the mechanisms of Short Term Memory.

Introduction

Our work explores the use of Answer Set Programming (ASP) (Gelfond & Lifschitz, 1991; Marek & Truszczynski, 1999) to formalize and reason about psychological knowledge. Whereas some psychological models have a clear quantitative nature, which allows modeling e.g. with neural networks or Bayesian networks, other models have a more logical or qualitative nature and are not suitable for formalization with these techniques. ASP is a knowledge representation formalism allowing for concise and simple representations of defaults, uncertainty, common-sense and evolving domains, and has been demonstrated to be a useful paradigm for the formalization of knowledge of various kinds (e.g., Baral & Gelfond, 2005; Son & Sakama, 2009). For this reason, we believe that ASP can be used successfully for the formalization of psychological knowledge that is of qualitative nature. ASP is also directly executable, in the sense that the consequences of collections of ASP statements can be directly computed using computer programs. Hence, ASP-based formalizations of psychological knowledge can be viewed as computational models of the underlying psychological theories.

Answer Set Programming

In ASP, terms and atoms are formed according to the standard rules of first-order logic. A literal is either an atom a or its strong (also called classical or epistemic) negation $\neg a$. In its simplest form, a *rule* is a statement:

$$h \leftarrow l_1, l_2, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where h and l_i's are literals and *not* is the so-called *default negation*. The intuitive meaning of the rule is that a reasoner who believes $\{l_1, \ldots, l_m\}$ and has no reason to believe $\{l_{m+1}, \ldots, l_n\}$, must believe h. The availability of two types of negation is one important feature of ASP, allowing for great flexibility in knowledge representation. In particular, the way default negation is treated in ASP allows to easily encode defaults (such as "an action is allowed unless it is explicitly stated that it is not") and also to represent uncertainty and alternative, different views of the world (e.g. "either symbol *a* or symbol *b*, but not both, will be forgotten, but we do not know which one). The precise definition of the meaning of sets of ASP rules (called an *ASP program*) is given by the *answer set semantics* (Gelfond & Lifschitz, 1991), which characterizes a suitable notion of logical consequence. We omit further details due to space constraints; rather, in the rest of the paper we rely on the informal meaning of rules given above. From a practical perspective, the logical consequences of sets of ASP rules can be computed automatically, and rather efficiently, using computer programs called *ASP solvers*. These solvers can of course be also interfaced to pre-processors, post-processors and user interfaces, to build sophisticated end-to-end systems (e.g., Balduccini, Gelfond & Nogueira, 2006).

ASP-Based Formalization of STM

To demonstrate that ASP is suitable for and successful at formalizing psychological knowledge, we have developed an ASP-based formalization of the mechanisms of operation of Short-Term Memory (STM), as described by Atkinson & Shiffrin (1971). This theory was selected because it reflects the type of psychological knowledge that we aim at formalizing: it is mostly of qualitative nature and is expressed in the literature at a rather high level of abstraction. Moreover, its formalization is challenging because it involves modeling of a sophisticated dynamic domain, involving non-determinism, fixed-capacity storage, and decay over time. In order to show that the use of ASP is not limited to a single theory, we have formalized not only the traditional theory of STM (Atkinson & Shiffrin, 1971), but also an alternative STM model (e.g., Card, Moran & Newell, 1983) in which decay is influenced not only by elapsed time but also by other variables such as the number of chunks a user is trying to remember and retrieval interference with similar chunks activated in working memory. It is worth stressing that the accounts of STM (Atkinson & Shiffrin, 1971; Card, Moran & Newell, 1983) that have been formalized by means of ASP are relatively well-established models from the existing literature. Our purpose in this study is not to modify the models but to show that they can indeed be formalized using ASP.

Using a common methodology in ASP-based knowledge representation, the formalization process starts by

condensing the description of STM in a number of *precisely formulated* statements in natural language. For example, the set of statements for the traditional theory of STM contains 16 items, including the following: (1) STM is a collection of symbols; (2) the size of STM is limited to ω elements (Cowan, 2000); (3) each symbol has an expiration time; (4) symbols can be added to STM; (5) if a new symbol is added to STM when ω elements are already in it, the symbol that is closest to expiring is removed from STM (i.e. forgotten).

Then, the logical representation of the formalization is created by choosing suitable relations and functions, and using them to encode the natural language statements and the underlying knowledge. Because we are interested in describing how the contents of STM change over time, we use two special relations, $holds(f, i)$, saying that property f holds at step i in the evolution of the contents of STM, and $occurs(a, i)$, saying that action a occurs at step i. For instance, statement (4) is encoded by a rule:

$$holds(in\_stm(S), I+1) \leftarrow occurs(store(S), I)$$

whose informal reading is: if the action of storing some symbol S (by convention an uppercase initial denotes a variable) occurs at some step I, then S will be in STM at the next step $I+1$. The rule is based on the stipulation that $store(S)$ represents the action of adding a symbol to STM, and that $in\_stm(S)$ encodes the fact that S is in STM.

Statement (5) is formalized by the following rule, as well as the definition (omitted to save space) of the auxiliary relations used in it:

$$\begin{aligned}\neg holds(in\_stm(S'), I+1) \leftarrow\ & S \neq S', occurs(store(S), I),\\ & stm\_max\_size(MX),\\ & curr\_stm\_size(MX, I)\\ & \text{not } some\_symbol\_expiring(I),\\ & oldest\_in\_stm(S', I)\end{aligned}$$

The informal reading of the rule is: when *S* is stored in STM, if the current size of STM equals its maximum size (represented by relations $curr\_stm\_size(MX, I)$ and $stm\_max\_size(MX)$, respectively) and no symbol is due to expire (written as $\text{not } some\_symbol\_expiring(I)$), then the symbol which is closest to expiring is removed from STM (encoded by means of the classical negation of $holds(in\_stm(S'), I+1)$, saying that S' will not be in STM at the next step).

Using terminology from the literature on ASP, the set of all rules formalizing STM is called *action description*, and denoted by Π_{STM}. To use the action description in order to predict the behavior of STM in a particular situation, one writes additional rules, say Π_{sit}, describing the situation, and then uses an ASP solver to compute the logical consequences of the ASP program consisting of Π_{STM} and Π_{sit}. For example, given the encoding of a memory-span test in which the subject is required to remember the sequence 1-7-3-2-6-5 and the maximum size of STM is of 4 items, the output of the ASP solver for $\Pi_{STM} \cup \Pi_{sit}$ contains statements such as $holds(in\_stm(digit\_1), 1)$ and $\neg holds(in\_stm(digit\_1), 5)$, showing that digit 1 was correctly stored initially, but forgotten at step 5 (because digit 6 was stored when STM was already at its full capacity).

In our study we also demonstrate the computational aspects of our modeling technique by creating an application of our formalization to the task of predicting a user's performance in the interaction with a graphical user interface. We developed an ASP-based representation of a scenario in which a user is told a sequence of tasks and is expected to execute it relying only on memory of the sequence. The prediction of the corresponding ASP program is in line with what the STM model (Atkinson & Shiffrin, 1971) predicts**,** correctly determining (1) if the user will or will not be able to remember and execute the sequence, and, in case the user forgets part of the sequence, (2) which pieces of information are (likely to be) forgotten and when. Execution of the ASP program is also rather fast, with most predictions computed in less than a second.

Conclusions

The ASP-based formalization appears promising in terms of producing accurate predictions of performance from mostly qualitative models of behavior. It also allows analysis and comparison of different psychological theories, as well as prediction of the outcome of experiments, thus making it possible to design better experiments and diminishing the need for prototyping and user testing. This success opens the door to the use of ASP for the formalization of other psychological knowledge and models, as well as for its practical use in HCI-oriented applications.

References

Atkinson, R. C., & Shiffrin, R. M. (1971). The control of short-term memory. *Scientific American, 225*, 82-90.

Balduccini, M., Gelfond, M., & Nogueira, M. (2006). Answer set based design of knowledge systems. *Annals of Mathematics and Artificial Intelligence, 47(1-2)*, 183-219.

Baral, C., & Gelfond, M. (2005). Reasoning about intended actions. *Proceedings of the 20th International Conference on Artificial Intelligence* (pp. 689-694). AAAI Press.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Mahwah, NJ: Lawrence Erlbaum Associates.

Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24*, 87-185.

Gelfond, M., & Lifschitz, V. (1991). Classical negotiation in logic programs and disjunctive databases. *New Generation Computing, 9*, 365-385.

Marek, V. W., & Truszczynski, M. (1999). Stable models and an alternative logic programming paradigm. In Apt, K., Marek, V., Truszczynski, M., Warren, D. (Eds), *The logic programming paradigm: A 25-year perspective*. Berlin, Germany: Springer Verlag.

Son, T. C. & Sakama, C. (2009). Negotiation using logic programming with consistency restoring rules. *21st International Joint Conferences on Artificial Intelligence (IJCAI).* Morgan Kaufmann Publishers Inc.

Towards a cognitive model of conceptual blending

Markus Guhe, Alan Smaill, Alison Pease ([m.guhe, a.smaill, a.pease]@ed.ac.uk)

School of Informatics, University of Edinburgh, 10 Crichton Street
Edinburgh EH8 9AB, Scotland

Abstract

We outline a way to use Goguen's (2006) account of conceptual blending in the cognitive architecture ACT-R. Despite recent advances in linguistics and general accounts of conceptual blending (for example, Fauconnier and Turner 2002, 2008) it has received scant attention in cognitive modelling, which is partly due to the fact that there are hardly any computational accounts of this phenomenon, Goguen's being one of them.

Keywords: conceptual blending; metaphor; analogy; linguistics; conceptualisation; scientific creativity; ACT-R; Theory of Institutions.

Analogy, metaphor, conceptual blending

A major factor for the power and flexibility of the human cognitive system is its ability to create new concepts, in particular by combining existing ones. It is both central in creating new scientific ideas as well as for 'everyday' thinking. We are particularly interested in the role of this mental machinery in the creation of new mathematical concepts (Guhe, Smaill and Pease 2009). Most current accounts of scientific creativity emphasise the role of analogy (Gentner & Markman, 1997) or metaphor (Lakoff & Núñez, 2000). Here, we outline the more general process of conceptual blending, its role in creating new concepts, and how it can be integrated into the cognitive architecture ACT-R (Anderson, 2007).

Analogy and metaphor, which we take to be essentially the same, are cognitive processes that (1) establish mappings between parts of a cognitive system's knowledge representations (usually called *domains*) and that (2) can transfer knowledge between domains for which a mapping was established. For example, in the extensively studied metaphor TIME IS SPACE, the expression *Christmas is two days away* recasts an event (*Christmas*) as a location with respect to the speaker's current location in time by specifying a temporal interval (*two days*) as a distance.

According to Fauconnier and Turner (2002) metaphors and analogies are only special cases of conceptual blending. A metaphor is simply a 'cross space mapping' (Goguen, 2006, p. 8). The TIME IS SPACE metaphor, for example, not only provides the basic mapping, but allows reconceptualisations as well as the integration of knowledge from other domains. A common reconceptualisation of the TIME IS SPACE conceptual blend is, for example, a change in perspective, where time is conceptualised as passing a static observer, e.g. in the expression *Time passes slowly* (Fauconnier and Turner 2008). It is important to note that a metaphorical or analogical mapping alone cannot account for this additional mental flexibility.

Goguen's approach

Fauconnier and Turner's account of concept blending is not directly suited for computational cognitive modelling, because it remains purely descriptive. Goguen (2006) outlines a computational account of conceptual blending – based on Fauconnier and Turner – using the theory of *Institutions*, a theory similar to *Information Flow*, which we used earlier (Guhe, Smaill, & Pease, 2009).

We cannot go into much detail here, so we will restrict ourselves to one of Goguen's (2006) motivating examples of a conceptual blend between the concepts HOUSE and BOAT, resulting in the conceptual blends HOUSEBOAT and BOATHOUSE, cf. Figure 1 for a depiction of the HOUSEBOAT blend. A base domain (shown at the bottom) provides the 'glue' needed for mapping two domains (in the middle, left and right) and creating a conceptual blend (at the top). The most important mapping here is the one of *live-in* and *ride*, which provides the reconceptualisation of a BOAT as an OBJECT in which a person can not only RIDE but also LIVE.

Goguen restricts the many possible conceptual blends by specifying sortal frames, which must match in order for a mapping between domains to succeed. Sortal restrictions are

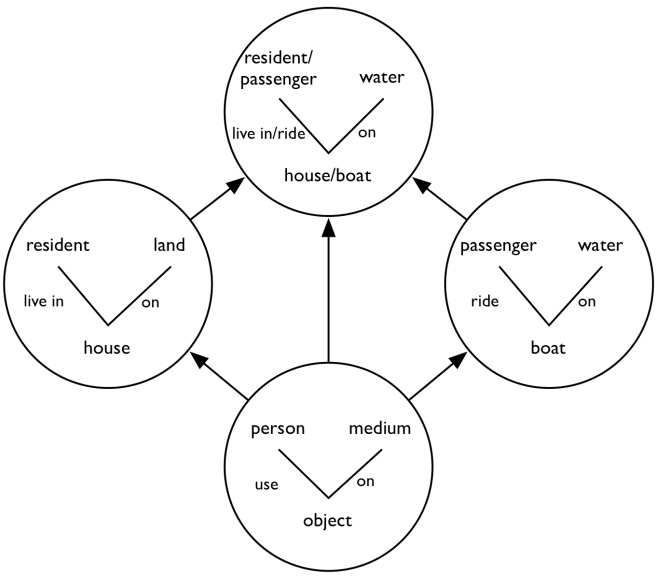


Figure 1: HOUSEBOAT conceptual blend

specified in a signature, for example for the HOUSEBOAT case Goguen defines the following sortal frames:

resident: → *Person*	*passenger :* → *Person*
house : → *Object*	*boat :* → *Object*
land, water: → *Medium*	*land, water:* →*Medium*
livein : Person Object → *Bool*	*ride : Person Object* → *Bool*
on: Object Medium → *Bool*	*on: Object Medium* → *Bool*
livein(resident, house)	*ride(passenger, boat)*
on(house, land)	*on(boat, water)*

Transfer to ACT-R

The translation of Goguen's proposal to ACT-R (Anderson, 2007) is rather straightforward. In our prototypical implementation, facts are represented as chunks and the matching and transfer operations are realised with production rules. The one major problem is that ACT-R does not have a sortal mechanism comparable to Goguen's. Although ACT-R uses sorts (in the form of chunk types), it does not automatically check for super-/subsort relations like in Goguen's conception. This means, for example, that WATER is not automatically understood to match frames specifying MEDIUM. Thus, the mapping of *on(house, land)* to *on(boat, water)* fails, because these facts cannot be linked via the base domain (by using *on(object, medium)*). We outline two basic solutions below. Which one provides a better model of the cognitive mechanisms will have to be established experimentally.

Solution 1 – Explicit sortal checks

The first solution is to explicitly perform sortal checks with a set of production rules. For such a model we coded information about subsorts as chunks of type

```
(chunk-type is-subsort sort1 sort2)
```

The production rules performing the sortal checks keep the information about the two facts that are being compared in the imaginal buffer while the information about the sortal hierarchy is retrieved from the declarative memory.

A variant for faster processing is to include sortal information with the facts, e.g. for predicates:

```
(chunk-type predicate name result-sort
            par1 sort1 par2 sort2)
```

The major disadvantage of this solution is that the representations contain much redundancy and do not provide the usual generalisations, e.g. that WATER is a subsort of MEDIUM.

Solution 2 – Amending the declarative module

An alternative solution is to change ACT-R on the architectural level, i.e. to amend the declarative module. A rather mild extension is to provide the declarative module with sortal information (e.g. a lattice of sorts) and let it consider not only chunks that directly match the sort of the chunk (i.e., that match in the `isa` slot) but also chunks that have a supersort of the chunk being requested.

A more severe alteration is to check all slot values that a chunk specifies and match not only the values themselves but check for values higher up in the sortal hierarchy. For example, if a request to the declarative module specifies a chunk with a slot–value pair like

```
retrieval>
   isa predicate
   name on
   par1 house …
```

the `par1` slot would also match for chunks like:

```
retrieval>
  isa predicate
  name on
  par1 object …
```

Solution 2 predicts much faster processing than solution 1, because all checks are performed within one memory retrieval. Thus, it neither requires firing multiple productions nor multiple retrievals from declarative memory.

Conclusion

Conceptual blending is a central, powerful and productive aspect of human cognition, allowing, for example, to conceptualise time in terms of space. However, cognitive modelling has not yet seriously addressed this issue. We outlined in broad terms a way to transfer Goguen's notion of conceptual blending into the cognitive architecture ACT-R as a first step to include conceptual blending in cognitive models of scientific creativity, in particular mathematical thinking. Whether a modification of ACT-R's declarative module will provide better cognitive adequacy will have to be decided on the basis of empirical data.

Acknowledgements

The research reported here was carried out in the *Wheelbarrow* project, funded by the EPSRC grant EP/F035594/1.

Bibliography

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford Univ. Press.

Fauconnier, G., & Turner, M. (2008). Rethinking metaphor. In *Cambridge Handbook of Metaphor and Thought* (pp. 53–66). New York: Cambridge Univeristy Press.

Fauconnier, G., & Turner, M. (2002). *The way we think: Conceptual blending and the mind's hidden complexities*. New York: Basic Books.

Gentner, D., & Markman, A. (1997). Structure Mapping in Analogy and Similarity. *Am. Psychologist, 52* (1), 45–56.

Goguen, J. (2006). Mathematical Models of Cognitive Space and Time. In D. Andler, Y. Ogawa, M. Okada, & S. Watanabe, *Reasoning and Cognition*.

Guhe, M., Smaill, A., & Pease, A. (2009). Using Information Flow for Modelling Mathematical Metaphors. In *Proc. of the 9th ICCM*.

Lakoff, G., & Núñez, R. E. (2000). *Where Mathematics Comes From*. New York: Basic Books.

Modeling Interaction and Integration of Perception and Action

Pascal Haazebroek (phaazebroek@fsw.leidenuniv.nl)
Bernhard Hommel (hommel@fsw.leidenuniv.nl)
Cognitive Psychology Unit & Leiden Institute for Brain and Cognition, Wassenaarseweg 52
Leiden, 2333 AK The Netherlands

Keywords: Perception, Action, Interaction, Integration, Binding, Modeling, Cognitive Architectures, Simulation

Introduction

To account for perceptual and action-related stages of information processing, most prominent cognitive architectures have extended their coverage from primarily cognitive processes to perceptual processing and response execution (e.g. EPIC (Kieras & Meyer, 1997); ACT-R/PM (Byrne & Anderson, 1998)). However, despite these extensions they are typically still too limited to explain some well-known phenomena from the perception-action domain in cognitive psychology such as stimulus-response compatibility effects (e.g., Simon Effect (Simon & Rudell, 1967), and action preparation influences on perception (e.g., Müsseler & Hommel, 1997). These phenomena suggest that perception and action are more intimately related than these architectures allow for. We are currently developing HiTEC (Haazebroek, Raffone & Hommel, submitted), a novel cognitive architecture that stresses both the interaction and integration between perception and action.

Here we describe the overall structure and general principles of HiTEC and we demonstrate how a variety of psychological phenomena in the perception-action domain can be replicated using computer simulations of the model.

HiTEC

As shown in Figure 1, HiTEC consists of three levels: the sensorimotor level, the common coding level and the task level. At the sensorimotor level, stimuli are encoded by activating sensory codes. Motor actions are executed by activating motor codes. Sensory codes and motor codes are both connected to feature codes at the common coding level. These feature codes represent a-modal perceptual features (e.g., location, intensity et cetera). Crucially, both stimulus features (e.g., location of a tone) and response features (e.g., location of a key press) are encoded using these common codes, thereby allowing for two-way interaction between stimuli and responses (Hommel et al., 2001).

Feature codes are connected to task codes at the task level. These connections reflect the task instruction allowing the model to respond according to specified stimulus-response (S-R) mappings. By dynamically setting up these connections, different S-R mappings can be implemented, allowing the model to simulate a variety of tasks, while keeping all other codes and connections unchanged.

Within the levels, codes are arranged in maps. Sensory codes are contained in sensory maps, corresponding to sensory dimensions (e.g., color), feature codes are contained in feature maps reflecting more cognitive feature dimensions (e.g., global location). At the task level there is one map containing task codes representing the different response alternatives within the current task. There is also one motor map containing motor codes representing a limited number of specific movements.

In the HiTEC architecture, stimulus processing and response selection are one and the same process: stimuli activate certain sensory codes, activation flows through the model and at all levels interaction takes place, letting the model converge to a condition with only one motor code having an activation value above a set threshold. This results in the selection of that motor action and execution of the corresponding response.

In addition to this propagation of activation there are integration processes at work that temporarily bind feature codes into event files (Hommel, 2004). These event files - illustrated by the gray feature codes in Figure 1 - modulate the overall dynamics of the model, by selectively enhancing feature codes belonging to one event file and at the same time making them less available to other processes.

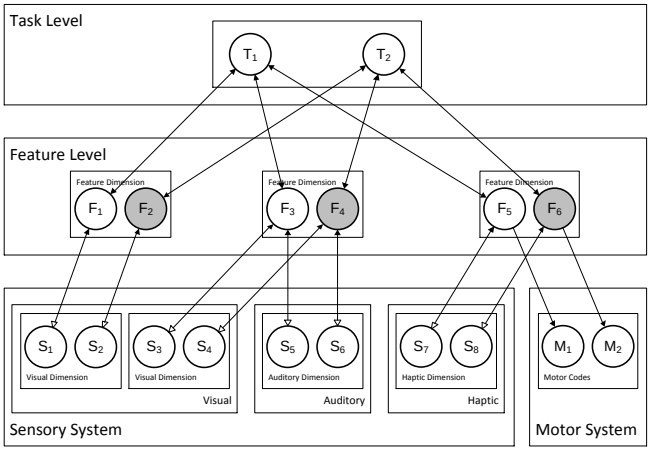


Figure 1: General structure of HiTEC architecture. Circles denote codes, lines denote connections, rectangles are maps, and gray feature codes belong to the same event file.

The above mentioned principles that govern the dynamics in HiTEC are strongly based on both theoretical and empirical work in cognitive psychology. By integrating them into a cognitive architecture that can be used to simulate a variety of well-known phenomena we aim at obtaining a richer understanding of the intricate interplay

between perception and action than theory alone can provide.

Other cognitive architectures such as EPIC and ACT-R/PM do address perceptual and motor related aspects of human performance. However, these architectures differ on several crucial aspects. Where HiTEC treats perceptual processing as part of the overall 'translation process' and therefor allows perception to be modulated by task preparation and even action planning, EPIC and ACT-R/PM treat this perceptual stage as 'additional waiting time' before the cognitive core system (using production rules) can start to work. In similar vein HiTEC treats action planning as part of the overall 'translation process', susceptible to influences from perception and task set.

Thus, where other architectures focus on the cognitive middle 'stage' between perception and action, HiTEC puts perception and action – and their interplay – at the center treating cognition mainly as a modulatory influence. By assuming common codes for both perception and action interactions can occur that are impossible when segregating perceptual, cognitive and motor stages as is common in other architectures. These interactions allow the replication of empirical phenomena related to stimulus-response crosstalk (both enhancement and impairment).

HiTEC is not yet as mature as other cognitive architectures and cannot be readily used to model the diversity of tasks that other architectures have been shown to successfully replicate. Yet, by taking perception-action as primary perspective we provide a line of research that complements existing approaches in cognitive architectures.

Acknowledgments

Support for this research by the European Commission (PACO-PLUS, IST-FP6-IP-027657) is gratefully acknowledged.

References

Byrne, M.D., & Anderson, J.R. (1998). Perception and action. In: Anderson, J.R., Lebiere, C. (eds.) *The atomic components of thought*. Erlbaum, Hillsdale.

Byrne, M.D. (2008). Cognitive architecture. In: Jacko, J.A., Sears, A. (eds.) *Human-Computer Interaction Handbook*. Erlbaum, Mahwah.

Haazebroek, P., Raffone, A., & Hommel, B. (submitted). HiTEC: A computational model of the interaction between perception and action.

Hommel, B. (2004). Event files: Feature binding in and across perception and action. *Trends in Cognitive Sciences, 8*, 494–500.

Hommel, B., Muesseler, J., Aschersleben, G., & Prinz, W. (2001). The theory of event coding (TEC): A framework for perception and action planning. *Behavioral and Brain Sciences, 24*, 849–937.

Kieras, D.E., & Meyer, D.E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*, 391–438.

Müsseler, J., & Hommel, B. (1997). Blindness to Response-Compatible Stimuli. Journal of Experimental Psychology: Human Perception and Performance, 23, 861-872.

Simon, J., & Rudell, A. (1967). Auditory s-r compatibility: The effect of an irrelevant cue on information processing. *Journal of Applied Psychology, 51*, 300–304.

LETF

A Lisp-Based Exploratory Testing Framework for Computational Cognitive Models

Clayton T. Stanley (clayton.stanley@wpafb.af.mil)
Air Force Research Laboratory
WPAFB, OH 45431 USA

Keywords: exploratory testing, ACT-R, high performance computing

Introduction

Developing a computational cognitive model is an iterative process. Due to this, any model validation/testing technique cannot be static, and must be agile enough to evolve alongside the model's development. Over the past few decades, a software validation paradigm called exploratory testing has emerged within the software engineering community. Exploratory testing is not static, favors a high level of interactivity between the tester and the program, and advocates that programmers should "build time and enthusiasm for parallel research, test development, and test execution" (Kaner, 2004). As cognitive modelers, trying to develop a model that performs within a certain range of objective performance metrics while maintaining a level of cognitive plausibility, exploratory testing is not new to us. In fact, testing frameworks are already available to the cognitive modeling community. In ACT-R, for example, there is the visual-location crosshair for the vision module, the buffer activity trace, and the fMRI BOLD visualization.

However, many of these exploratory testing formalisms are specific to the cognitive architecture that the modeler is using, and there are a fair amount of exploratory tests that are architecturally agnostic. For example, exploring the architectural/strategy/parameter space of a cognitive model, computing objective performance measures, and capturing the central tendency of stochastic models are all common modeling issues. In order to enable modelers to easily explore these sorts of generic performance metrics, I have developed LETF, a lisp-based exploratory testing framework for computational cognitive modelers.

LETF

LETF is a lightweight configurable lisp program that layers on top of a cognitive model. After LETF is configured and launched, it spawns the cognitive model as a separate process, sending inputs to the model as command line arguments, and grabbing outputs from the model by capturing the standard output stream. Model inputs are specified in a work file, where each row in the file is a particular parameter combination, and columns correspond to the values for a parameter in the model. A flexible configuration file allows for extended processing of model outputs, and configurable display format by means of a modular print method. The flexibility of the configuration file is an important consideration, because once the model is set up to interface with LETF, modifying and adding model tests does not require altering the model's code. Instead, you express the tests by modifying the configuration file.

In order to express a large range of different model tests in the configuration file, we are not providing a large number of specific APIs to support each test. Instead, we have removed the API layer altogether, and grounded the syntax of the configuration file to the underlying language of the generic testing framework. That is, the syntax of the configuration file *is* lisp. And it is this critical feature that makes adding and changing model tests in LETF so expressive and agile.

The best way to make this point is with a simple example. Suppose we have a model written in lisp (e.g., an ACT-R model), with independent variables (IVs) 'noise', 'speed', dependent variables (DVs) 'rt1', 'rt2', 'rt3', and we want to compute the correlation between the observed and model data. The configuration file to express this model test (i.e., compute the correlation) could look like Figure 1.

```
IV= noise
IV= speed
file2load= extras.lisp
modelRTs= (list [rt1] [rt2] [rt3])
observedRTs= (getObservedRTs)
correlRTs= (correl [modelRTs] [observedRTs])
DV= correlRTs
```

Figure 1: Example configuration file.

The model code that would communicate with LETF for this example could look like Figure 2.

```
(defun run-model (&key (noise) (speed))
   …run the model using the values for IVs 'noise' and
   'speed' that were passed to 'run-model'
   (format t "rt1=1.1~%")
   (format t "rt2=2.2~%")
   (format t "rt3=3.3~%"))
```

Figure 2: Example model.

Note that LETF offers a more direct interface when the model is in lisp (shown in Figure 2). In this case, LETF can communicate with the model by calling the entry function 'run-model' instead of spawning a separate process.

Many cognitive models are stochastic, and so it is common to run them multiple times to reveal the central tendency. This can be accomplished with the simple addition to the configuration file

```
collapseQuota= 100                                    (1)
```

And, if we wanted to use a collapsing function other than the default 'mean', we could specify our own directly in the configuration file (note the lisp syntax)

```
collapseFn= (lambda (a) (+ (mean a) 1000))            (2)
```

LETF calculates the value of a variable X on a 'DV=X' line by expanding the string expression that variable X represents, and then invoking the lisp reader to interpret that expression and return a value. Using the example in Figure 1, the 'DV=correlRTs' line tells the program to find the expanded string expression for the variable 'correlRTs'. LETF finds the 'correlRTs=' line in the configuration file, and sets the value of 'correlRTs' to

```
(correl [modelRTs] [observedRTs])                     (3)
```

Then, all variable names within brackets '[]' are recursively expanded to their values, and the program evaluates the expression by invoking the lisp reader

```
(eval (read-from-string
 "(correl  (list 1.1 2.2 3.3) (getObservedRTs))")     (4)
```

Note that once the brackets are recursively expanded, the API between the configuration file and LETF *is* lisp, matching the underlying language of the generic testing framework. Going back to Figure 1, the modeler is writing a lisp expression *around* the variables sent by the model (rt1-3) in order to calculate a DV of interest ('correlRTs'). The function 'getObservedRTs' (which returns a list of observed response times for trials 1 through 3) would be defined in the lisp file 'extras.lisp', which is loaded (by specifying the 'file2load=extras.lisp' line) and therefore visible to LETF when it evaluates the string expression in [4]. The 'correl' function has already been defined in LETF, which computes the correlation between two lists, so the expression in [4] will bind the correlation of RTs between the model and observed data to the variable 'correlRTs'. Having this on-the-fly configurability available directly in the configuration file – both syntactically and semantically anchored in the experimental testing framework's own language – can be a very powerful paradigm.

Discussion

LETF supports the exploratory testing that a modeler might perform during the early stages of model development. For example, with modest computational resources (e.g., a laptop workstation) and a few lines of code, a modeler can build an exploratory test framework for their cognitive model. Iterative changes in the model might drive evolution of the exploratory tests, while results of the tests might drive iterative changes in the model. Test results can be displayed in whatever format the modeler thinks is informative (e.g., printing to the terminal, communicating via a socket to a data visualization program, writing to a text file) and can evolve over time as well. In fact, this sort of exploratory testing technique is currently being used to test incremental changes to LETF's own code.

LETF also supports large-scale exploration of cognitive models by taking care of data aggregation and restructuring (e.g., calculating DVs, collapsing to determine central tendency) before outputting the results in a configurable format that can be coupled with specific parameter search algorithms and High Performance Computing (HPC) systems. For example, it was recently coupled with Moore's regression tree search algorithm 'Cell' (2010) to run a computational cognitive model of the Change Signal Task (Moore, Gunzelmann, & Brown, 2010) on the Maui High Performance Computing Center, Mana. In just a few minutes, LETF was configured to interface the model and the search algorithm (Cell) for an exploratory analysis that used over 6000 processor hours (running in less than 12 hours wall clock time) on Mana.

We recognize that there are a fair amount of exploratory tests that apply generally across architectures, and have provided a generic exploratory testing framework to easily specify those tests. Further, LETF has no API layer between the configuration file and the language that the framework was built in (i.e., lisp), allowing for an unbound number of tests that can be expressed. It is light enough to run a model on a standalone desktop computer through a small range of hand-coded configurations, and generic enough to couple with large scale exploratory tests on HPC clusters. Very much in the spirit of exploratory testing, LETF helps accelerate the pace that cognitive modelers can develop, test, and improve their computational models[1].

References

Kaner, C. (2004). The ongoing revolution in software testing. *Proceedings of Software Test and Performance Conference.* Baltimore, MD.

Moore, L. R., Jr., Gunzelmann, G., & Brown, J. W. (in press - 2010). Modeling Statistical Learning and Response Inhibition with the Change Signal Task. In *Proceedings of the 10th International Conference on Cognitive Modeling, Manchester, United Kingdom*.

Moore, L. R. (2010). Cognitive model exploration and optimization: a new challenge for computational science. In T. Jastrzembski (Ed.), *Proceedings of the 2010 Behavior Representation in Modeling and Simulation (BRIMS) Conference*. Orlando, FL: Simulation Interoperability Standards Organization.

[1] Please contact the author if you are interested in obtaining the source code

A Cognitive Model of the Acquisition and Use of Referring Expressions

Jacolien van Rij (J.C.van.Rij@rug.nl)
Center for Language and Cognition, University of Groningen
P.O.Box 716, 9700AS Groningen, The Netherlands

Hedderik van Rijn (D.H.van.Rijn@rug.nl)
Department of Psychology, University of Groningen
Grote Kruisstraat 2/1, 9712TS Groningen, The Netherlands

Petra Hendriks (P.Hendriks@rug.nl)
Center for Language and Cognition, University of Groningen
P.O.Box 716, 9700AS Groningen, The Netherlands

Keywords: ACT-R; language acquisition; processing efficiency; referring expressions; working memory

Introduction

Referring expressions are used to describe a person, object or event. Different referring expressions can be used to describe the same person or object. For example, to describe a specific person, one could use a full noun phrase (NP) such as *the pirate,* or a pronoun, such as *he*. However, in certain discourse contexts using a pronoun would lead to an incorrect interpretation for the listener. Adult speakers use a full NP instead of a pronoun in these cases, suggesting that adult speakers take into account the listener's perspective. In contrast, children up to the age of 6 prefer to use a pronoun in these cases. In this study, we investigate how children acquire adult-like performance on their use of referring subjects by modeling experimental data using the cognitive architecture ACT-R (Anderson, 2007). The cognitive model allows us to investigate the complex interaction between formal linguistic constraints and cognitive factors. In addition, the model generates detailed and testable predictions with respect to linguistic performance.

Experimental data

To test children's performance on the production and comprehension of pronouns in subject position, Wubs, Hendriks, Hoeks & Koster (2009) asked 31 4- to 7-year-old children and 23 adults controls to perform a production task, a comprehension task and a working memory task.

In their production task participants were asked to tell stories on the basis of series of six pictures (cf. Karmiloff-Smith, 1981). These stories were about two characters of the same gender. At the end of the story, the participants had to refer to the character that was introduced earlier in the story, but was not the current topic[1] of the story. Wubs et al. (2009) looked at the type of referring subject used to re-introduce this referent: a pronoun (*he*) or a full NP (*the pirate*). Selecting a pronoun would result in potential ambiguity for the listener, as pronouns are interpreted as reference to the current topic (a.o., Grosz, Weinstein, & Joshi, 1995). Adults mainly used full NPs (97%). However, children showed a preference for using pronouns (63%) over full NPs (34%) (see Figure 1). That is, children often produced pronouns that are unrecoverable for a listener. These results support the hypothesis that adults take into account the listener's perspective. In contrast, children seem to only take into account their own perspective as a speaker. They preferably use the most economical form, a pronoun.

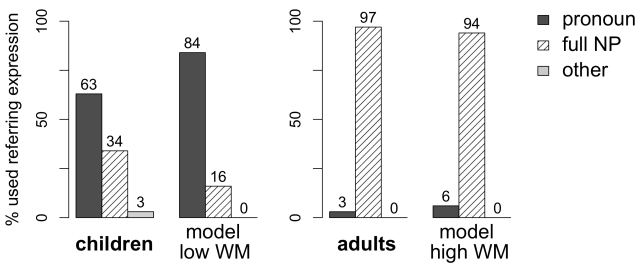


Figure 1: The type of referring subject used to re-introduce a character. The performance of the participants of Wubs et al. (2009)'s experiment is compared with the performance of our ACT-R model.

In the comprehension task of Wubs et al (2009), the same participants were asked to name the referent of an ambiguous subject pronoun at the end of pre-recorded stories with or without a topic shift. In contrast to adults, children showed no significant difference in their answers between the two types of stories. This suggests that they did not use discourse structure to resolve ambiguous pronouns. Notably, children's higher working memory scores were positively correlated with performance on the production and comprehension tasks.

Cognitive model

We have implemented a cognitive model within the cognitive architecture ACT-R (Anderson, 2007) to explain children's difficulties with the production and comprehension of referring subjects. In this model,

children's non-adult performance is caused by (i) lack of processing efficiency and (ii) limitations in working memory capacity (WM). Although we will only explain the acquisition of adult-like *production* of referring expressions, the same model can also explain the acquisition of adult-like *comprehension* by using the same mechanisms.

Processing efficiency

Adult speakers take into account the listener's perspective as a speaker to check whether the referring expression they intend to use can be interpreted correctly by the listener. As a result of this process, adult speakers will use a full NP to refer to a character that is *not* the current topic, because they know that a listener will interpret a pronoun as reference to the current topic. This process requires sufficient processing efficiency (as shown in a previous model of object pronouns, Van Rij, Van Rijn, & Hendriks, 2010). Initially, the model's processing is not efficient enough to carry out this process within a limited amount of time. Simulations show that the process gradually becomes more efficient as a result of frequent application of the same rules (i.e., production compilation mechanism of ACT-R, Taatgen & Anderson, 2002), ultimately resulting in adult-like performance.

Working memory capacity

In addition, the model needs to determine the current discourse topic for using the grammar correctly, because the model will also produce pronouns that are unrecoverable for the listener when it incorrectly determines that the character to be referred to is the current topic. The model implements the hypothesis that children have difficulties to incorporate previous discourse structures in their interpretation and use of referring expressions. For adults the subject of the previous utterance is often the most salient discourse referent (a.o, Grosz et al., 1995). However, children do not seem to use information about grammatical roles in determining the current topic as a result of their limited WM capacity. For children the saliency of discourse referents is only determined by their frequency and recency of mentioning in the discourse. This follows from our implementation of differences in WM as differences in source activation, i.e., the activation used to maintain task-relevant information (cf. Daily, Lovett, & Reder, 2001). Only when WM increases, will children be able to use grammatical information of the previous utterance to determine the current discourse topic.

To summarize, not only sufficient processing efficiency is necessary for adult-like production and comprehension of referring expressions in subject position (cf. Van Rij et al., 2010), but also sufficient WM capacity.

Future directions

Our cognitive model allows us to generate very precise and testable predictions with respect to linguistic performance, which can be tested with experiments. We are investigating two of the predictions of the model. The model predicts i) that in a situation of increased WM load, adults will show difficulties in determining the current topic, because WM capacity affects the ability to incorporate discourse structure in determining the current topic, and ii) that manipulating the frequency and recency of mentioning of characters in the discourse will affect low WM children's performance on the comprehension task more than manipulating the grammatical roles.

In addition, we are planning to re-implement the sentence-processing component, because the sentence-processing component of the model is highly simplified. With the re-implemented model that not only processes structural information (cf. Lewis & Vasishth, 2005), but also semantic and discourse information, we can investigate how discourse information, syntactic and semantic information interact in resolving ambiguous pronouns during on-line sentence comprehension.

Footnotes

[1] The discourse topic is the most salient referent in the current linguistic context, the discourse.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press, USA.

Daily, L. Z., Lovett, M. C., & Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science, 25*(3), 315.

Grosz, B. J., Weinstein, S., & Joshi, A. K. (1995). Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics, 21*(2), 203-225.

Karmiloff-Smith, A. (1981). The grammatical marking of thematic structure in the development of language production. In W. Deutsch (Ed.), *The Child's Construction of Language* (pp. 121-147). London: Academic Press.

Lewis, R. L., & Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science, 29*(3), 375-419.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition, 86*(2), 123-155.

Van Rij, J., Van Rijn, H., & Hendriks, P. (2010). Cognitive architectures and language acquisition: A case study in pronoun comprehension. *Journal of Child Language, 37*(3), 731-766.

Wubs, E., Hendriks, P., Hoeks, J., & Koster, C. (2009). Tell me a story! Children's capacity for topic shift. In J. Crawford, K. Otaki & M. Takahashi (Eds.), *Proceedings of the 3rd Conference on Generative Approaches to Language Acquisition North America (GALANA 2008)* (pp. 313-324). Somerville, MA: Cascadilla Press.

Contextual Memory for Goals: On the Role of Context, Attention, and Intention in Cognitive Control

Michel E. Brudzinski (brudzm@rpi.edu)
Cognitive Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180

Abstract

The contextual memory for goals (CMFG) model is presented as a theory of the role of context in cognitive control. CMFG has three components: (1) *contextual chunking*, (2) *perceptual priming*, and (3) *goal setting*. CMFG proposes that the contents of the cognitive buffers (perceptual, motor, intentional, etc.) become bound in declarative memory based on their co-occurrence during each cognitive cycle. The re-occurrence of buffer contents that have previously co-occurred spreads activation to associated chunks of memory. Goals are conceived of as declarative structures representing desired perceptual states that compete for control of cognition, are activated by perceptual priming, and are selected on the basis of activation. CMFG represents an integration of principles from Memory for Goals (MFG) model of cognitive control, Adaptive Control of Thought Rational (ACT-R), a unified theory of cognition, Perceptual Symbol Systems (PSS), and the Theory of Event Coding (TEC). CMFG will be examined in a series of experiments, implemented in the ACT-R cognitive architecture, and used to model experimental results.

Keywords: cognitive control; contextual memory; cognitive modeling; cognitive architecture; goals; feature integration.

Introduction

Goals are a central concept in cognitive control, representing the intentions of the cognitive system. Theories of the cognitive control of attention seek to explain how human behavior balances the need to be both (a) reactive; changing goals due to critical changes in the environment, and (b) proactive; maintaining goals over extended time periods, ignoring changes in the environment. Cognitive science needs good theories about the role of context, attention, and intention in the control of cognition.

Memory for Goals (MFG; Altmann & Trafton, 2002) is a theory of cognitive control that explains goal memory in terms of general declarative memory constructs, such as activation and associative priming, rather than using a special goal memory or control structure, such as a goal stack. Goals in memory compete for control of cognition. The goal with the highest instantaneous activation value becomes the active goal. MFG consists of three components: (1) *the interference level,* (2) *the strengthening constraint*, and (3) *the priming constraint*. Although MFG emphasizes the role of cues in cognitive control, it does not yet specify how cues become associated with goals. MFG has been implemented in cognitive models using the ACT-R cognitive architecture.

ACT-R is a cognitive theory and a production-rule based computational cognitive architecture that is used to model psychological processes (Anderson & Lebiere, 1998; Anderson et al, 2004). ACT-R lacks automatic, general-purpose mechanisms for associative memory, episodic memory, or contextual memory. Associations between modalities require specifically programmed declarative and procedural knowledge.

Goals in ACT-R are abstract symbols that can represent intentions at various levels of behavioral and temporal analysis. Goals are set by production rules and maintained in the goal buffer without cost. ACT-R needs a less ambiguous representation of intention so that goals can be created, suspended, and achieved by cognitive models, rather than by cognitive modelers.

Perceptual Symbol Systems theory (PSS) proposes that all mental representation, including abstract concepts and plans for action are inherently modal (Barsalou, 1999).

The Theory of Event Coding (TEC) proposes that perceptual and action symbols are bound into event files in memory based on their co-occurrence (Hommel, 2009; Hommel, Musseler, Aschersleben, & Prinz, 2001). TEC proposes that perception and action are representationally and functionally equivalent.

In this dissertation, I propose a computational mechanism for the role of contextual associative memory in cognitive control. This model integrates principles from MFG, TEC, and PSS into the ACT-R architecture.

Theoretical Framework

The Contextual Memory for Goals (CMFG) theory proposes that the attended features of perception are bound into contextual chunks based on there co-occurrence, prime the activation of actions and goals, with the highest activation goal driving cognition. CMFG consists of 3 components: (1) *contextual chunking*, (2) *perceptual priming*, and (3) *goal setting*.

Contextual chunking is a form of associative memory. It is the binding of features of the current context into a representation in declarative memory. The current context is conceived as being the contents of the cognitive buffers from ACT-R, and the contextual representations are similar to the event files proposed by TEC. The contextual chunk is limited in its representation of the context based on attention.

Perceptual priming is a form of spreading activation. The re-occurrence of a percept, that has been associated with a

goal in a contextual chunk, increases the activation of that goal. MFG, TEC and ACT-R, all propose priming by context.

Goal setting concerns both the focus and the form of intentions. The assignment of the active goal is based on instantaneous activation. The representation of intentions is based on the principle of common coding of perception and action. Goals are to-be-produced perceptual states.

Computational Implementation

CMFG will be computationally implemented in the ACT-R cognitive architecture in two forms: (1) using the standard architecture, and (2) using a modified architecture.

Standard ACT-R implementation

CMFG will be implemented in the ACT-R architecture using new and modified modules and buffers, relying on productions rules and Lisp functions calls to achieve CMFG's three theoretical principles.

Modified ACT-R implementation

CMFG will be implemented in the ACT-R architecture using new and modified modules and buffers to achieve CMFG's three theoretical principles (Figure 1).

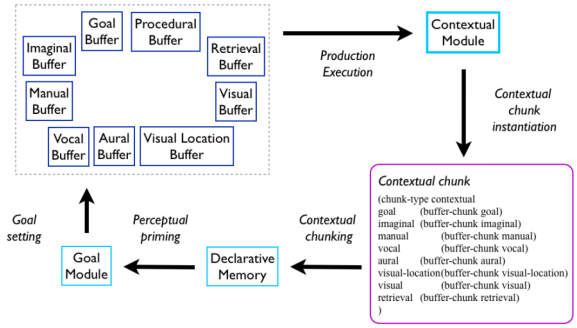


Figure 1: Proposed architectural changes to ACT-R.

Contextual chunking occurs within the architecture after the execution of a production rule. The new context module instantiates a contextual chunk with slots for each buffer in ACT-R, sets the value to be the value of the current chunk in each buffer, and harvests the chunk into declarative memory.

Perceptual priming occurs within the architecture through ACT-R's standard spreading activation mechanism. The breadth and depth of the pool of declarative chunks involved in spreading activation is massively increased by CMFG.

Goal setting occurs within the architecture through change to the operation of the goal module. The active goal is updated, through retrieval, on every production cycle, making the highest activation goal chunk the new goal buffer chunk. Goals are not be abstract concepts, but concrete imaginal buffer chunks.

Experiments

CMFG will be examined in four experiments, using two experimental paradigms. The first paradigm, Argus Army is an eye-tracked, computer-based environment. Experimental participants will learn associations between icons for military units and goals for action. The strength of these associations will be manipulated in a training phase and its effects will be examined during a testing phase in which participants will select the order of goals to pursue. The purpose of these experiments will be to demonstrate that CMFG can explain the process of learning cross-modal associations and that the activation of goals can predict priorities in goal-directed behavior.

The second paradigm, Coffee Challenge is a table-top, mobile-eye-tracked task. Experimental participants interact with abstract or real-world objects to perform a sequence of coffee-making actions.

Simulations

The two computational implementations of CMFG will be used in three simulations. Simulation 1 will use data from Hommel (2007), experiment 2, to demonstrate the ability of CMFG to account for response compatibility effects in a binary free-response task. Simulation 2 will model data from the Argus Army experiments. Simulation 3 will model data from the Coffee Challenge experiment. The ACT-R model, using CMFG, will connect to the Tekkotsu robotics framework (Touretzky et al, 2007) to control a custom-built robot consisting of 2 Crustcrawler AX-12 robotic arms, and a pan-and-tilt-capable webcam.

Conclusions

The Contextual Memory for Goals (CMG) theory includes 3 components: (1) *contextual chunking*, (2) *perceptual priming*, and (3) *goal setting*. The theory will implemented in the ACT-R computational cognitive architecture, supported by experimentation and computer simulation. CMFG represents a new embodied, reactive, distributed, automatic approach to cognitive control.

References

Altmann, E.M., Trafton, J.G. (2002). Memory for goals: An activation-based model. *Cognitive Science*, 26, 39-83.

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., & Qin, Yulin, Q. (2004). An integrated Theory of the Mind. *Psychological Review*, 111(4), 1036-1060.

Barsalou, L.W. (1999). Perceptual symbol systems. *Behavioral Brain Science*, 22, 577-660.

Hommel, B. (2009). Action control accoding to TEC (theory of event coding). *Psychological Research*, 73(4), 512-526.

Hommel, B. (2007). Feature integration across perception and action: event files affect response choice *Psychological Research*, 71, 42-63.

Touretzky, D.S., Halelamien, N.S., Tira-Thompson, E.J., Wales, J.J., & Usui K. (2007). Dual-coding representations for robot vision in Tekkotsu. Autonomous Robots, 22(4):425-435.

Long-Term Symbolic Memories for Long-Living Learning Agents

Nate Derbinsky (nlderbin@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Keywords: cognitive architecture; declarative memory; Soar.

Introduction

We humans, as prime exemplars of long-lived, learning agents, are frequently bombarded with dense and varying torrents of information, including data that is autobiographical (Laird & Derbinsky, 2009; Tulving, 1983), lexical (Miller, 1995), conceptual (Kolodner, 1983), and commonsensical (Lenat, 1995). Despite this deluge of experience, humans do not drown; we push forward, drawing on our knowledge and reasoning abilities to flourish in challenging and novel situations and tasks. The human cognitive architecture efficiently manages large stores of experience and supports precise retrievals, bringing to bear pertinent knowledge to effectively act in dynamic environments (Laird & Wray, 2010).

A review of prior psychological and computational work (Derbinsky & Laird, 2010) suggests that this robust behavior is due in part to our multiple, dissociated memory systems, citing significant functional and computational tradeoffs when utilizing a single memory mechanism for different types of learning tasks. While research into cognitive architectures for artificial learning agents typically reflects this dissociation strategy (Langley et al., 2009), significant work must still be done to understand the specific functionalities these memory systems must support to achieve human-level intelligence, as well as how to efficiently implement these mechanisms over long lifetimes.

In my thesis work, I seek to improve our functional and computational understanding of two long-term, symbolic memory systems, *episodic* and *semantic*, within the context of a general cognitive architecture. Semantic memory stores general facts that the agent *knows*, independent of the context in which they were originally learned, which can be applied to improve understanding and task performance in numerous, potentially unrelated situations. In contrast, episodic memory stores autobiographical, contextualized agent experience that allows an agent to *remember* its own past, such as recalling what occurred in similar situations and using that knowledge to decide how to act presently. I will first summarize my work with these memory systems to date, and then continue to my plans for future research.

Prior Work

My initial work has been to understand the computational challenges involved in extending the Soar cognitive architecture (Laird, 2008) with basic, task-independent episodic and semantic functionality that scales with large bodies of knowledge.

Episodic Memory

In Soar, episodic memory automatically stores and temporally indexes snapshots of the agent's current situation, which is represented as a directed, connected graph. To access episodic knowledge, the agent creates a symbolic cue, which represents contextualized features of interest in the episode to be retrieved. The retrieval mechanism then searches the episodic store for the best matching episode, biased by recency, and reconstructs the result in full for agent deliberation (Nuxoll & Laird, 2007).

We have found that in general, maintaining bounded episodic processing as the agent contends with multiple, complex tasks over long lifetimes presents a significant computational challenge (Laird & Derbinsky, 2009). However, we have developed data structures and algorithms that perform within reasonable limits in practice (Derbinsky & Laird, 2009). For instance, we have demonstrated sub-100msec. retrievals for a variety of cues on commodity hardware within a competitive tile-based game after 1 million episodes, each containing over 2500 features.

Semantic Memory

In Soar, semantic memory is a repository for long-term declarative knowledge. Sharing many similarities to the declarative memory module in ACT-R (Anderson et al., 2004), the semantic store can be conceived as a collection of chunks, each with features and relations to other chunks.

We have formulated and analyzed the computational challenges involved with supporting efficient access to large stores of declarative knowledge (Derbinsky et al., 2010). We demonstrated performance optimizations that support efficient retrievals over millions of declarative chunks. For instance, we presented sub-millisecond retrievals for many classes of cues on the entirety of the WordNet lexicon, consisting of more than 820K chunks.

Research Plan

My existing work has focused on efficiently supporting basic episodic and semantic functionality. I intend for future work to emphasize enhanced storage, retrieval, and functionality, while maintaining efficient performance.

Enhancing Storage

Episodic. Currently, Soar's episodic memory captures all details of the agent's current situation. One interesting

modification to this policy is to not encode in episodic memory the features and relations of semantic concepts, but instead allow the agent to reconstruct these details on-demand from the current contents of semantic memory. While this proposal shares some surface-level similarities with cognitive theories of human episodic reconstruction (Hassabis & Maguire, 2007), it will also reduce the amount of knowledge episodic memory must manage, thus improving storage and retrieval performance. It remains to be seen whether these performance gains outweigh the potential for inconsistencies and confusion that arises when integrating the contents of two long-term memory stores.

Semantic. Currently Soar, unlike ACT-R, has no automatic mechanism for storing new, and updating existing, declarative knowledge. I am interested in exploring architectural policies for storing agent experience. I am also interested in how episodic meta-data, such as the temporal stability of concepts and features, may be used to boost retrieval quality.

Enhancing Retrievals

In preliminary exploration, I have found that parallelism and heuristic search may be key to maintaining efficient retrievals, given large amounts of episodic and semantic knowledge. I plan to investigate these paths further in context of the extensions below.

Episodic. Soar scores episodic retrievals primarily on match cardinality, with recency used as a tie-breaking bias. I am interested in the degree to which feature activation, as well as overall episode appraisal, can improve match quality.

Semantic. The current declarative matcher in Soar implements a basic activation bias function. I am interested in efficiently incorporating some of the more extensive activation components in ACT-R, such as retrieval history and current context.

New Functionality

When comprehensive long-term memory systems are embedded within a general cognitive architecture, I am interested in the interfaces to agent experience, other than cue-based retrievals, that may be functionally beneficial to agent reasoning. For instance, as episodic memory encodes the current situation, it can aggregate the degree to which features are novel, an appraisal which may be useful for reasoning about actions (Mariner & Laird, 2008).

Evaluation

One major challenge of my proposed work is that there do not exist accepted benchmarks or metrics for comparing task-independent memory systems in context of a general cognitive architecture; thus proper evaluation is a research goal in and of itself. I foresee two categories of evaluation.

First, across a spectrum of problems, I intend to seek empirically optimal tradeoffs between computational resources (space and time) and task performance as I explore the mechanism changes described above.

Second, I intend to explore general cognitive capabilities, many of which we associate with human intelligence, supported by the availability and interaction of the semantic and episodic long-term memory systems within a single cognitive architecture. For instance, when choosing actions, an agent can "play forward" prior episodes with similar features and intentions, providing an agent a general and task-independent source of action evaluation knowledge.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., Qin, Y. (2004). An Integrated Theory of the Mind. *Pychological Review* 111, (4). 1036-1060.

Derbinsky, N., Laird, J. E. (2009). Efficiently Implementing Episodic Memory. *Proc. of the 8th International Conference on Case-Based Reasoning (ICCBR)*.

Derbinsky, N., Laird, J. E. (2010). Extending Soar with Dissociated Symbolic Memories. *Proc. of the Symposium on Human Memory for Artificial Agents, 36th AISB*.

Derbinsky, N., Laird, J. E., Smith, B. (2010). Towards Efficiently Supporting Large Symbolic Declarative Memories. *Proc. of the 10th International Conference on Cognitive Modeling*.

Hassabis, D., Maguire, E. A. (2007). Deconstructing Episodic Memory with Construction. *Trends in Cognitive Sciences* 11. 299-306.

Kolodner, J. L. (1983). Maintaining Organization in a Dynamic Long-term Memory. *Cognitive Science* 7, (4). 243-280.

Laird, J. E. (2008). Extending the Soar Cognitive Architecture. *Proc. of the First Conference on Artificial General Intelligence (AGI)*.

Laird, J. E., Derbinsky, N. (2009). A Year of Episodic Memory. *Proc. of the Workshop on Grand Challenges for Reasoning from Experiences, 21st IJCAI*.

Laird, J. E., Wray III, R. E. (2010). Cognitive Architecture Requirements for Achieving AGI. *Proc. of the Third Conference on Artificial General Intelligence (AGI)*.

Langley, P., Laird, J. E., Rogers, S. (2009). Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research* 10. 141-160.

Lenat, D. (1995). CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38 (11). 33-38.

Mariner, R., Laird, J. E. (2008). Emotion-Driven Reinforcement Learning. *Proc. of the 30th Annual Conference of the Cognitive Science Society (CogSci)*.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM* 38, (11). 39-41.

Nuxoll, A. M., Laird, J. E. (2007). Extending Cognitive Architecture with Episodic Memory. *Proc. of the 22nd Conference on Artificial Intelligence (AAAI)*.

Tulving, E. (1983). *Elements of Episodic Memory*. Oxford: Clarendon Press.

Towards Descriptive and Prescriptive Double-Loop Learning Agents

Ceyhun Eksin (ceksin@seas.upenn.edu)
Ackoff Collaboratory for Advancement of the Systems Approach (ACASA),
Department of Electrical and Systems Engineering,
University of Pennsylvania, Philadelphia, PA 19107 USA

Keywords: Double-loop learning

Introduction

The rise of complex computational models is due to the desire for white-box models with higher resolution of explanation and representation. Usually, the reason for complexity within models is because we are trying to explain real world phenomena that include humans. Descriptive qualitative and quantitative models of human behavior have mainly been the goal of social sciences (psychology, economy, sociology etc.) as well as fields such as cognitive science. Most of the time, studies are primarily interested in behavior within a specific context or situation in that domain. Therefore, generated theories are restricted to apply within the domains that they are designed for, constrained by further assumptions. Hence, often a single theory is not sufficient to properly represent human behavior in an evolving or dynamic socio-economic systems model. This makes a systems approach that contains adaptive feedback mechanisms to this problem necessary.

A possible framework that highlights this kind of mechanism is *double-loop* learning (Argyris and Schon, 1978) (Figure 1). The first loop of learning is based on an existing mental model (Johnson-Laird, 1983). A mental model is an implicit internal image of how the current system works (Senge, 1990). In other words, mental models can be interpreted as the theory that results in a strategy or decision making mechanism such as a heuristic. Most of the behavioral theories and heuristics can be interpreted as possible mental models that we utilize under certain conditions. The single loop learning only considers the existing mental model and modifies it based on information fed back, i.e. consequences of our actions. In this loop, the way we view the world does not change and we just make fine tuning adjustments on the existing mental model. The second loop of learning is where we consider whether our current mental model is still satisfactory to explain the world dynamics or not. Within a system, certain behavior around us might lead to a paradigm shift in our explanation or we might explain certain situations with one mental model and other situations with other sets of mental models. Hence, our reasoning mechanism adapts to the changes in the world. Although humans are capable of doing double-loop learning, none of these learning loops is done perfectly. Therefore, a descriptive human behavior model based on a double-loop learning framework would have to reflect human faults in application. A prescriptive approach would point to our faults in the learning processes and in our mental models. Hence, an ideal agent would utilize correct mental models, rules, and/or heuristics at the right time with correct settings in a complex system. In this study, I plan to provide a general framework for descriptive and prescriptive (ideal) models of double-loop learning.

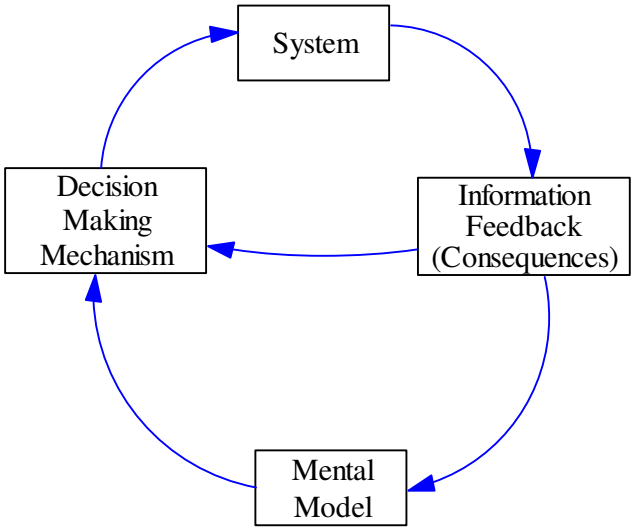


Figure 1: Double Loop Learning

Methodology

While it is possible to discuss optimal decision making and convergence in tractable analytical models, these discussions do not often apply to real world complex computational agent-based models. Complex computational models are not analyzable via conventional analysis methods hence the concept of optimality is void within this domain. In complex systems, we often define metrics of performance that indicate how well we are doing and we can discuss ideal approaches based on these metrics. These metrics can be common to everyone or can be our subjective goals. In a double-loop learning process, the first loop of learning refers to tuning of the existing decision-making process, i.e. updating model parameters based on these metrics. Learning in the sense of artificial intelligence or evolutionary algorithms would provide ideal ways to do single loop learning. Heuristics, local algorithms and other theories would provide descriptive ways of doing this update procedure. A general framework for second-loop learning requires a methodology for model comparison. An ideal approach would answer the question: "Which decision-making process is the best in this case?" A descriptive approach relates to literature on model comparison, validation and verification. Literature on these processes tries to answer questions such as "Which model is

the one that explains human behavior the best in this situation?" or "Does my model provide a valid explanation and representation of human behavior?" The metrics will help to come up with a general framework that governs how to update mental models, when to replace them and what models to replace them with. The first loop considers performance metrics based on the system where as the second loop can consider a more general system independent set of metrics.

The final part of the research deals with coming up with a set of descriptive or normative and 'optimal' decision making models (heuristics, rules, theories etc.) that work under certain conditions. The set of models will depend on the system and will not consist of algorithms that work for a general problem and include no reasoning (such as search algorithms).

Current and Future Research

There are two paths of research that can be pursued in parallel. The first deals with developing a methodology for metrics of comparison amongst models within the same framework. The author has ongoing research on this topic that attempts to evaluate validity and performance of alternative models within a complex system (Eksin et al., 2010). The second path deals with computational representations of descriptive and ideal double-loop learning mechanisms. The descriptive part relates to knowledge based search and reasoning literature. The ideal part will relate to artificial intelligence (stochastic games, learning) and control theory (adaptive control, fuzzy logic, stochastic optimization algorithms) literature. Fields such as artificial intelligence and control theory provide a set of adaptive and feedback-based algorithms that can still work in practice for complex systems. The author's current work in this track includes incorporating a Q-learning agent to a complex system (Eksin, 2010). Although the study was approached from the perspective of policy design for an agent, one can utilize similar kinds of learning algorithms to establish ideal forms of first and second loops of learning. In this study, the agent is essentially a Q-learning agent which by definition does a first loop of learning until convergence. However, the Q-learning mechanism misses a second loop where either the update mechanism or the state summary that is used for decision-making is modified. Future work would also concentrate on a literature review on existing approaches to reasoning to utilize a descriptive double loop learning agent.

Application

After coming up with a general framework for double loop learning, they will be applied to two socio-cognitive agent based models: 1) A country with factional-conflicts and existing insurgency (Silverman et al., 2007), and 2) A village model that includes tribal and ethnic differences. The first task would be to develop a list of descriptive and prescriptive theories, heuristics and rules that is relevant to the model. There will be two sets of experiments on each model. On the first set of experiments, the model will only have a single agent that has double-loop learning. The double-loop learning agent decision making mechanism will be descriptive in one case and prescriptive in the second case. These cases will be compared to the benchmark cases where none of the agents are double-loop learners. The second set of experiments will have all the agents as double-loop learners. Similarly, in one case all the agents will have descriptive models of a decision making mechanism and in the second case all agents will utilize prescriptive models. In this set of experiments, I will be looking at the changes in the emerging behavior within the system compared to benchmark cases.

Conclusion

In this paper, I present an overview of a research thread that will produce a descriptive computational model framework, or double-loop learning, that will require use of multiple alternative descriptive theories as mental models. Additionally, the framework can be a prescriptive model that utilizes correct mental models at the right time. I propose to make use of multiple literatures to develop both descriptive and prescriptive models of double-loop learning.

References

Argyris, C., & Schon, D. (1978). *Organizational Learning: A Theory of Action Perspective.* Reading: Addison Wesley.

Eksin, C. (2010). Policy Analysis Using Q-Learning. *Poster presentation at 19th Behavior Representation in Modeling and Simulation.* Charleston, SC.

Eksin, C, Silverman, B.G., Pietrocola, D. & Kang, R. (2010). Dimensions of Leader-in-Context Models. *Submitted to the International Conference on Cognitive Modeling 2010.* Philadelphia, PA.

Johnson-Laird, P.N. (1983). *Mental Models: Toward a Cognitive Science of Language, Inference and Consciousness.* Harvard University Press.

Senge, P.M. (1990). *The fifth discipline : the art and practice of the learning organization.* New York: Doubleday

Silverman, B. G., Bharathy, G., K., Nye, B.,Eidelson, R. (2007). Modeling Factions for 'Effects Based Operations': Part I – Leader and Follower Behaviors. *Journal of Computational & Mathematical Organization Theory, 13*, 379-406.

Learning to Use Memory

Nicholas A. Gorski (ngorski@umich.edu)
Computer Science & Engineering, University of Michigan
2260 Hayward St., Ann Arbor MI 48109-2121 USA

Keywords: Reinforcement learning; memory; working memory; declarative memory.

Introduction

Reinforcement learning (RL) provides a general approach to support intelligent agents that learn to act in their environments (Sutton & Barto, 1998). The foundational reinforcement learning algorithms of Q-Learning and SARSA, however, are purely reactive and thus not generally applicable to problems in which knowledge must be maintained in memory.

My research focuses on investigating how memory can extend the range of possible behaviors that RL can achieve, and in particular how RL agents can learn to use biologically-inspired memory models. In this context, *using memory* has two senses: first, making use of the knowledge that is retrieved from memory in order to better perform the task at hand, thus making use of the declarative knowledge from memory; second, selecting actions (such as encoding, storage and retrieval) over memory as appropriate for the task, thus using memory through procedural knowledge. One view of this research is that it is an attempt to discern which procedural knowledge over memory *must* be architectural and which *may* be adaptive.

Some prior work has begun to investigate this direction. We demonstrated that it is possible to learn to use a human-inspired episodic memory model in certain specific cases, but that in others an agent cannot learn the optimal control strategy (Gorski & Laird, 2009). Other researchers have also found that RL agents endowed with episodic and working memory models can learn to achieve some tasks, but not others (e.g. Zilli & Hasselmo, 2007).

My primary research question is: how and when can RL be used to learn to use memory? To address this in my thesis, I will perform a comprehensive empirical exploration of learning to use memory in order to better understand the dynamics that arise when an RL agent is endowed with an internal memory model. I will identify characteristics of tasks that can be explored independently across sets of parameterized problems. My initial exploration will begin with three memory models: a simple bit memory model, a gated working memory model (inspired by human working memory), and an associative memory model (inspired by human episodic memory). I precede a more detailed discussion of my research plans with an overview of my research to date.

Progress to Date

My research initially focused on learning to use Soar's episodic memory model (Derbinsky & Laird, 2009; Laird, 2008). Nuxoll (2007) had previously identified a set of cognitive capabilities that could be supported by episodic memory, and demonstrated agents that performed a subset of these capabilities. However, these agents required significant background knowledge and performed no learning. We studied whether it was necessary to provide the knowledge to utilize these cognitive capabilities, or whether RL could learn to use episodic memory in specific ways, and eventually performed specific cognitive capabilities solely as an emergent response to environmental and architectural constraints and pressures.

We succeeded in demonstrating agents that learned to perform two specific cognitive capabilities: virtual sensing, in which an agent uses episodic memory to recall a portion of the environment state that it cannot directly perceive; and remembering past actions, in which an agent uses knowledge of past actions to guide current behavior (Gorski & Laird, 2009).

In the course of this work, we found three interesting results. First, trivial-seeming changes to the environment had dramatic effects on how well agents were able to learn to use memory. Similarly, it can be very difficult to construct a task that is "just right" such that it elicits the desired cognitive capability and in which an agent uses memory in desired way.

Second, it is significantly easier to learn to perform virtual sensing than to use the knowledge that results from remembering past actions. When learning to perform virtual sensing, the agent was retrieving knowledge from memory that was a reliable indicator of the state of the environment, regardless of the duration of the agent's existence. However, knowledge of past actions was useful only after the agent had converged to a relatively stable behavior in the environment, as the knowledge that was retrieved was more sensitive to interference effects of taking a related action at an inopportune time.

Third, in certain settings agents converged to nearly optimal behaviors, but used episodic memory essentially as a single bit of memory (similar to the bit memory of Littman, 1994). Even though the learned behavior was suboptimal, it was a sufficiently stable equilibrium such that the agent was not able to find the globally optimal behavior through additional exploration.

The third result motivated us to explore using a bit memory model in the same domain (Gorski & Laird, forthcoming). In this work, we determined that while bit memory was sufficiently capable of being used to represent the optimal policy when the agent was provided with some initial background knowledge, the agent could not learn to use bit memory effectively. We additionally identified

important ways in which bit memory differed from the episodic memory model.

Agents learning to use memory were sensitive to small changes in the task specification; furthermore, the behaviors of agents using different memory models were very different in the same domain. These results motivated a more comprehensive exploration of the space of tasks and memory models.

Research Plan

In order to understand the dynamics of learning to use memory, I propose a methodical and comprehensive empirical exploration of the space of possible tasks and memory models. As the space of possible tasks and memory models is infinite, it will be necessary to focus my empirical study on a particular set of tasks and memory models, which will be used to draw conclusions that can apply to tasks and memory more generally.

The tasks that I will explore have been selected on the basis of understanding how varying specific aspects (or characteristics) of a task affect the ability to learn to use memory in it. We have identified a very simple task, inspired by T-Maze tasks from the experimental psychology literature, that can be parameterized across independent dimensions. When these dimensions correspond to characteristics that are relevant to how memory must be used in a task, then observing the behavior that emerges in those tasks will inform how learning to use memory scales and what patterns of behavior take place in the course of the learning process.

The task characteristics that we are primarily interested in are those that directly relate to how knowledge must be retained while performing a task (we refer to this knowledge that must be maintained over time as *salient knowledge*). These characteristics include:

- The temporal delay between when salient knowledge is acquired and a task action that depends on it
- The quantity of salient knowledge that must be maintained simultaneously in a task
- The number of actions in a task that depend on salient knowledge.

I have identified a preliminary set of tasks that are parameterized along these relevant characteristics.

Exploring the space of memory models will require a different approach. While it is possible to design tasks that isolate individual characteristics and explore them over a parameterized task set, a given memory model cannot exist without architecturally committing to a number of simultaneous points in the various dimensions that define a memory model. Therefore, we will explore the space of memory models using a top-down approach.

We will explore bit memory, gated working memory, and an associative long-term memory in the context of the set of tasks discussed above. In a first pass, we will perform a comprehensive sweep exploring artificial agents that learn to use each memory model across all tasks (the cross product of memory models and tasks). After analyzing the results of this study, we will then modify the three memory models in an attempt to explore functional differences that they exhibit when an agent learns to use them, so as to be able to determine which characteristics of memory are directly responsible for supporting the necessary learning behavior, or not supporting it.

Throughout my investigation, my focus will be on the dynamics that arise between memory and task. I intend to be agnostic regarding specific RL algorithms as much as possible, and consistently apply the same algorithm (e.g. SARSA, Sutton & Barto, 1998) in all of my experiments.

My evaluation will focus on two issues: how agent performance scales with characteristics of task, and which characteristics of memory are most directly tied to which task characteristics.

Although my research is grounded in the field of artificial intelligence, I aim to draw conclusions from my work that inform cognitive scientists as to the nature of how procedural knowledge that uses memory (both controls it and makes use of the knowledge from it) can be learned. While most memory models assume some architectural basis for certain internal actions over memory, such as encoding and storage to long-term declarative memory, the procedural knowledge that governs memory retrievals and how that retrieved memory impacts task performance is adaptive. By better understanding in which tasks it is computationally feasible to learn to use specific memory models, we might better understand the constraints on human memory (and learning). In the field of artificial intelligence, learning to use memory is one approach to answering challenging problems of overcoming tasks with incomplete information while maintaining responsive learning and decision making.

References

Derbinsky, N. & Laird, J.E. (2009). Efficiently Implementing Episodic Memory. *Proceedings of the 8th Intl. Conf. on Case-Based Reasoning*. Seattle, WA.

Gorski, N.A. & Laird, J.E. (2009). Learning to Use Episodic Memory. *Proceedings of the 9th Intl. Conf. on Cognitive Modeling*. Manchester, UK.

Gorski, N.A. & Laird, J.E. (forthcoming). Learning to Use Episodic Memory. *Under Review.*

Laird, J.E. (2008). Extending the Soar Cognitive Architecture. *Proceedings of the 1st Conf. on Artificial General Intelligence*.

Littman, M.L. (1994). Memoryless Policies: Theoretical Limitations and Practical Results. *Proceedings of the 3rd Intl. Conf. on Simulation and Adaptive Behavior.*

Nuxoll, A. (2007). Enhancing Intelligent Agents with Episodic Memory. Ph.D. diss., Computer Science & Engineering, U. of Michigan, Ann Arbor.

Sutton, R.S. & Barto, A.G. (1998). Reinforcement Learning: An Introduction. Cambridge: MIT Press.

Zilli, E.A. & Hasselmo, M.E. (2007). Modeling the Role of Working Memory and Episodic Memory in Behavioral Tasks. *Hippocampus, 18*, 193-209.

Understanding Strategic Adaptation in Multitask Settings

Christian P. Janssen (c.janssen@ucl.ac.uk)
University College London, Gower Street, London WC1E 6BT, UK

Introduction

How do people interleave their attention when performing multiple tasks, such as dialing a phone number while driving, or checking e-mail while writing a paper? To investigate these issues a variety of modeling frameworks have been used, for example EPIC (Meyer & Kieras, 1997), SOAR (Lallement & John, 1998), ACT-R Threaded Cognition (Salvucci & Taatgen, 2008) and Cognitively Bounded Rational Analysis models (Howes, Lewis, & Vera, 2009). The majority of these frameworks focus on understanding how multiple tasks interfere with each other, for example as a result of having limited resources (e.g., two eyes, two hands) to dedicate to each task.

Within the cognitive modeling community, relatively less attention is given to understanding how more top-down aspects, such as instructions and priorities, interact with these architectural aspects. However, some exploration has been done elsewhere. For example, it has been demonstrated that people adapt their performance to instructions to spend more time on a task (e.g., Gopher, 1993), or to changes in payment associated with performance (e.g.,Wang, Proctor, & Pick, 2007). In situations like these, the adaptation process can be understood as making trade-offs between performance on each of the tasks (e.g., Navon & Gopher, 1979; Norman & Bobrow, 1975).

In my doctoral dissertation work I try to understand this flexible adaptation of dual-task performance, where people interleave attention in different ways despite being exposed to the same stimuli. As a modeling approach, I use Cognitively Bounded Rational Analysis Models (Howes, et al., 2009). However, I also have an interest in informing and using other architectural frameworks.

CBRA Models of Multitasking

So far, my work has focused on developing explanations of human multitasking behavior for two dual-task settings: (1) manually dialing a phone number while driving a simulated car, and (2) typing digits while tracking a cursor. In both domains, the central questions are: when is attention for one task interleaved to pay attention to the other task, how is this moderated by the set priorities, and why is attention interleaved in this specific way?

Our first dual-task setting, manually dialing a phone number while driving a simulated car, has been well studied before. One way of understanding interleaving in this situation is that people make use of "natural break points": a prevalent task structure in which some points are more natural to interleave performance than others (Salvucci, 2005). However, whether this structure is used depends on the priority that the driver sets (Brumby, Salvucci, & Howes, 2009; Janssen & Brumby, in press; Janssen, Brumby, & Garnett, 2010). If the priority is to dial the number as fast as possible, more digits are dialed consecutively before turning attention back to driving, often omitting natural break points. When the priority is to drive as safe as possible, participants interleave dialing for driving at the natural breakpoints, and at more positions if these points are not sufficient (Janssen & Brumby, in press). Using a cognitively bounded rational analysis model we demonstrated the trade-offs that drivers make in these situations (Janssen & Brumby, in press).

While the above work illustrates the trade-offs that are made between tasks, it does not illustrate why a *specific* way of performing the task is chosen (Howes, et al., 2009). In the driving studies we found that a different number of digits is dialed in sequence before interleaving dialing for driving depending on the set priority. But why were not more (or less) digits typed?

Howes et al. (2009) argue that in order to understand what it is the cognitive system is adapting to it is important to specify an explicit objective function that determines the quality of a given task interleaving strategy (Howes, et al., 2009). Based on this assessment, the strategy with the highest payoff can be determined and compared with human performance. We applied this methodology in a new task paradigm in which participants have to track a cursor with a joystick while typing in a series of digits as fast as possible (Janssen, Brumby, Dowell, & Chater, 2010a). Critically, participants can only control one task at a time (i.e., they can either type a series of digits, or track the cursor) and have to determine how many digits they type in one sequence and how much time they spend on tracking. Experimental results show that participants adapt their strategy to the difficulty of the task, making trade-offs in task performance. A succeeding modeling effort demonstrated why participants adapted their strategy: the adopted strategies maximized their pay-off. In this sense, the explanation given by our model went beyond traditional demonstrations of performance trade-offs.

Conclusions and Future Work

The preceding work has demonstrated that multitasking participants adapt their performance not only to stimuli characteristics, but also to more internal characteristics such as priorities and instructions. Our modeling work demonstrated why certain trade-offs are made: participants trade-off performance on one task versus performance on the other task. In addition, our more recent work was able to demonstrate that participants not only adapt performance to instructions, but that they also try to adapt in an *optimal*

way, to maximize pay-off (Janssen, Brumby, Dowell, & Chater, 2010a).

In the remainder of my PhD I want to take this work further in a couple of novel angles. First of all, I want to explore models of individual differences in performance. Cognitively Bounded Rational Analysis models describe *spaces* of performance (instead of just one strategy for performance, as is often the case in production rule systems). Given that there often is a variety of ways in which tasks can be performed, it seems unlikely that participants only act in one way. By fitting cognitive models to individual characteristics (e.g., typing speed), I want to explore whether rational strategies for multitasking can be explained at an individual level (cf. ,Howes, et al., 2009).

Another angle of future research is to investigate *how* optimal performance is learned. My current work has mainly focused on explaining *why* performance is adapted (to maximize pay-off, or to suit an instruction). However, it does not explain *how* performance is adapted given experience. Using Cognitively Bounded Rational Analysis models I want to demonstrate that if participants have to learn to interleave two tasks, they change their strategies over time by (systematically) moving performance towards the optimum strategy. In addition, I want to look at other modeling frameworks to see how these models would explain performance. In particular the theory of Threaded Cognition is appealing, as it is one of the most integrated and unifying theories of multitasking (being able to explain performance across a range of multitask settings with different time scales, Salvucci, Taatgen, & Borst, 2009). Moreover, as this theory is integrated within a cognitive architecture, it is relatively easy to combine theories of multitasking with theories of for example skill learning. For some initial ideas on this see (Janssen, Brumby, Dowell, & Chater, 2010b). At the doctoral consortium I hope to further discuss these and other ideas.

Acknowledgments

This work was supported by EPSRC grant EP/G043507/1. I would like to thank my advisors Duncan Brumby, John Dowell and Nick Chater for their support and advice.

References

Brumby, D. P., Salvucci, D. D., & Howes, A. (2009). Focus on driving: How cognitive constraints shape the adaptation of strategy when dialing while driving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1629-1638). New York, NY: ACM Press.

Gopher, D. (1993). The skill of attention control: Acquisition and execution of attention strategies. In D. E. Meyer & S. Kornblum (Eds.), *Attention and performance XIV: Synergies in experimental psychology, artificial intelligence, and cognitive neuroscience* (pp. 299–322). Cambridge, MA: MIT Press.

Howes, A., Lewis, R. L., & Vera, A. (2009). Rational adaptation under task and processing constraints: Implications for testing theories of cognition and action. *Psychological Review, 116*, 717-751

Janssen, C. P., & Brumby, D. P. (in press). Strategic adaptation to performance objectives in a dual-task setting. *Cognitive Science*.

Janssen, C. P., Brumby, D. P., Dowell, J., & Chater, N. (2010a). A cognitively bounded rational analysis model of dual-task performance trade-offs. In *Proceedings of the 10th International Conference on Cognitive Modeling 2010*. Philadelphia, NJ.

Janssen, C. P., Brumby, D. P., Dowell, J., & Chater, N. (2010b). Strategic adaptation in a dual-task setting. In *Proceedings of the First European ACT-R Workshop*. Groningen, The Netherlands.

Janssen, C. P., Brumby, D. P., & Garnett, R. (2010). Natural break points: Utilizing motor cues when multitasking. In *Proceedings of the 54th annual meeting of the Human Factors and Ergonomics Society*. San Francisco, CA, USA: Human Factors and Ergonomics Society.

Lallement, Y., & John, B. (1998). Cognitive architecture and modeling idiom: An examination of three models of the wickens task. *Proceedings of the twentieth annual conference of the Cognitive Science Society*, 597-602.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review, 104*, 3-65.

Navon, D., & Gopher, D. (1979). On the economy of the human-processing system. *Psychological Review, 86*, 214-255.

Norman, D. A., & Bobrow, D. G. (1975). On data-limited and resource-limited processes. *Cognitive Psychology, 7*, 44-64.

Salvucci, D. D. (2005). A multitasking general executive for compound continuous tasks. *Cognitive Science, 29*, 457-492.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review, 115*, 101-130.

Salvucci, D. D., Taatgen, N. A., & Borst, J. P. (2009). Toward a unified theory of the multitasking continuum: From concurrent performance to task switching, interruption, and resumption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY: ACM Press.

Wang, D. D., Proctor, R. W., & Pick, D. F. (2007). Acquisition and transfer of attention allocation strategies in a multiple-task work environment. *Human Factors, 49*, 995-1004.

Recognizing Behaviors and the Intentional State of the Participants

Wesley Kerr (wkerr@cs.arizona.edu)
Department of Computer Science
1040 E. 4th Street
Tucson, AZ 85721 USA

Keywords: activity recognition; intention recognition; time series;

Introduction

Psychological research has demonstrated that subjects shown animations consisting of nothing more than simple geometric shapes perceive the shapes as being alive, having goals and intentions, and even engaging in social activities such as chasing and evading one another (Blythe, Todd, & Miller, 1999; Heider & Simmel, 1944). While the subjects could not directly perceive affective state, motor commands, or the beliefs and intentions of the actors in the animations, they still used intentional language to describe the moving shapes. For example, subjects in the Heider and Simmel (1944) study consistently labeled the larger triangle, shown in Figure 1, as a bully who harassed the smaller triangle and circle.

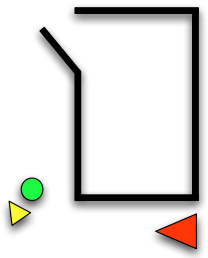


Figure 1: Single frame from an animations similar to the original Heider and Simmel animation.

When subjects ascribe intentions to geometric primitives like those shown in Heider and Simmel's research (see Figure 1), which information guides the process? Blythe et al. (1999) showed that the motion of the actors in animations is sufficient to classify the activities that occur in the animations. The system generated to perform classification even outperformed human subjects on the same task.

Blythe's system mapped patterns of motion onto class labels for intentional states, which isn't quite the same as knowing anything about intentional states. One of Heider and Simmel's subjects described the larger triangle in Figure 1 as "blinded by rage and frustration." Blythe's system couldn't come up with such a description. An agent that classifies episodes by patterns of motion knows about patterns of motion, not about rage and frustration, even if these words are provided as episode labels. So how might an agent infer affective states?

In both the Heider and Simmel animations and the animations developed by Blythe et al., subjects can only observe a subset of the features that are available, i.e. positions, velocities, sizes, colors, etc. The subjects cannot directly perceive the affective state, motor commands, and the beliefs and intentions of the actors in the animations. Yet they infer affective states and describe them with intentional language. We think humans infer affective states given non-affective observables such as positions and velocities by calling on their own affective experiences. Observables cue, or cause to be retrieved from memory, schemas that include learned affective components, which are inferred or "filled in" as interpretations of patterns of motion or other non-affective observables.

In this dissertation, we present representations and algorithms that enable an artificial agent to correctly recognize other agents' activities by observing their behavior. In addition, we demonstrate that if the artificial agent learns about the activities through participation, where it has access to its own internal affective state, motor commands, etc., it can then infer the unobservable affective state of other agents.

Activity Recognition

We begin with definitions: An *episode* is a collection of *intervals*. Each interval is a tuple containing a proposition and the times at which the proposition becomes true and false. A proposition can become true (and false) multiple times within an episode; each of these instances is represented as a separate interval. Each episode is given a class label and is a single example of an activity. In the activity recognition task we are given a collection of episodes for training, and then tested on episodes that were not part of the training set.

We assume that different examples of one activity share patterns of intervals. More colloquially, the intervals in similar episodes tell the same story with minor variations. Thus, one may classify episodes by their constituent patterns of intervals. This is not the only way to do it: A cleaning agent might classify a cleaning episode by the objects it interacts with, such as pots and pans, rather than what was done with the pots and pans. But our focus here is classifying episodes by patterns of activities, represented by intervals.

Episodes and intervals have different durations, start times, end times, and constituent propositions, so our representation of episodes must be able to accommodate and generalize over these variations. For example, the activity "capture" involves one agent chasing another agent until the second agent is cornered or held in a single place. The participants might be a prisoner and a guard or some other pair of agents, and the amount of time spent chasing can vary from minutes to hours,

but all episodes share the same common pattern: One actor chasing another until the other agent is cornered or caught.

Relationships between intervals can be described by *Allen relations* (Allen, 1983). Allen recognized that, after eliminating symmetries, there are only seven possible relationships between two intervals. Allen relations are qualitative in the sense that they represent the temporal order of events, specifically, the beginnings and endings of intervals, but not the durations of intervals.

Our episode representation, which we call a *qualitative sequence*, is a sequence of Allen relations between intervals in the episode. We construct the sequence by combining the Allen relations between *all* of the pairs of intervals in the order in which the Allen relation completes. An illustrative episode and the resulting qualitative sequence is shown in Table 1. The letters **A, B** and **C** denote propositions, and an assertion such as (**C** 1 3) means that proposition **C** was true in the interval [1,3].

Intervals	Sequence
	(**C** *meets* **A**)
(**C** 1 3)	(**C** *before* **B**)
(**A** 3 6)	(**A** *overlaps* **B**)
(**B** 4 9)	(**C** *before* **C**)
(**C** 6 10)	(**A** *meets* **C**)
	(**B** *overlaps* **C**)

Table 1: An episode comprising four intervals and the corresponding qualitative sequence.

Episodes are first converted into qualitative sequences of Allen relations and learning is done with these sequences. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a set of qualitative sequences with the same activity label. We define the *signature* of the activity label, $\mathcal{S}_c$, as an ordered sequence of *weighted* Allen relations. (The only difference between a signature and a qualitative sequence is these weights.) We select a sequence at random from $\mathcal{S}$ to serve as the initial signature, $\mathcal{S}_c$, and initialize all of its weights to 1. After this, $\mathcal{S}_c$ is updated by combining it with the other sequences in $\mathcal{S}$, processed one at a time.

Two problems are solved during the processing of the sequences in $\mathcal{S}$. First, the sequences are not identical, so $\mathcal{S}_c$ must be constructed to represent the most frequent relations in the sequences. The weights in $\mathcal{S}_c$ are used for this purpose. Second, because a relation can appear more than once in a sequence $\mathcal{S}_i$, there can be more than one way to align $\mathcal{S}_i$ with $\mathcal{S}_c$. These problems are related because the frequencies of relations in $\mathcal{S}_c$ depend on how sequences are successively aligned with it.

Updating the signature $\mathcal{S}_c$ with a sequence $\mathcal{S}_i$ occurs in two phases. In the first phase, $\mathcal{S}_i$ is optimally aligned with $\mathcal{S}_c$ using the Needleman-Wunsch global sequence alignment algorithm (Needleman & Wunsch, 1970). The alignment algorithm penalizes candidate alignments for relations in $\mathcal{S}_c$ that are not matched by relations in $\mathcal{S}_i$, and rewards matches. These penalties and rewards are functions of the weights stored with the signature. In the second phase, the weights in the signature $\mathcal{S}_c$ are updated. If a relation in $\mathcal{S}_i$ is aligned with one from $\mathcal{S}_c$, then the weight of this relation is incremented by one. Otherwise the weight of the relation is initialized to one and it is inserted into $\mathcal{S}_c$ at the location selected by the alignment algorithm.

The signatures function as classifiers as follows. Recall that $\mathcal{S} = \{S_1, \ldots, S_k\}$ is a set of qualitative sequences with the same activity label; for example, all the sequences in $\mathcal{S}$ might be examples of *jump over*. Now suppose we have N sets of qualitative sequences, $\Sigma = \{\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^N\}$ each of which has a different activity label, and its own signature. A novel, unlabeled sequence matches each signature to some degree, determined by aligning it with each signature, as described earlier. The novel sequence is given the activity label that corresponds to the signature it matches best.

Inferring Hidden State

Episodes have observable and unobservable propositions depending on which agent is doing the observing. For example, when $agent_1$ is chasing $agent_2$, $agent_1$ observes all of the propositions pertaining to its motor commands, emotional state, and intentional state, but when $agent_1$ observes $agent_3$ chasing $agent_2$, $agent_1$ cannot perceive the motor commands, emotional state, and intentional states of $agent_2$ nor $agent_3$.

By *hidden relations* we mean relations that include one or more propositions that are not directly observable in the behavior of other agents, and so must be inferred. Our approach to inferring hidden relations is to have agents learn signatures of their own behaviors, in which these relations are *not* hidden. Then, when an agent observes another's behavior, it matches the observable relations to signatures of its own behavior, and uses these to infer unobservable relations in other's behavior.

In general, sequences can contain many hidden relations. The most frequent are the most likely when observing other agents. Therefore, our agent selects the most frequently occurring hidden relations to be the inferred hidden state.

References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, *26*(11), 832–843.

Blythe, P. W., Todd, P. M., & Miller, G. F. (1999). How motion reveals intention: Categorizing social interactions. In *Simple heuristics that make us smart.* Oxford University Press, USA.

Heider, F., & Simmel, M. (1944). An experimental study of apparent behavior. *The American Journal of Psychology*, *57*(2), 243.

Needleman, S. B., & Wunsch, C. D. (1970, March). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, *48*(3), 443-453.

A probabilistic model of phonetic cue restructuring

James P. Kirby (jkirby@uchicago.edu)
University of Chicago, Department of Linguistics,
1010 E. 59th St., Chicago, IL 60622 USA

Keywords: Phonetic change; speech perception; agent-based modeling; categorization; mixture models

Introduction

Research demonstrating that both infants and adults track statistical distributions of acoustic-phonetic cues and use this information when making phonetic category judgements (Maye, Werker, & Gerken, 2002; Clayards, Tanenhaus, Aslin, & Jacobs, 2008) has led to interest in computational models of phonetic category acquisition, which can shed light on the requirements and limitations of statistical learning. The results of a number of studies (Vallabha, McClelland, Pons, Werker, & Amano, 2007; Toscano & McMurray, 2008) have yielded encouraging results, indicating that Gaussian mixture (GMs) may be an appropriate means of representing phonetic category structure. However, these structures are not static; they can and do change over a speaker's lifetime, albeit in ways which are not yet fully understood. This work builds on previous research by embedding the GM approach in an agent-based framework to explore the ways in which phonetic category structure changes over time.

Sound change as phonetic cue restructuring

Speech sound categories (consonants and vowels) are not monolithic entities, but are instead signaled by a multitude of acoustic dimensions, called *cues*. Lisker (1978) cites 16 acoustic dimensions relevant for the perceptual distinction between voiced (e.g., [b]) and voiceless (e.g., [p]) obstruents in word-medial position in English, including duration of the preceding vowel, fundamental frequency (f_0) contour, and timing of voice onset (VOT). While many cues are truly independent, others, such as VOT and f_0 contour, are redundant: vowels following voiced obstruents have lower f_0 than vowels following voiceless obstruents; in addition, some cues contribute more information to the identity of a contrast than others. Accurate categorization of an utterance involves weighting these of these cues, a task which finds a natural analog in density estimation (Ashby & Alfonso-Reese, 1995) and closely related 'ideal observer' models of speech perception as optimal Bayesian inference (Clayards, 2008; Feldman, Griffiths, & Morgan, 2009).

The distribution of cues to a speech sound category are not static, however, and may shift and change over time. An oft-cited example is the idea that lexical tone – the use of pitch to distinguish between words, familiar from languages such as Mandarin Chinese or Thai – finds its origins in consonantally-induced pitch perturbations (Hombert, Ohala, & Ewan, 1979). On this account, the physiologically-based, consonantally-induced differences in vowel f_0 first become part of a perceptual cue distinguishing two types of consonants. If f_0 comes under speaker control, it may then be used to actively to enhance the perception of this contrast. More generally, when the primary cue to a contrast becomes uninformative, the contrast may still be maintained through increased attention to a secondary cue, a process termed *phonologization* (Hyman, 1976).

Table 1: Phonologization of f_0 in Seoul Korean.

manner		*1960s*	*2000s*	*gloss*
fortis	뿔	[ppul]	[púl]	'horn'
lenis	불	[pul]	[p^hùl]	'fire'
aspirated	풀	[p^hul]	[p^húl]	'grass'

Empirical support for such an account may be found in the phonologization of f_0 currently taking place in Seoul Korean (Kang & Guion, 2008). In this language, a three-way contrast between fortis, lenis, and aspirated word-initial voiceless obstruents once distinguished chiefly by differences in VOT is now distinguished chiefly by differences in f_0. As shown in Table 1, fortis and aspirated stops are both produced with high f_0, but distinguished along the VOT dimension, whereas lenis stops are distinguished from aspirated by low f_0.

As it happens, VOT and f_0 are not the only cues relevant for the perception of word-initial obstruents in Seoul Korean: a number of studies (reviewed in Kang and Guion (2008)) have shown that other acoustic characteristics, such as length of the following vowel and spectral tilt at vowel onset, are also important cues to obstruent category. If phonetic categories are signaled by a multiplicity of cues, however, it is not immediately obvious why should f_0, and not some other cue, should have been phonologized, nor why this change took place in Korean, but not in other languages which displays a similar redundancy between VOT and f_0, such as English.

In this work, I propose that this type of phonetic category restructuring is the result of an adaptive strategy of cue enhancement designed to ensure robust communication in noise. Speakers enhance phonetic cue dimensions probabilistically, in proportion to their contribution to the successful perception and categorization of a phonetic contrast, based on the *informativeness* of a cue and the *precision* with which the contrast may be recovered. This predicts that phonetic cue restructuring will result from the loss of contrast precision due to noise or external bias, with the degree of enhancement proportional to the loss of precision.

Modeling phonetic cue restructuring

A series of agent-based simulations were conducted to better understand the effects of probabilistic enhancement on cue weights. Five cue dimensions known to be relevant for the perception of Korean stops (VOT, vowel length, closure duration, spectral tilt, and f_0) were represented as a set D of three 5-dimensional GMs, corresponding to the three word-initial obstruent categories. Both the initial and target parameters of each GM were estimated from data in the apparent time study of Kang and Guion (2008), represented as an exemplar list $E_k = \{e_1^k, \ldots, e_n^k\}$, e_i^k a 5-dimensional column vector of cue values plus a category label k and a decay weight τ.

Agent-based simulations

The simulations reported here consist of simple 'telephone' conversations in which two agents alternate between producing and categorizing utterances. At each iteration, the speaker agent selects a phonetic category target k, computes maximum likelihood estimates of the parameters $\mu_{d|k}, \sigma_{d|k}$ for all $d \in D$ based on E_k, and samples from each conditional density $x_d \sim \mathcal{N}(d|k; \mu_d, \sigma_d)$ to generate an utterance vector $\mathbf{x}$. The agent then enhances cue dimension d of $\mathbf{x}$ with some probability, proportional to both (i) the cue's *weight* (based on normalized d') and (ii) the current contrast *precision*, defined as the error rate of a naive Bayes classifier. Finally, $\mathbf{x}$ may be further modified by a transmission bias term λ, used to implement systematic biases such as articulatory drift.

The utterance $\mathbf{x}$ is then presented to the listener agent for classification. The listener agent assigns a category label k with probability $P(k|x_1, \ldots, x_D)$, where the posterior probability of each category k is calculated as

$$P(k|x_1, \ldots, x_D) = \frac{p(x_1|k)p(x_2|k), \ldots, p(x_D|k)p(k)}{\sum_{i=1}^{K} p(x_1|k_i)p(x_2|k_i), \ldots, p(x_D|k_i)p(k_i)}. \quad (1)$$

After classification, the agent adds $\mathbf{x}$ to the top of the appropriate exemplar list E_k, re-computes decay weights, and deletes exemplars with sufficiently low τ (to simulate memory decay). In the next iteration, when the listener agent becomes the speaker, the contribution of this newly categorized exemplar will be reflected in production when the agent computes new maximum likelihood parameter estimates.

Results

Simulations of up to 50,000 iterations were conducted with and without enhancement and for various settings of the bias term λ. Neither the proposed probabilistic enhancement strategy nor systematic bias alone were sufficient to induce a shift in cue weights that resembled the empirical target distributions, but simultaneous application of both gave a close approximation of the attested distributions and cue weights, as measured by the Kullback-Leibler divergence between the simulated results and the empirical targets. A second series of simulations in which the weights of secondary cues to the contrast were equalized at initialization, systematic bias in the production of VOT (the primary cue) led to either partial or total category merger or stability of the existing cue structure, depending on exact nature of the transmission bias.

Conclusions

Sound change resulting from a cognitive restructuring of phonetic cue weights may be modeled as an adaptive strategy of probabilistic enhancement interacting with systematic biases in speech production. Computational simulations show that such a restructuring may come about without appealing to either (a) inherent perceptual bias for or against any particular cues or (b) a system-wide pressure or preference for contrast maintenance. Given just the initial state and characterization of transmission bias, this model allows us to make (probabilistic) predictions about directionality in sound change. Ongoing extensions of this work include experimental testing of the model predictions using human subjects.

References

Ashby, F. G., & Alfonso-Reese, L. A. (1995). Categorization as probability density estimation. *Journal of Mathematical Psychology*, *39*, 216–233.

Clayards, M. (2008). *The ideal listener: making optimal use of acoustic-phonetic cues for word recognition*. Unpublished doctoral dissertation, University of Rochester.

Clayards, M., Tanenhaus, M. K., Aslin, R., & Jacobs, R. A. (2008). Perception of speech reflects optimal use of probabilistic speech cues. *Cognition*, *108*, 804–809.

Feldman, N. H., Griffiths, T. L., & Morgan, J. L. (2009). The influence of categories on perception: Explaining the perceptual magnet effect as optimal statistical inference. *Psychological Review*, *116*, 752–782.

Hombert, J.-M., Ohala, J. J., & Ewan, W. G. (1979). Phonetic explanations for the development of tones. *Language*, *55*(1), 37-58.

Hyman, L. (1976). Phonologization. In A. Juilland (Ed.), *Linguistic studies presented to Joseph H. Greenberg.* Saratoga: Anma Libri.

Kang, K.-H., & Guion, S. G. (2008). Clear speech production of Korean stops: Changing phonetic targets and enhancement strategies. *Journal of the Acoustical Society of America*, *124*(6), 3909–3917.

Lisker, L. (1978). Rapid vs. rabid: a catalogue of acoustic features that may cue the distinction. *Haskins Laboratories Status Report on Speech Research SR-54*, 128–32.

Maye, J., Werker, J. F., & Gerken, L. (2002). Infant sensitivity to distributional information can affect phonetic discrimination. *Cognition*, *82*(3), B101–B111.

Toscano, J., & McMurray, B. (2008). Using the distributional statistics of speech sounds for weighting and integrating acoustic cues. In *Proceedings of the Cognitive Science Society.* Mahwah, NJ: Erlbaum.

Vallabha, G. K., McClelland, J. L., Pons, F., Werker, J. F., & Amano, S. (2007). Unsupervised learning of vowel categories from infant-directed speech. *Proceedings of the National Academy of Sciences*, *104*(33), 13273–13278.

Canonical Behavior Patterns

Walter C. Mankowski (walt@cs.drexel.edu)
Department of Computer Science, Drexel University, 3141 Chestnut Street
Philadelphia, PA 19104 USA

Keywords: Protocol analysis; sequential data analysis

Introduction

In the development of cognitive models, data are often collected in the form of behavioral protocols — sequences of actions performed by the user during the execution of a task. Behavioral protocols have been employed to study a wide variety of actions, including mouse clicks and keystrokes (e.g., Card, Newell, & Moran, 1983), eye movements (e.g., Byrne et al., 1999), and driving (e.g., Salvucci, 2006). While protocols are a rich source of data, they have one significant limitation — often so much data are recorded that it is impractical to analyze by hand. Researchers have sometimes tried to get around this issue by performing some form of aggregation on their data. While this can help in seeing overall behavior, it masks potentially interesting patterns in individual users and subsets of users. Alternatively, researchers have sometimes laboriously studied individual protocols by hand to identify interesting behaviors. While some work has been done on automated protocol analysis, such techniques typically focus on matching observed behaviors to the predictions of some type of user process model.

The goal of my dissertation is to develop a new, automated method of protocol analysis to find canonical behaviors — a small subset of behavioral protocols that is most representative of the full data set, providing a reasonable high-level view of the data with as few elements as possible. The method I am proposing takes advantage of recent algorithmic developments in computational vision, and the method has already been successfully employed in diverse fields such as image analysis and software engineering. By adapting this algorithm to the analysis of behavioral protocols, I hope to provide a new tool for cognitive modelers working with large protocol data sets that are infeasible to study using current methods. My method can also be used as an important complement to existing protocol analysis techniques, allowing researchers to build their models based on a few highly representative samples.

Finding Canonical Behaviors

My technique for computing canonical behaviors derives from work in the area of computational vision, where techniques have been developed to identify canonical members of a class of visual patterns (Denton et al., 2008). The goal is to reduce a large set of patterns (in this context, behavioral protocols) to a smaller (often much smaller) subset of patterns that is most representative of the entire data set. Specifically, I define a canonical set of behaviors as a subset of protocols such that behaviors within the canonical set are minimally similar to each other, and behaviors in the canonical set are maximally similar to behaviors not in the set. The problem of finding such a set of patterns is known to be intractable (Garey & Johnson, 1979), and thus an approximation algorithm is utilized. Please refer to Denton et al. (2008) for a full description of the algorithm.

The key aspects of the method I propose are the specification of a similarity measure between behaviors and the determination of canonical behaviors given this similarity measure. The similarity measure is dependent upon the nature of the particular protocol. For web browsing, it might be the edit distance between two sequences of URLs (i.e., the number of insertions, deletions, or substitutions needed to transform one sequence to the other). For eye-tracking data, an appropriate measure might compare the x, y coordinates of the fixations, the number of fixated items, or the exact sequence of items fixated upon. Similarly, the determination of canonical behaviors is also dependent upon the context. For example, canonical behaviors for going to the next page in a word processor might include "press the page down key on the keyboard" and "click the scroll bar".

The principal benefits of my canonical set technique are (1) it is an unsupervised algorithm: no training data set is needed; and (2) no a priori knowledge of the number of sets is needed: both the sets themselves and the most representative elements of the sets arise naturally from the algorithm.

Preliminary Work

To test the application of the canonical set algorithm to human behavior protocols, I have done initial experiments in two problem domains. In the domain of web browsing, I have identified canonical web browsing patterns. I have also found canonical lane changes in a driving experiment. I briefly summarize each of these experiments below.

Web Browsing

As an initial experiment to validate this automated method of finding canonical sets, my colleagues and I collected data from users performing typical web browsing tasks. Each subject was asked to answer 32 questions that could be found on a college web site. The questions covered a range of realistic topics such as finding information about professors, athletic programs, and academic departments. (Please see Mankowski et al. (2009) for a full description of this experiment.)

Figure 1 shows the various behaviors for a single question ("What is the phone number of ⟨*department*⟩ professor ⟨*name*⟩") for (a) an expert human coder with significant experience in analyzing behaviors and cognitive modeling, and

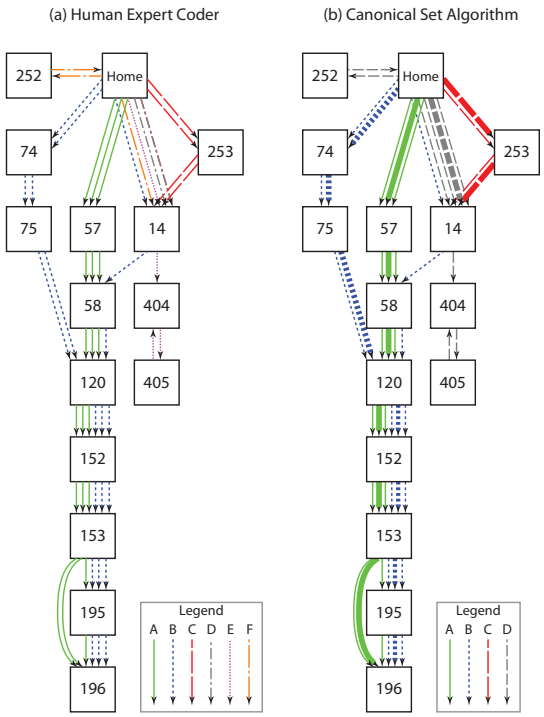


Figure 1: Sample analysis graphs.

(b) the canonical behavior algorithm. In both graphs, each node represents a web page (labeled with a unique integer) and each edge represents a clicked link from one page to another taken by one of our subjects. The expert coder found 6 sets of behaviors, labeled A-F: A and B are different ways of finding the professor's home page via their department's website; C and D are different ways of accessing a directory search page (node 14); and E and F are slight variations on C and D. The canonical set algorithm found 4 canonical behaviors in the same graph; these are shown as bold in graph b, and the other behaviors are labeled in terms of their most similar canonical behavior. The behaviors it found correspond exactly with sets A-D found by the expert coder. Instead of identifying E and F as separate behaviors, the algorithm decided to group these behaviors with their nearest canonical behavior (D).

Grouping behavior patterns is clearly a subjective process, since expert coders could each have their own notions about whether two behaviors are similar enough to be grouped together. For example, our second coder put behaviors A and B into the same group. To model this, the algorithm can be tuned to be more tolerant of differences in a grouping, or to allow more significant variations to become canonical elements themselves.

Driving

I have also applied the canonical set algorithm to the domain of driving, specifically the problem of identifying canonical lane changes (Mankowski, Shokoufandeh, & Salvucci, in press). Our data came from a previous experiment examining

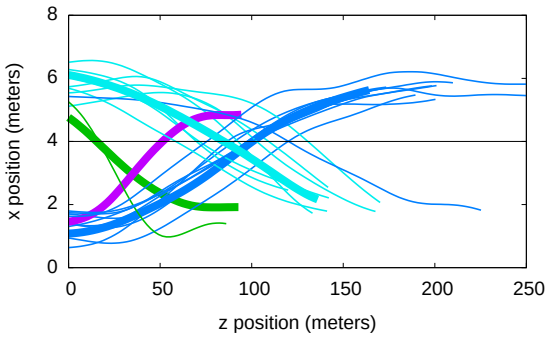


Figure 2: Lane changes for a representative subject.

driving behavior (Salvucci, 2006), where subjects navigated a simulated straight, flat highway and were required to pass a number of automated vehicles. For each lane change we constructed a histogram of the car's position and lateral velocity, and computed the similarity between each histogram.

Figure 2 shows the lane changes our algorithm found for a typical subject. Results were similar for the other subjects with these settings. The canonical lane changes are shown in bold, and the other lane changes are drawn in the same color as their most similar canonical lane change.

Acknowledgments

This work was supported in part by ONR grants #N00014-03-1-0036, #N00014-08-1-0925 and #N00014-09-1-0096, and by NSF grant #IIS-0426674.

References

Byrne, M. D., Anderson, J. R., Douglass, S., & Matessa, M. (1999). Eye tracking the visual search of click-down menus. In *Proc. CHI 1999* (pp. 402–409). New York: ACM Press.

Card, S. K., Newell, A., & Moran, T. P. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates.

Denton, T., Shokoufandeh, A., Novatnack, J., & Nishino, K. (2008). Canonical subsets of image features. *Computer Vision and Image Understanding*, *112*(1), 55–66.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractibility: A guide to the theory of NP-completeness*. San Francisco: W.H. Freeman and Co.

Mankowski, W. C., Bogunovich, P., Shokoufandeh, A., & Salvucci, D. D. (2009). Finding canonical behaviors in user protocols. In *Proc. CHI 2009* (pp. 1323–1326). New York: ACM Press.

Mankowski, W. C., Shokoufandeh, A., & Salvucci, D. D. (in press). Canonical patterns of oriented topologies. In *Proc. ICPR 2010.* IEEE Computer Society Press.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, *48*(2), 362–380.

Modeling Memes, A Memetic View of Affordance Learning

Benjamin D. Nye (benjamid@seas.upenn.edu)
School of Engineering and Applied Science
Department of Electrical and Systems Engineering
3320 Smith Walk
Hayden Hall, Room 120B, ACASA Lab
Philadelphia, PA 19104

Keywords: Memes; Information Theory; Social Psychology; Social Learning Theory; Systems Theory; Affordance Theory; Agent Based Modeling; Cognitive Agents

Overview

The purpose of this doctoral research is to apply a systems approach to defining and understanding memes. A meme is a piece of social information which transmits and replicates within a society (Heylighen, 1998). Memetics allow insight into the evolution of ideas and behavior, a fundamental question pertinent to all fields of social science. Three mechanisms guide the evolution of memes: reproduction, variation, and selection (Dennett, 1995). These mechanisms have to be understood in terms of empirical research on individual, social, and environmental factors that influence transmission and change of ideas. However, the body of relevant empirical study and theory is vast. This raises the basic research question: What synthesis of theories usefully explains meme behavior? The thesis of this research addresses this question using a three step process:

1. Synthesis of Theories - A conceptual model is synthesized which connects social science research to the mechanisms guiding meme transmission and evolution.
2. Computational Model - A cognitive agent simulation model is coded which operationalizes insights and theory captured by the conceptual model.
3. Testing the Model - Experiments conducted using the model examine the validity, flexibility, and types of insight the computational model provides.

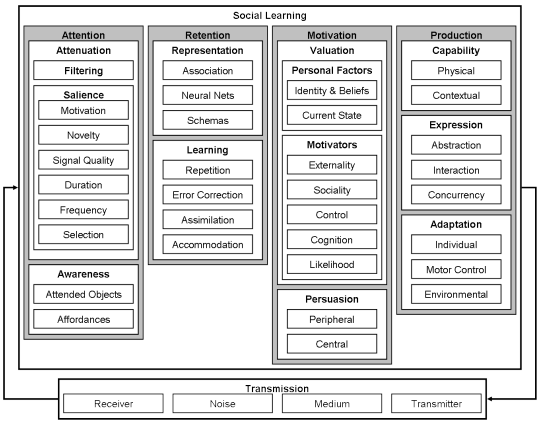
Hypotheses

This process is being used to test three hypotheses. The first hypothesis is that the theoretical relationships used to build the computational model will be statistically significant in the data collected from simulation runs. This hypothesis is intended to show internal validity. For example, the halo effect states a positive correlation of the likeability of a source with their persuasiveness. For this to hold true, a meme should be more likely to be repeated when received from a likable source. Relationships will be reported from the collected data by running logistic regression and statistical tests are being used to test significance.

The second hypothesis is that this model will provide a effective framework for representing and analyzing individual and situational characteristics that influence meme fitness. This hypothesis will be tested by applying classification techniques to detecting agents that receptive or resistant to different memes. The differences between classes will be examined statistically. These classifications will be compared against a human analysis of the scenario, as part of basic Turing test.

The third hypothesis is that memes can improve its correspondence with empirically collected behavioral data. This test involves building a scenario based on empirical data by tuning the scenario based upon personality factors and behavioral frequencies. The behavior of most interest is the first time an agent takes an action that express a meme. The order that agents first express memes can then be statistically compared against the real world observed ordering. The independent variable in this hypothesis is the set of agents who are initially aware of the meme. For this hypothesis to hold true, the trials with meme transmission must match the real-world ordering better than the trials where no social learning occurs (due to agents starting with full information). This tests if the propagation pattern improves the match of behavior to the ground truth.

Figure 1: Conceptual Model for Meme Transmission

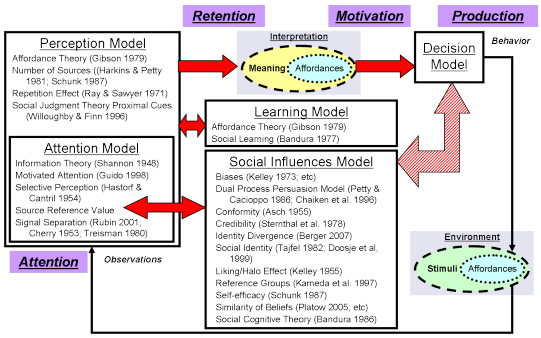


Experimental Design

As part of the thesis proposal, Shannon Information Theory and Bandura Social Learning Theory were synthesized to form an end-to-end model for retransmission of memes (Bandura, 1986; Shannon, 1948). Information theory considers the effects of noise and environmental influences on a physical transmission. Social learning theory considers the

processes that affect an agent's likelihood of repeating a transmission to which it is physically exposed. Social learning has a concept of observational learning that consists of 4 steps: attention, retention, motivation, and production. The synthesis of information theory and social learning theory provides a conceptual framework for connecting empirical findings together into a single process. Findings from perception, social psychology, learning, marketing, and other fields have been connected within this conceptual framework as noted in Figure 1. The thesis proposal for this project provides a significant mapping of theories into this framework, along with their implications for the mechanisms of meme evolution.

Figure 2: PMFServ Implementation of Meme Transmission



Based upon this synthesis of literature into a unified model, a simulation consisting of cognitive agents has been built using the PMFServ socio-cognitive agent architecture (Silverman, 2004). Figure 2 shows the key elements of the conceptual model that implemented as cognitive components for PMFServ agents. These agents are simulated within a shared environment, with meme transmission occurring when agents learn about affordances from each other's behavior. This implementation will concentrates on the cognitive factors that affect memes. The agents used within this model consider not only the intrinsic information of a meme, but also the appeal of the source, and the influence of the environment. Computational models for social influence, attention, and learning have been implemented according to empirically based findings and theory. The halo effect (Kelley, 1955), selective attention (Simons & Chabris, 1999), and conformity (Asch, 1963) are examples of over a dozen constituent theories used to build cognitive components.

The computational model is being used to simulate two scenarios: a reproduction of the Stanford Prison Experiment and an archetypal Iraqi village of Hamariyah based on US Marine Corps human terrain data. The Stanford Prison experiment is an infamous landmark field study in which seemingly normal participants were assigned roles as guards or prisoners in a simulated prison (Haney, Banks, & Zimbardo, 1973). The Stanford Prison experiment scenario has been calibrated and tested using de-identified Comrey Personality inventories and hourly coded behavioral logs. The potential memes in the Stanford Prison Experiment are the practice of throwing prisoners in "the hole" and the spread of prisoner resistance. All three hypotheses will be examined using the Stanford Prison Experiment simulation.

The Iraqi village scenario is being used to examine a pair of competing memes, one for informing to the US group and one for helping to plant an IED. Since there is no ground-truth data, only the first two hypotheses can be examined. However, the Iraqi village will be better suited to classification due to its larger number of agents and actions.

Contribution

The main goal of this research topic is to present a useful conceptual model for the transmission of memes, accompanied by a working and useful implementation. The theoretical contribution of the work has been to synthesize established models to help explain meme dynamics. It has also identified gaps in social science literature where the interaction of different theories is not well understood. The cognitive architecture implementation provides insight into the conceptual model's value for operationalizing and analyzing memes. The end result should help advance the capabilities of simulated societies to analyze real societies.

Acknowledgments

Many thanks to my advisor Dr. Barry Silverman, as well as my esteemed committee members Dr. Tony Smith, Dr. Kathleen Carley, and Dr. Joseph Bordogna.

References

Asch, S. E. (1963). Opinions and Social Pressure. *People and Productivity*.

Bandura, A. (1986). *Social Foundations of Thought and Action. A Social Cognitive Theory*. Englewood Cliffs, NJ: Prentice-Hall.

Dennett, D. C. (1995). *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Simon & Schuster.

Haney, C., Banks, W. C., & Zimbardo, P. G. (1973). Interpersonal Dynamics in a Simulated Prison. *International Journal of Criminology and Penology*, *1*, 69–97.

Heylighen, F. (1998). What makes a meme successful? Selection criteria for cultural evolution. In *16th international congress on cybernetics* (pp. 418–423). Namur, Belgium: Association internationale de cybernétique.

Kelley, G. A. (1955). The psychology of personal constructs (Vols. 1 & 2).

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Key Papers in the Development of Information Theory*. Available from `http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf`

Silverman, B. G. (2004). Toward Realism in Human Performance Simulation. In J. W. Ness, V. Tepe, & D. R. Ritzer (Eds.), (Vol. 5, pp. 469–498). JAI Press.

Simons, D. J., & Chabris, C. F. (1999). Gorillas in our midst: sustained inattentional blindness for dynamic events. *Perception*, *28*, 1059–1074.

Exploring a Novel Training Paradigm for Knowledge and Skills Acquisition

Jaehyon Paik (jaehyon.paik@psu.edu)
Department of Industrial and Manufacturing Engineering
Pennsylvania State University, University Park, PA

Keywords: Spacing effect, Distributed practice schedule, Massed practice schedule, Hybrid Practice schedule, Procedural stage

Introduction

Due to the importance of training, many scientists have studied effective training schedules, and they have compared distributed practice schedules to massed practice schedules. Most of the results consistently show that a distributed practice schedule outperforms a massed practice schedule in a retention test, because of the spacing effect in human memory.

This result may lead new scientists who want to investigate knowledge and skills acquisition to explore just these two schedules and compare them in retention tests without examining other options. However, I think that we need to consider and approach another way for knowledge and skills acquisition. According to Anderson's (1982, 1993) ACT-R theory, skill acquisition is the process of transition from declarative memory to procedural memory, and in the fully procedural stage, human beings do not need to retrieve their declarative memory to implement a task, even if they forgot the knowledge in declarative memory, they can perform the whole task without any errors. Based on his findings and theory, the most important factor in learning is how to transform learned knowledge to the procedural stage of the learning framework, and the research for learning should not compare two relatively extreme schedules, but make an appropriate schedule that could transfer learners to the procedural stage for each piece of knowledge. In this paper, I will present a candidate approach to make an appropriate schedule for getting better performance in knowledge and skill retention, and as a doctoral consortium paper, I hope I have useful advices for theoretical approach of the ACT-R cognitive architecture in this topic.

Theory

Spacing effects exists in human memory. This explains the reason that a distributed practice schedule has better performance than a massed practice schedule in retention tests. However, I mentioned in previous section, research for training should be focused on how to transfer a learned skill to the procedural stage.

Unfortunately, we do not have any measurement whether learners are in the procedural stage or not. One of the candidate measurements could be differences between the performances of the last training session and the retention session, however, it is difficult to fix the amount of differences that could represent the procedural stage. So, I think that we should consider how to increase performance in retention, and it may the only way to approach for explaining the status of procedural stage.

Pavlik (2005) studied practice and forgetting effects on vocabulary. In this research, he found that the spacing effects could be increased through distributed practice with massed practice; in other words, a mixed schedule could produce better performance than distributed schedule in vocabulary memory task.

I also think some kinds of tasks, such as procedural or perceptual-motor tasks, may show even better performance through an initial or distributed massed practice schedule. For example, we can learn how to ski perhaps better not in a distributed way (1 hour per day over 5 days), but in massed way (5 hours in a row in one day).

From the above results, I argue that a hybrid practice schedule that is a mixed schedule including distributed and massed practice, could increase the spacing effect, and generate better performance than a purely distributed or massed practice schedules on the retention test.

Methodology

To explore the better schedule on retention test, four kinds of experiment environment were developed. These tasks are presented in Table 1.

Table 1: Four tasks with respect to knowledge type.

Knowledge Type	Task
Declarative Memory	Japanese Vocabulary
Procedural Memory	Tower of Hanoi
Procedural to Declarative	Permutation Problem
Perceptual-Motor	BalanceMe® Game

The Japanese Vocabulary test that is similar to the task of Pavlik and Anderson (2003, 2005), is a web-based test, and participants will be tested with 15 Japanese vocabulary words. The accuracy (the number of correct answers) and RT (the completion time per correct answer) will be measured.

There are two kinds of tasks in procedural memory type. One is the Tower of Hanoi puzzle, and the other is solving permutation problem. A Tower of Hanoi game that will be modified from its original style has 3 rods with 6 disks. Participants will be asked to move 6 disks from the leftmost rod to the rightmost rod. The number of movements and the duration time will be measured.

For the permutation problem test, I will use the task of Rohrer and Taylor (2006). Participants will be taught to solve the number of unique ordering (or permutations) of a letter sequence with at least one repeated letter, such as

aabbbb, aabbbcc, etc.;, 12 problems will be presented. The accuracy (the number of correct answers) and RT (the completion time per correct answer) will be measured.

The BalanceMe® game, an Iphone® Application, will be used for the perceptual-motor task. This is very similar to an inverted pendulum. Participants will be asked to keep balancing the stick in the screen by tilting the device. The time duration of balancing will be measured.

Participants will be divided into 3 groups, massed group, distributed group, and hybrid practice group, and they will be asked to perform 8 sessions for training and 1 session to test retention. Each session consists of 4 tasks, and the order of tasks will be decided randomly.

Expected Results

As I mentioned in the Theory section of this paper, I assume that the participants of hybrid practice may show the best performance in all four tasks of retention test. The reason is that I believe the hybrid practice including the distributed practice and massed practice could increase spacing effect in human memory. I expect that the distributed practice schedule may outperform massed practice schedule in declarative memory task and the procedural to declarative task, because learners mainly depend on their declarative memory in these kinds of tasks. However, massed practice schedule may outperform distributed practice schedule in procedural memory task and perceptual-motor task, because massed practice may be needed in these kinds of tasks. These expected results are presented in Table 2. Figure 1 and figure 2 present expected learning curves of each of these practice schedules.

Table 2: The expected results for each task.

Schedules	Task
H > D > M	Japanese Vocabulary Permutation Problem
H > M > D	Tower of Hanoi BalanceMe® Game

Note: H means Hybrid, D means Distributed, and M means Massed Practice.

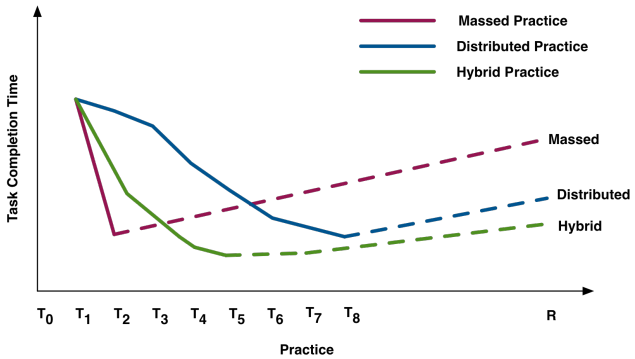


Figure 1: Expected results for declarative memory task and procedural to declarative task.

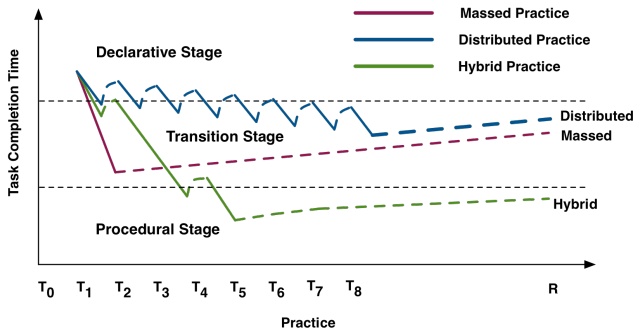


Figure 2: Expected results for procedural memory task and perceptual-motor task.

Conclusions

In this paper, I present a hybrid practice schedule that includes distributed and massed practice, and I assume that this is one of the candidate practice schedule for transferring learners to the procedural stage of learning framework. To explore this, I created four kinds of tasks that represent different knowledge types, declarative, procedural, and perceptual-motor.

At least 30 participants, 10 for each schedule, will be recruited by 31 July 2010, and I will explore the candidate hybrid schedule based on the ACT-R theory. By comparing human data and the theory of ACT-R, I may verify or extend the theory. Finally, this experiment will show the learning curves with the same subject in different types of tasks.

Acknowledgements

This work was supported by the U.S. Office of Naval Research (ONR) under contract N00014-06-1-0164 and the Defense Threat Reduction Agency under contract 1-09-1-0054.

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*(4), 369-406.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Pavlik, P. I., & Anderson, J. R. (2003). An ACT-R model of the spacing effect. In (pp. 177-182).

Pavlik, P. I., & Anderson, J. R. (2005). Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science, 29*(4), 559-586.

Rohrer, D., & Taylor, K. (2006). The effects of overlearning and distributed practise on the retention of mathematics knowledge. *Applied Cognitive Psychology, 20*(9), 1209-1224.

Modeling of Modality Selection in Multimodal Human-Computer Interaction

Stefan Schaffer (stefan.schaffer@zmms.tu-berlin.de)
Research Training Group prometei, Berlin Institute of Technology, Germany

Keywords: strategy selection; multimodal user behavior.

Research Interests

The main interest of my research is the development of model-based methods for simulation and automated usability prediction of multimodal interfaces. In particular I want to investigate how modality choice of users can be predicted and simulated by a computational model that estimates the quality of multimodal interfaces. Therefore I am also interested in exploring rules and cognitive processes that impact users' modality selection in multimodal human-computer interaction.

Previous Multimodal HCI Related Work

During my master thesis at Deutsche Telekom Laboratories (T-Labs) I worked on the integration of a speech recognition module in so-called Attentive Displays. The Attentive Displays are an interactive wall-mounted information system for employee and room search in a smart office environment. Originally the system was controlled with a touch screen only. To enhance the input facility I embedded a speech interface. Thereby the system input was altered from unimodal to multimodal.

Several studies have shown multimodal interfaces to be more robust, efficient and flexible than unimodal systems (e.g. Oviatt, 2003). As a part of my master thesis a user study with 36 participants and six tasks was conducted to investigate the effect of multimodality on user behaviour and the perceived quality of the system. In contrast to the assumption that the multimodal system is judged best, the evaluation revealed the perceived quality of both the touch screen and the multimodal version of the system were rated equally. The distinct malfunction of the speech recognition module in the multimodal setup could be a reason for this result. While accomplishing the tasks with the multimodal system it could also be observed that users switched from speech to touch input, after experiencing repeated speech recognition errors. Otherwise speech was the preferred input modality for tasks that were solvable with less interaction steps via speech (Metze et al., 2009).

Current Research Work

Currently I am working towards my PhD, where I am developing user models for the simulation of interaction between users and multimodal dialogue systems. Thereby the modality choice of users has to be simulated in each interaction step. A literature research within HCI related topics exploring user behavior and our previous work show that miscellaneous factors like e.g. expertise (Kamm et al., 2008; Seebode, 2009), task and efficiency (Naumann, 2008) and task success (Wechsung et al., submitted) influence modality selection. According to these findings the Attentive Display user study indicates that efficiency of interaction and system errors affect user behavior (Metze et al., 2009). Usage of shortcuts via speech and modality switch after repeated malfunction of the speech recognition module was observed in the study. Users appear to prefer more efficient interaction strategies.

In the following two subsections I give a closer description of two of my current research projects.

Project 1: Modeling Efficiency-Guided Multimodal Strategy Selection

In order to build a model for selected Attentive Display tasks, human data about modality usage, recorded during the experiment, serves as a target value. Currently the employee search task including a shortcut via speech input is modeled with the cognitive architecture ACT-R (Anderson & Lebiere, 1998). To search the employee "Patrick" by means of touch screen or speech input the following interaction steps have to be fulfilled:

[pre 1] Search button "SEARCH" [post 2]
[pre 2] Press button "SEARCH" [post 3]
[pre 2] Speak "SEARCH PATRICK" [post 11]
[pre 3] Search button "P" [post 4]
[pre 4] Press button "P" [post 5]
[pre 5] Search button "A" [post 6]
[pre 6] Press button "A" [post 7]
[pre 7] Search "T" [post 8]
[pre 8] Press button "T" [post 9]
[pre 9] Search button "PATRICK" [post 10]
[pre 10] Press the button "PATRICK" [post 11]
[pre 11] Search goal cue [post end]

This simple task analysis is implemented in the ACT-R model as instructions in declarative memory. The model also provides a couple of production rules for retrieving the instructions, searching in the interface, pressing buttons and speaking commands. The structure of the model is similar to the model presented by Taatgen et al. (2006) where declarative instruction chunks are associated through pre- and postconditions. This makes it easy to reuse instructions which are used for speech and touch interaction (e.g. production [pre 1] and [pre 11]). Additionally this representation features a practical flexibility which can be used for simulating multimodal interaction with ACT-R. If two chunks with the same precondition are added to declarative memory, different interaction strategies can be retrieved and different postconditions can be set. Thereby it has to be taken into account that chunks where the

precondition occurs twice are chosen randomly. Hence the model does not reproduce human behavior. One possibility of solving this problem is to use the ACT-R inherent mechanisms production compilation (Taatgen & Lee, 2003) and utility learning. The production compilation mechanism combines two production rules into one new rule and substitutes retrievals from declarative memory directly into the new rule. Thus specialized productions for speech and touch interaction are created. Utility learning rewards all rules which are involved in reaching the goal. The total reward is a stated value and spreads over the involved rules. Consequently the reward per rule is lower if more rules were involved. By means of these mechanisms it should be possible to let ACT-R learn the utilities of new production rules during an initial training. After the training the strategy involving less production rules should have a higher utility. Hence the more efficient modality should be used with a higher probability.

The aim of this research project is to investigate if ACT-R could be applied directly as a decision mechanism for modality selection in a development environment (the MeMo Workbench), which is based on prior work of the T-Labs (Möller et al., 2006). Furthermore rules for modality selection will be derived.

Project 2: Efficiency-Dependent Thresholds for Modality-Changing

This research project aims to develop a multimodal prototype for purposes of investigating thresholds for modality changing. Users of multimodal systems often have the possibility to choose a specific input modality to perform an interaction step during the processing of a task. Diverse factors influencing modality choice including efficiency-related factors like time to solve the task, interaction steps and cognitive load have to be considered. The objective is to examine whether users change their input modality from touch to speech interaction or vice versa, if the modalities offer different efficiencies. Therefore I propose a task which systematically allows varying the number of interaction steps to solve a goal. Additionally cognitive load should be kept as constant as possible. The task will be integrated in an application on a mobile device.

The findings of this project should be translated into rules which will be used by the MeMo Workbench.

Future Work

In addition to the aforementioned projects further research is required on factors like cognitive load, dual task, experience, system errors and individual user attributes. A detailed factor model describing the effects and relations of the factors to each other should be deployed. Furthermore the developed models should be validated by transferring to other tasks.

My findings about modality selection will be integrated into the MeMo Workbench which so far only facilitates the evaluation of unimodal system models. After the extension MeMo will be validated again. Therefore systems and tasks which have been explored in prior experiments will be modeled with MeMo. The empirical data gathered during the experiments will serve as target values.

Acknowledgments

The research is funded by the German Research Foundation (DFG - 1013 'Prospective Design of Human-Technology Interaction') and supported by Deutsche Telekom Laboratories.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Erlbaum.

Kamm, C., Litman, D., & Walker, M (2008). From novice to expert: the effect of tutorials on user expertise with spoken dialogue systems. *Proccedings of the 9th Annual Conference of theInternational Speech Communication Association*.

Metze, F., Wechsung, I., Schaffer, S., Seebode, J., & Möller, S. (2009). Reliable Evaluation of Multimodal Dialogue Systems. In *Proceedings of the 13th international Conference on Human-Computer interaction.* Springer-Verlag, Berlin, Heidelberg, 75-83.

Möller, S., Englert, R., Engelbrecht, K., Hafner, V., Jameson, A., Oulasvirta, A., Raake, A., Reithinger, N. (2006). MeMo: Towards automatic usability eval-uation of spoken dialogue services by user error simulations. *In Proceedings of INTERSPEECH 2006*. Pittsburgh, PA.

Naumann, A. B., Wechsung, I., & Möller, S. (2008). Factors Influencing Modality Choice in Multimodal Applications. In *Proceedings of the 4th IEEE Tutorial and Research Workshop on Perception and interactive Technologies For Speech-Based Systems: Perception in Multimodal Dialogue Systems*. Springer-Verlag, Berlin, Heidelberg, 37-43.

Oviatt, S. (2003). Multimodal interfaces. In *the Human-Computer interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (pp. 286-304). Hillsdale, NJ: L. Erlbaum Associates.

Seebode, J., Schaffer, S., Wechsung, I. & Metze, F. (2009). Influence of Training on Direct and Indirect Measures for the Evaluation of Multimodal Systems. *10th Annual Conference of the International Speech Communication Association*, (Interspeech 2009).

Taatgen, N.A., Huss, D. & Anderson, J.R. (2006). How Cognitive Models can Inform the Design of Instructions. *Proceedings of the seventh international conference on cognitive modeling* (pp. 304-309). Trieste, Italy: Edizioni Goliardiche.

Taatgen, N.A. & Lee, F.J. (2003). Production Compilation: A simple mechanism to model Complex Skill Acquisition. *Human Factors*, 45(1), 61-76.

Wechsung I., Schaffer S., Schleicher R., Naumann A. & Möller S. (submitted). *The Influence of Expertise and Efficiency on Modality Selection Strategies and Perceived Mental Effort.*

Visual Search Strategies and the Layout of the Display

Bella Z. Veksler (zafrib@rpi.edu)
Cognitive Science Department, Rensselaer Polytechnic Institute
110 8th St., Troy, NY 12180

Keywords: ACT-R; visual search; memory; finsts; lockout

Introduction

The role of visual search in everyday tasks is paramount. Whether we are searching for an item in the grocery store, trying to find our car in a busy parking garage, or looking for an important piece of information on a web page, the visual search mechanism is crucial. We are also quite efficient at performing all of these tasks. The main focus of the current proposal will be to further our understanding and modeling of what makes the process so efficient. The key emphasis here is on the process of visual search – the actual strategies that people utilize as they search for things and the degree to which memory plays a role in aiding in this process.

Visual search as a paradigm has been studied meticulously for the better part of the last 50 years. The paradigm consists of the detection of a target among a varying number of distractors with the dependent measure being whether the search is serial or parallel (Duncan & Humphreys, 1989; Treisman & Gelade, 1980; Wolfe, 1994).

The role of memory within visual search has also been greatly debated. In some instances, researchers have inferred from response time data that memory is not utilized during search because there was not a difference in response time between static and dynamic search conditions (Horowitz & Wolfe, 2003; Korner & Gilchrist, 2007; Melcher & Kowler, 2001; Peterson, Beck, & Wong, 2008; Peterson, Kramer, Wang, Irwin, & McCarley, 2001). In other instances, it has been shown that visual search is guided by memory for previously viewed items (Korner & Gilchrist, 2007; Peterson et al., 2008; Peterson et al., 2001). In particular, eye movement provides a more detailed picture of the underlying search process (Geyer, von Muhlenen, & Muller, 2007). Geyer et. al. used the same search paradigm as Horowitz and Wolfe but analyzed the eye movement behavior in addition to the response time data and found that participants rarely re-fixate items suggesting a role for memory in visual search. Furthermore, path memory has also been shown to exist suggesting that more of the distractor space is represented (Dickinson & Zelinsky, 2007).

In all of these studies, however, it was specifically visual search that was being manipulated and measured. As such, these tasks have been relatively simple – presenting items on the screen for varying lengths of time and measuring how long it took for participants to find the target. In the current work, the visual search process will be analyzed and modeled embedded within the context of a larger task. In particular, I am interested in how the search process is modulated when people are forced to wait for information to appear (during a timed lockout) and by having searched for other items on the same display. In the course of attempting to model performance on this task (using ACT-R), it was found that the model had problems with the basic visual search process. It is therefore the goals of the current work to explore the visual search strategies employed by participants and implement them in the model, which will consequently aid in modeling the rest of the task.

The Task

A simple radar task was used to determine how people allocate attention when forced to wait for information to appear. As compared to traditional visual search tasks where each trial consists of a single target among varying numbers of distractors, this task had distractors that on another visual pass through the display could be targets. Therefore memory for previous distractors would be beneficial and may guide subsequent searches.

Procedure

Participants were eye-tracked while they completed 60 trials of the task. A radar screen (Figure 1) was displayed on the left and was comprised of a static display of 20 2-digit numbers arranged randomly on the display. On the right side of the screen was the table of alternatives (TOA).

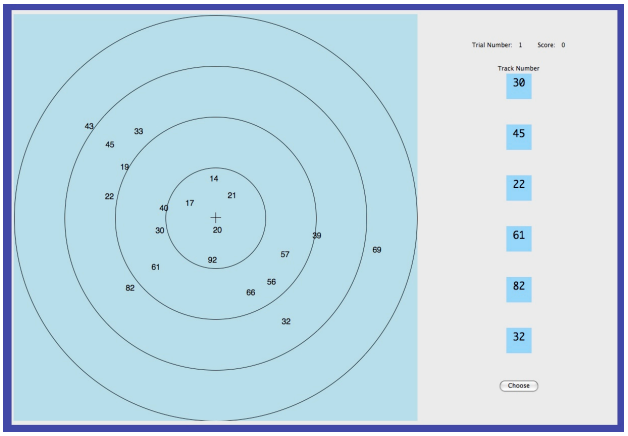


Figure 1: Task display, seen both by human participants and the model.

During the trial, the task of the participant was to determine which of the six targets from the TOA had the highest threat value. Threat values ranged from 0(lowest) to 9(highest). In order to discover the threat value of a particular alternative (target), the participant had to find and

click on the target in the radar display. Once a target was clicked (selected), there was a lockout delay of 1, 2, 4 or 8s depending on the participant's condition. The threat value would then appear next to the selected item. Consequently, the participant had to repeat this process with the rest of the items from the TOA until the highest threat-valued target was discovered.

Preliminary Analyses & Model

Preliminary analyses of the data were done with respect to the first several fixations on each trial to determine the search strategies used to find the first target clicked. A typical sequence of first fixations involved participants looking at 1-2 TOA items and then moving their gaze to the radar. Participants were able to find the first item they selected in an average of 6 fixations, with no differences between the four conditions. Participants also tended to re-fixate items on the radar display in ~15.37% of fixations prior to selecting the first target.

In order to inform the model's search initiation, I also looked at where participants tended to begin their search. There were several possibilities: a) closest to TOA, b) closest to center of the radar, c) closest to one of the corners of the radar, d) to the item that had the most other items around it (most 'clustered'), e) to the item that had the least other items around it (singleton). The results are beyond the scope of this paper, but they will be used to inform the model.

The ACT-R cognitive architecture will be used to model this task because of its ability to ground the model in the same environment that human participants saw (Anderson et al., 2004). The goal of the proposed work will be to use human data to inform the model's visual search process as in its current state it is considerably more inefficient than human participants in finding the targets in the radar. ACT-R currently uses the finst mechanism for ensuring that items previously fixated are not re-fixated within a given amount of time. However, although people find the item they are searching for efficiently (within 6 fixations), they also tend to revisit items they have viewed before suggesting that relying on the finst mechanism is insufficient to model behavior.

Future Work

Instead of relying on the finst mechanism, the proposed work will determine the degree to which the visual segmentation of the display allows for the efficient search process. Others have shown that fixations and saccades progress in a course-to-fine strategy whereby fixation durations increase while saccade amplitudes decrease as search continues (Over, Hooge, Vlaskamp, & Erkelens, 2007). The current work will explore whether people systematically search the displays such that they look within visual 'clusters' of items, thereby minimizing the number of areas they need to search to find the targets.

Currently, a k-means clustering algorithm has been used to quantitatively assess which items appear to cluster together on each screen. However, k-means has the limitation that it is difficult to know what value of k is appropriate for each screen layout. Therefore, a new study is being run which presents the same screen layouts the original participants saw to naïve participants who are asked to make these judgments.

The modeling work will take into account the findings from this new study and will incorporate the visual search strategies employed both at the beginning of each trial, during subsequent searches, and during lockouts.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An Integrated Theory of the Mind. *Psychological Review, 111*(4), 1036-1060.

Dickinson, C. A., & Zelinsky, G. J. (2007). Memory for the search path: Evidence for a high-capacity representation of search history. *Vision Research, 47*(13), 1745-1755.

Duncan, J., & Humphreys, G. W. (1989). Visual-Search and Stimulus Similarity. *Psychological Review, 96*(3), 433-458.

Geyer, T., von Muhlenen, A., & Muller, H. J. (2007). What do eye movements reveal about the role of memory in visual search? *Quarterly Journal of Experimental Psychology, 60*(7), 924-935.

Horowitz, T. S., & Wolfe, J. M. (2003). Memory for rejected distractors in visual search? *Visual Cognition, 10*(3), 257-298.

Korner, C., & Gilchrist, I. D. (2007). Finding a new target in an old display: Evidence for a memory recency effect in visual search. *Psychonomic Bulletin & Review, 14*(5), 846-851.

Melcher, D., & Kowler, E. (2001). Visual scene memory and the guidance of saccadic eye movements. *Vision Research, 41*(25-26), 3597-3611.

Over, E. A. B., Hooge, I. T. C., Vlaskamp, B. N. S., & Erkelens, C. J. (2007). Coarse-to-fine eye movement strategy in visual search. *Vision Research, 47*(17), 2272-2280.

Peterson, M. S., Beck, M. R., & Wong, J. H. (2008). Were you paying attention to where you looked? The role of executive working memory in visual search. *Psychonomic Bulletin & Review, 15*(2), 372-377.

Peterson, M. S., Kramer, A. F., Wang, R. X. F., Irwin, D. E., & McCarley, J. S. (2001). Visual search has memory. *Psychological Science, 12*(4), 287-292.

Treisman, A. M., & Gelade, G. (1980). Feature-Integration Theory of Attention. *Cognitive Psychology, 12*(1), 97-136.

Wolfe, J. M. (1994). Guided Search 2.0 - a Revised Model of Visual-Search. *Psychonomic Bulletin & Review, 1*(2), 202-238.

Cognitive Control: A Symposium

Andrew Howes (Andrew.Howes@mbs.ac.uk)
Manchester Business School
University of Manchester
Booth Street West, Manchester, M15 6PB, UK

Richard P. Cooper (R.Cooper@bbk.ac.uk)
Department of Psychological Science,
Birkbeck, University of London
Malet Street, London, WC1E 7HX, UK

Abstract

Cognitive control may be defined as the mechanisms or processes invoked in order to engage in goal directed behaviour under system constraints. This symposium explores a range of recent computational approaches to understanding problems of cognitive control. It comprises five presentations which each discuss a different aspect of cognitive control and a discussion session.

Keywords: Cognitive control; executive function; rational adaptation; task switching; monitoring; multitasking.

Introduction

In cognitive science there is a substantial research tradition of studying control problems such as how the cognitive system ensures the selection of a desired action in circumstances where an automated, learned, action might otherwise be selected. Control problems are often understood as arising from the necessity to serialise the multi-threaded processing contributions of a parallel neural architecture, but some work (e.g. Rieskamp, 2008) has tried to extend the application of control metaphors, derived from control theory and reinforcement learning, to a broader range of phenomena, including those associated with higher level decision making tasks. In the most general terms we might define the control problem as the problem of what to do next. A view that, perhaps, encourages an integrative approach to cognition that eschews prior commitments to particular forms of processing mechanisms. Control is about engaging in goal directed behaviour under system constraints.

Questions concerning cognitive control include, for example: how people switch among the short-term goals that govern everyday behaviour (Altmann & Gray, 2008); how people allocate perceptual, motor and cognitive resources in the control of interactive behaviour (Gray et al., 2006); how people adjust architectural parameters in the light of feedback (Botvinick et al, 2001); how people inhibit prepotent but inappropriate or unintended behaviours (Norman & Shallice, 1986); how the cognitive system resolves the problem of producing multiple responses when processing or physical constraints prevent them from being produced in parallel (Howes et al., 2009); how the cognitive system may manage strategies in demanding memory tasks (Juvina & Taatgen, 2007); and how the cognitive system learns to prefer specific strategies in judgement and decision making tasks (Rieskamp, 2008). It is also critical to real world applied problems such as driving (Salvucci, 2006; Janssen and Brumby, in press; Gunzelmann et al., 2009).

The control problem is difficult for a number of reasons:

1. *The temporal credit assignment problem.* Control is adaptive, so the problem of control encompasses the problem of how to make use of feedback. However, multiple actions can contribute to feedback and feedback may be delayed. This raises the problem of which actions should be assigned credit/blame when feedback is received. (cf. Lovett and Anderson's (1996) utility learning within ACT-R).
2. *The uncertainty problem.* Frequently information that we do have (e.g., feedback) is uncertain. We may know that information is uncertain, but how should information about uncertainty be processed?
3. *The scaling problem.* When many choices are available considering them all is computationally expensive. Scaling problems are found in, for example, both reinforcement learning and Bayesian approaches to modelling control and inference (Botvinick, Niv & Barto, 2009).
4. *The bounds problem.* The brain is a physically instantiated neural processing mechanism that imposes limits on what information can be encoded and effectively deployed. Cognitive control involves making efficient use of the neural mechanism subject to these limits (Howes, Lewis & Vera, 2009).
5. *The concurrency problem.* In many situations behaviour is under the control of multiple goals which we work towards concurrently, as in the example of driving while navigating or holding a phone conversation (Salvucci & Taatgen, 2008).

The symposium will explore a range of recent computational approaches to understanding the control problem through five diverse presentations and a discussion session.

Botvinick's recent work emphasises the hierarchical structure of control knowledge: the divisibility of ongoing behavior into discrete tasks, which are comprised of subtask sequences, which in turn are built of simple actions. Botvinick, Niv and Barto (2009) reexamines behavioral hierarchy and its neural substrates from the point of view of recent developments in computational reinforcement learning. Specifically, a set of approaches known collectively as hierarchical reinforcement learning is considered. A close look at the components of hierarchical reinforcement learning suggests how they might map onto neural structures, in particular regions within the dorsolateral and orbital prefrontal cortex. A particularly important question that hierarchical reinforcement learning brings to the fore is that of how learning identifies new

action routines that are likely to provide useful building blocks in solving a wide range of future problems.

Cooper will discuss the potential roles of so-called forward and inverse models in cognitive control. Forward models are representations of a future state of a system given its current state and a plan or course of action, while inverse models "invert the causal flow" and allow one to predict, given a desired state and a course of action will should result in that state. Both forward and inverse models have been argued to play important roles in motor control (e.g., Wolpert & Ghahramani, 2000). Cooper will argue that such models, possibly learned through associative and reinforcement learning mechanisms, may equally play a significant role in cognitive control, allowing the cognitive system to predict appropriate processing parameters and thereby configure itself prior to task performance.

Both Howes and Lewis will explore computational and empirical approaches to understanding people as *bounded optimal* control systems (Howes, Lewis & Vera, 2009). They contend that through learning people solve the constrained optimisation problem presented by their architecture. Howes will present evidence concerning bounded optimal control of working memory strategies. The work demonstrates that people do not only adapt strategies to changes in the cost structure of the task environment but rather they adapt optimally. Lewis will present a boundedly optimal control perspective on interference resolution. He will report a computational model of how people adapt strategically to interference in memory.

Taatgen's task will aim to initiate a discussion about asking the right questions; clearly a precusor to the search for answers.The standard way to think about cognitive control in multitasking is that control is needed to schedule the use of resources between tasks (e.g., Kieras, 2007). A different view, prompted by the threaded cognition theory of multitasking (Salvucci & Taatgen, 2008), is that not all cognitive processing can be understood in terms of tasks. As soon as we consider something as a task, some measure of cognitive control is needed to make sure all the steps in the task are carried out to achieve the goal. Miyake's (Miyake et al, 2000) three categories of cognitive control (inhibition, working memory and task switching) are all needed to protect a task from interference, but they are not the whole story. Cognitive modeling can help complete the picture.

References

Altmann, E. M., & Gray, W. D. (2008). An integrated model of cognitive control in task switching. *Psychological Review*, *115*(3), 602-639.

Botvinick, M.M. Braver, T.S., Barch, D.M., Carter, C.S. & Cohen, J.D. (2001). Conflict monitoring and cognitive control. *Psychological Review*, *108*, 624–652.

Botvinick, M. M., Niv, Y., & Barto, A. C. (2009). Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, *113*(3), 262-280.

Brumby, D. Salvucci, D. & Howes, A. (2007). Dialing while driving? A bounded rational analysis of concurrent multi-task behaviour. In R. L. Lewis, T. A. Polk & J. E. Laird (eds.), *Proceedings of the 8th International Conference on Cognitive Modeling* (pp. 121-126). Ann Arbor, MI.

Gunzelmann, G., Moore, Jr, L.R., Salucci, D., & Gluck, K. (2009). Fluctations in alertness and sustained attention: Predicting driver performance. In A. Howes, D. Peebles & R. P. Cooper (eds), *Proceedings of the 9th International Conference on Cognitive Modeling* (pp. 44-49). Manchester, UK.

Gray, W. D., Sims, C. R., Fu, W., & Schoelles, M. J. (2006). The Soft Constraints Hypothesis: A Rational Analysis Approach to Resource Allocation for Interactive Behavior. *Psychological Review*, *113*(3), 461-482.

Howes, A., Lewis, R. L., & Vera, A. (2009). Rational adaptation under task and processing constraints: Implications for testing theories of cognition and action. *Psychological Review*, *116*(4), 717-751.

Janssen, C.P. & Brumby, D.P. (in press). Strategic adaptation to performance objectives in a dual task setting. Cognitive Science, in press.

Juvina, I. & Taatgen, N. A. (2007). Modeling control strategies in the B-back task. In R. L. Lewis, T. A. Polk & J. E. Laird (eds.), *Proceedings of the 8th International Conference on Cognitive Modeling* (pp. 121-126). Ann Arbor, MI.

Kieras, D. E. (2007). Control of cognition. In W. Gray (Ed.), *Integrated models of cognitive systems* (pp. 327-355). New York: Oxford University Press.

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem solving: Combined influences on operator selection. *Cognitive Psychology*, *31*(2), 168-217.

Miyake, A., Friedman, N. P., Emerson, M. J., Witzki, A. H., & However, A. (2000). The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: a latent variable analysis. *Cognitive Psychology, 41*, 49-100.

Norman, D.A. & Shallice, T. (1986). Attention to action: willed and automatic control of behaviour. In R. Davidson, G. Schwartz, and D. Shapiro (eds.) *Consciousness and Self Regulation, Volume 4*, pp. 1–18. Plenum: NY.

Reiskamp, J. (2008). The importance of learning when making inferences. *Judgment and Decision Making*, *3(3)*, 261-277.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, *48*(2), 362-380.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*, *115*(1), 101-130.

Wolpert, D. M., & Ghahramani, Z. (2000). Computational principles of movement neuroscience. *Nature Neuroscience*, *3*, 1212-1217.

Multi-Agent Activity Modeling with the Brahms Environment

Maarten Sierhuis (sierhuis@parc.com)
Palo Alto Research Center, 3333 Coyote Hill Road
Palo Alto, CA 94304 USA

Keywords: agent-based modeling, BDI, cognitive modeling,

Introduction

More and more people are interested in developing "day in the life" models and simulations of people's behavior at the second and longer timeframe, the interaction between groups of people and systems, as well as the movement and interaction within the environment. Cognitive modeling tools (e.g. SOAR, ACT-R) focus on detailed modeling of individual cognitive tasks at the sub-second level. In contrast, activity modeling focuses on higher-abstraction behaviors that enable modeling of people's daily activities and enable a focus on how informal, circumstantial, and located behaviors of a group of individuals occur and where communication and synchronization happen, such that the task contributions of people and machines flow together to accomplish goals. This is referred to as "work practice modeling."

Brahms includes an activity-oriented Belief-Desire-Intention (BDI) language, a compiler and virtual machine for executing Brahms models, as well as an Eclipse plug-in and a post-execution viewer of agent execution, communication and interaction. Brahms enables the creation of multi-agent models that include aspects of reasoning found in cognitive models, task execution, plus the impact of interaction and geography, such as agent movement and physical changes in the environment. Brahms is currently used to automate the work of a flight controller in NASA's International Space Station's Mission Control Center (ISS MCC). This system, called OCAMS, has been in production in the ISS MCC, 24x7, since July of 2008, and is based on a Brahms model of the work practices of the flight controllers. OCAMS is a distributed Multi-Agent System (Sierhuis et al., 2009b).

Motivation for Brahms

Brahms was developed as a multiagent modeling and simulation language to visualize the social systems of work for business redesign projects. The Brahms language was originally conceived of as a language for modeling contextual behavior of groups of people, called work practice.

> *Work Practice:* The collective performance of contextually situated activities of a group of people who coordinate, cooperate and collaborate while performing these activities synchronously or asynchronously, making use of knowledge previously gained through experiences in performing similar activities.

This created two very important ideas for the language: First, to model a group of people it is very natural to model them as software agents. Second, modeling situated behavior of a group imposes a constraint on the level of detail that is useful in modeling the dependent and independent behavior of the individuals. The right level is a representational level that falls between functional process models and individual cognitive models (Clancey et al., 1998). If we are interested in modeling a day-in-the-life of say ten or more people, modeling the individual behavior at the level of cognitive task models will be very time consuming, because these models are generally at the millisecond decision-making level. To overcome this kind of detail, the Brahms language uses a more abstract level of behavioral modeling that is derived from Activity Theory (Leont'ev, 1978; Vygotsky, 1978) and Situated Action (Suchman, 1987). An individual's behavior is represented in terms of activities that take an amount of discrete time and can be decomposed into more detailed subactivities if necessary.

Brahms demonstrates how a multiagent belief-desire-intention (BDI) language, symbolic cognitive modeling, traditional business process modeling, activity- and situated cognition theories are brought together in a coherent approach for analysis and design of organizations and human-centered systems.

The Brahms Language

The Brahms language is a pure AOL (Sierhuis et al., 2009a). It is not a set of Java libraries enabling agent-based programming in the Java language. Instead, Brahms is a full-fledged multiagent language allowing the modeler to easily and naturally represent *multiple agents*. Although Brahms was originally developed for modeling people's behavior, the Brahms language is a domain independent language. This means that the modeler decides what a Brahms model represents. Agents can represent whatever autonomous entity the modeler wants to represent, such as a person, an animal, or an autonomous or intelligent system. Brahms includes the following language features:

- *Mental attributes*: attributes, relations, beliefs and facts.
- *Deliberation and Intention*: concluding new beliefs, and use of production rules for reasoning.
- *Adaptation*: changing beliefs, execution activity behavior and reasoning based on context.
- *Social Abilities*: groups and group inheritance, communication, and modeling the environment (objects, geography and location).

- *Reactive and Cognitive-based behavior*: modeling activity behavior, versus pure cognitive behavior, detectables, workframe-activity subsumption.
- *Agent Communication*: communication activities, and communicative acts.

Brahms is an agent-oriented BDI-like language. It allows easy creation of groups of agents that execute parallel activities based on local beliefs. Below is a simple taxonomy of the language concepts:

GROUPS are composed of
 AGENTS having
  BELIEFS and doing
  ACTIVITIES executed by
   WORKFRAMES defined by
    PRECONDITIONS, matching agents beliefs
    PRIMITIVE ACTIVITIES
    COMPOSITE ACTIVITIES, decomposing the activity
    DETECTABLES, including INTERUPTS, IMPASSES
    CONSEQUENCES, creating new beliefs and/or facts
  DELIBERATION implemented with
   THOUGHTFRAMES defined by
    PRECONDITIONS, matching agents beliefs
    CONSEQUENCES, creating new beliefs

Cognitive Modeling in Brahms

Brahms borrows some of the theoretical underpinnings of the ACT-R theory about human knowledge. Just as in ACT-R, Brahms assumes there are two types of knowledge—declarative and procedural.

Declarative knowledge in Brahms is represented as "beliefs" of *individual agents*. A *belief* of an agent in Brahms plays a similar role in processing the procedural knowledge of the agent as chunks do in ACT-R, i.e. they are matched to preconditions of rules. However, beliefs are semantically and syntactically simpler than chunks. A belief is a first-order predicate statement.

Brahms represents the *procedural knowledge* of an *individual agent* as rule-like constructs called *workframes* and *thoughtframes*. The condition-part—called *pre-conditions*—are matched against the belief-set of the Brahms agent. When all the preconditions of a workframe or thoughtframe match, the rule is put onto the agent's work stack. Each of these rule-types is processed independently by the virtual machine. Hence the reason for separate stacks.

Thoughtframes are similar to production rules in ACT-R, in that their action-part consists *only* of changes to, or additions of beliefs in the agent's declarative memory, i.e. its belief-set. Thoughframes are executed in a forward-chaining mode. Workframes are the main type of rule in Brahms. Activities are executed in the action-part of a workframe (the workframe's *body*). The time it takes to fire a workframe depends on the total time it takes to execute all of the activities in its body. The workframe rule-type in Brahms corresponds to the goal-directed production rules in ACT-R, with the addition that in a workframe we can include the changes in the external world, and not only the internal declarative changes in the memory of the agents.

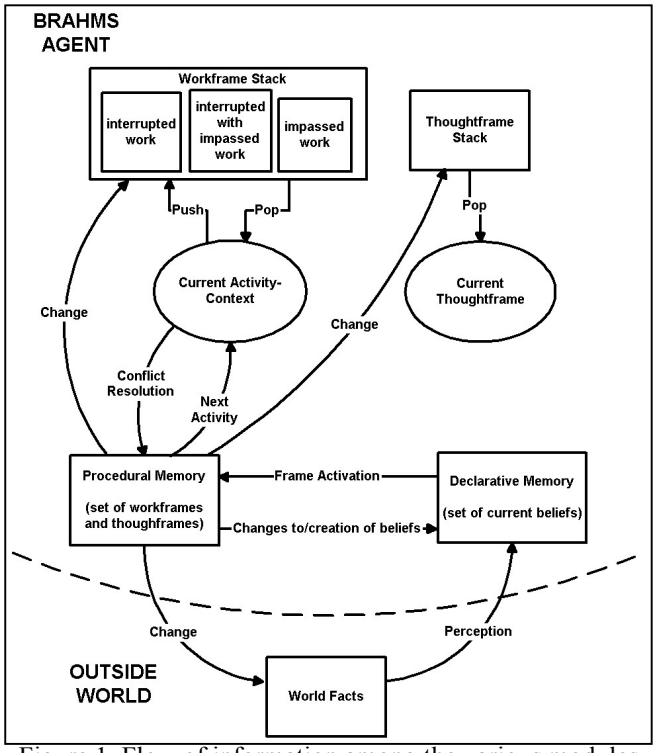


Figure 1. Flow of information among the various modules in a Brahms Agent

We show a similar figure (Figure 1) for a Brahms agent as the figure in Anderson's book about ACT-R (Anderson and Lebiere, 1998)

References

Anderson, J. R. and Lebiere, C. (1998) The atomic components of thought. Mahwah, NJ.: Lawrence Erlbaum Associates.

Clancey, W. J., Sachs, P., et al. (1998) "Brahms: Simulating practice for work systems design", International Journal on Human-Computer Studies 49: 831-865.

Leont'ev, A. N. (1978) Activity, Consciousness and Personality. Englewood Cliffs, NJ: Prentice-Hall.

Sierhuis, M., Clancey, W. J. et al. (2009a) "Brahms: An Agent-Oriented Language for Work Practice Simulation and Multi-Agent Systems Development ", in M. D. Rafael H. Bordini, Jürgen Dix, Amal El Fallah-Seghrouchni (ed.) Multi-Agent Programming, 2nd Edition: Springer.

Sierhuis, M., Clancey, W. J., et al (2009b) "NASA's OCA Mirroring System: An application of multiagent systems in Mission Control", 8th Int. Conf. on Autonomous Agents and Multiagent Systems 2009. Budapest, Hungary

Suchman, L. A. (1987) Plans and Situated Action: The Problem of Human Machine Communication. Cambridge, MA: Cambridge University Press.

Vygotsky, L. S. (1978) Mind in Society: The Development of Higher Psychological Processes. Cambridge, MA: Harvard University Press.

Tutorial: The CLARION Cognitive Architecture

Nicholas Wilson (wilson3@rpi.edu)
Cognitive Science Department
Rensselaer Polytechnic Institute,
110 Eighth Street, Troy, NY 12180 USA
Telephone: (518) 276-2692 | Fax: (518) 276-3017

Michael Lynch (lynchm2@rpi.edu)
Department of Language, Literature and Communication
Rensselaer Polytechnic Institute, Troy, NY 12180 USA

The half-day tutorial introduces participants to the CLARION cognitive architecture and presents a detailed description, as well as simulation examples, advanced topics, and demonstrations. It will combine conceptual (psychological), theoretical, and implementation aspects of the architecture. Participants should have some prior exposure to cognitive architectures and artificial neural networks. Preferably, participants should also have some experience with programming languages (in particular Java). However, prior understanding of these areas can be limited, as both basic and advanced topics related to cognitive modeling using CLARION will be covered.

Tutorial Outline

A General Overview of CLARION (15 min.)

In this section, an introduction to cognitive architectures in general, and CLARION in particular, will be presented. CLARION will be compared to various other architectures and a brief discussion of some past and current applications of CLARION will be presented along with cognitive justifications and implications.

CLARION is a unified, comprehensive theory of the mind based on two basic theoretical assumptions: representational differences and learning differences of two different types of knowledge --- implicit vs. explicit (Sun, Merrill, & Peterson, 2001; Sun, Slusarz, & Terry, 2005), among other essential assumptions/hypotheses (Sun, 2003).

The first assumption, the representational difference between these two types of knowledge, relates to accessibility. In each subsystem of CLARION, the top level contains easily accessible explicit knowledge whereas the bottom level contains less accessible implicit knowledge.

The second assumption of CLARION concerns the different learning processes in the top and bottom levels of each subsystem (Sun et al., 2001, 2005). In the bottom level, implicit associations are learned through gradual trial-and-error learning. In contrast, learning of explicit knowledge is one-shot and captures its abrupt availability. The emphasis on bottom-up learning (i.e., the transformation of implicit knowledge into explicit knowledge) is, in part, what distinguishes CLARION from other cognitive architectures (although top-down learning is also a capability of CLARION).

In addition to the aforementioned theoretical assumptions, CLARION is a cognitive architecture composed of four main subsystems: the Action-Centered Subsystem, the Non-Action-Centered Subsystem, the Motivational Subsystem, and the Meta-Cognitive Subsystem.

The Action-Centered Subsystem (60 min.)

In this section, the Action-Centered Subsystem (ACS) will be defined in detail. The structure and design of the various aspects of the ACS, along with the learning mechanisms and the properties of the model, will be presented. Finally, a series of simulation examples related to the operations within the ACS will be presented.

The Action-Centered Subsystem is used mainly for action decision-making. In the ACS, the top level generally contains simple "State → Action" rules, while the bottom level uses multi-layer perceptrons to associate states and actions. Reinforcement learning algorithms (usually with backpropagation) are used in the bottom level while rule learning in the top level is mostly "one-shot" and can be performed bottom-up (via "explicitation") or independently (e.g., through linguistic acquisition).

The ACS has been used to model anything from navigation in minefields (Sun et al., 2001) to Towers of Hanoi, etc. In addition, because CLARION focuses on the dichotomy between explicit and implicit knowledge, benchmark psychological tasks used to demonstrate implicit learning have also been successfully modeled and explained (Sun et al., 2005).

The Non-Action-Centered Subsystem (45 min.)

Similar to the section on the ACS, this section will detail the Non-Action-Centered Subsystem (NACS). The structure and design of the various aspects of the NACS, along with the learning mechanisms and the theorems describing the properties of the model, will be presented. In addition, as with the section on the ACS, a series of simulation examples demonstrating the operations within the NACS will be presented.

The Non-Action-Centered Subsystem is used to store declarative ("semantic" and episodic) knowledge and is responsible for reasoning in CLARION. In the NACS, the top level contains simple associations while the bottom level involves a nonlinear neural network. Associative learning

algorithms (e.g., backpropagation or contrastive Hebbian) are generally used in the bottom level whereas associations in the top level are mostly learned "one-shot" (similar to the ACS).

The NACS has mostly been used to simulate memory and reasoning. In particular, CLARION was able to capture the effect of mixed rule-based and similarity-based reasoning (e.g., when judging the likelihood of simple deductive forms). In addition, other reasoning phenomena (e.g., inheritance-based reasoning, reasoning from incomplete information, etc) have also been explained using CLARION (e.g., Sun & Zhang, 2006).

The Motivational and Meta-Cognitive Subsystems (30 min.)

In the fourth section, the structure and design of the motivational (MS) and meta-cognitive (MCS) subsystems will be explored in detail. In addition, several past and current simulation examples related to the operations within the MS and the MCS will be presented.

The Motivational Subsystem contains both low-level (physiological) and high-level (social) primary drives that take into account both environmental and internal factors in determining drive strengths. These drive strengths are reported to the Meta-Cognitive Subsystem, which regulates not only goal structures but also other cognitive processes as well (e.g., monitoring, parameter setting, etc). For more details on motivation and meta-cognition see Sun (2003, 2007, 2009).

Simulations using these subsystems, for example, have shown how anxiety-inducing drives can affect the parameters within the ACS in terms of explicit vs. implicit response weighting and overall performance (Wilson et al., 2009). Other simulations have addressed the combination of drives in the MS toward the setting of goals by the MCS. On this basis, models of human personality have been developed.

Introduction to the CLARION Library (30 minutes)

The CLARION implementation (in Java) has recently undergone a number of improvements and enhancements allowing for the simulating of a wide variety of tasks, as well as interfacing with a variety of virtual environments. In the last section of the tutorial, an overview of the CLARION Library will be presented. Participants will be given copies of the newest release of the library and will be shown how it can be used to run new and existing simulations.

Relevance for Cognitive Science

The CLARION cognitive architecture is well established and has been the subject of more than 100 scientific papers and several books. CLARION is particularly relevant to cognitive scientists because of its strong psychological plausibility and the breadth of its application to cognitive modeling and simulation. In CLARION, each structure corresponds to a psychological process/capacity. CLARION-based models have been used to explain data as diverse as implicit learning, cognitive skill acquisition, inductive and deductive reasoning, meta-cognition, motivation, personality, and social simulations (Sun, 2006).

Presentation Details & History

Descriptions and demonstrations during the presentation will be provided using PowerPoint and the Eclipse Java development environment.

Participants in the tutorial are encouraged to ask questions throughout the presentation to clarify any ideas described. The presenters are versed in both the conceptual and implementation details of the CLARION cognitive architecture.

An older variation of the proposed tutorial had been presented at the 30th Annual Meeting of the Cognitive Science Society in Washington D.C. as well as the 2009 International Joint Conference on Neural Networks in Atlanta, GA. In addition, this tutorial has been given as a lecture series on several occasions for various courses in Cognitive Science at Rensselaer Polytechnic Institute.

Sample Materials

- A complete technical specification of CLARION: http://www.cogsci.rpi.edu/~rsun/sun.tutorial.pdf
- A list of CLARION-related publications: http://www.cogsci.rpi.edu/~rsun/clarion-pub.html
- Current versions of the CLARION Library, slides, etc.: http://www.cogsci.rpi.edu/~rsun/clarion.html

References

Sun, R. (Ed.). (2006). *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge, UK: Cambridge University Press.

Sun, R. (2003). A Tutorial on CLARION. Technical report, Cognitive Science Dept., Rensselaer Polytech. Institute. http://www.cogsci.rpi.edu/~rsun/sun.tutorial.pdf

Sun, R. (2007). Motivation and metacognitive control of CLARION. In: W. Gray (ed.). *Modeling Integrated Cognitive Systems*. New York, NY: Oxford University Press.

Sun, R. (2009). Motivational representations within a computational cognitive architecture. *Cognitive Computation*, 1, 91-103.

Sun, R., Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science, 25*, 203-244.

Sun, R., Slusarz, P., & Terry, C. (2005). The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review, 112*, 159-192.

Sun, R. & Zhang, X. (2006). Accounting for a variety of reasoning data within a cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, *18*, 169-191.

Wilson, N., Sun, R., & Mathews, R. (2009). A Motivationally-based Simulation of Performance Degradation Under Pressure. *Neural Networks*, 22, 502-508.