Toward Undifferentiated Cognitive Models

Colin Kupitz[1] (colin.kupitz.ctr@us.af.mil), Aaron Eberhart[2] (aaron.eberhart@gmail.com)
Daniel Schmidt[3] (danielschmidt98@gmail.com), Christopher Stevens[3] (christopher.stevens.28@us.af.mil)
Cogan Shimizu[2] (coganshimizu@ksu.edu), Pascal Hitzler[2] (hitzler@ksu.edu)
Dario D. Salvucci[4] (salvucci@drexel.edu), Benji Maruyama [3] (benji.maruyama@us.af.mil),
& Christopher W. Myers[3] (christopher.myers.29@us.af.mil)
[1]Oak Ridge Institute for Science & Education at AFRL
[2]Kansas State University
[3]Air Force Research Laboratory
[4]Drexel University

Abstract

Autonomous systems are a new frontier for pushing sociotechnical advancement. Such systems will eventually become pervasive, involved in everything from manufacturing, healthcare, defense, and even research itself. However, proliferation is stifled by the high development costs and the resulting inflexibility of the produced systems. The current time needed to create and integrate state of the art autonomous systems that operate as team members in complex situations is a 3-15 year development period, often requiring humans to adapt to limitations in the resulting systems. A new research thrust in interactive task learning (ITL: Laird et al., 2017) has begun, calling for natural human-autonomy interaction to facilitate system flexibility and minimize users' complexity in providing autonomous systems with new tasks. We discuss the development of an undifferentiated agent with a modular framework as a method of approaching that goal.

Keywords: cognitive model; cognitive agent; instruction following; learning

Introduction

Autonomous systems are a new frontier for socio-technical advancement. Such systems will be required to team with humans, potentially operating at the level of peers and not just subordinates. One such *autonomous synthetic teammate* (AST) demonstrated that they can be included in teams without detriment to teams' or team members' performance (McNeese, Demir, Cooke, & Myers, 2017; Myers et al., 2019). Nonetheless, there remain two significant obstacles to the wider adoption of synthetic agents operating as peers or subordinates in complex environments: 1) their development and evaluation time and 2) their limited scope of transfer once developed. The AST took approximately nine years to develop plus an additional year to evaluate (Ball et al., 2010; Rodgers, Myers, Ball, & Freiman, 2013), and yet it would require further research and development to adapt it to perform a different task within the same domain and even more to adapt it to an entirely new domain.

Instructions are a ubiquitous part of the human experience. They provide guidance through a space of potential states to a solution (i.e., the problem space; Newell and Simon 1972). Without instruction, one is free to roam about the problem space freely in an attempt to find, or discover, the solution. Instruction also plays a critical role in our ability to advance as a civilization: calculus, laws of physics, and other advanced domains do not need to be re-discovered by each generation, but are taught through instruction. Although learning is sometimes characterized as the acquisition of all skills needed for a given task, in fact, the learning of complex tasks more typically reflects the integration of already-known processes (e.g., interactive routines; Gray 2008) in novel ways (Gray, Sims, Fu, & Schoelles, 2006) – of which one way is through instruction (Salvucci, 2013).

Recent advances have demonstrated the ability to turn a set of instructions into declarative knowledge that is then used to enable performance across paradigms of different complexity: ranging from those typically used by experimental psychology to dialing while driving an automobile (Salvucci, 2013). In a similar vein, Kirk, Mininger, & Laird (2016) have successfully demonstrated the ability to train robots on novel tasks through direct interaction. The objective associated with the presented research is to leverage past work on instruction learning to address the development and transfer issues, simultaneously. Specifically, we propose a generalizable, *undifferentiated agent* (uAgent) that can learn a new task relatively independently through written instruction and be trained to a desired level of proficiency with reduced developer intervention.

To achieve these goals, the uAgent was developed with a modular architecture, to allow for expansion into other tasks and fields with minimal burden to other researchers. The components of the architecture are instruction parsing, an ontology of instruction, declarative memory representation, and procedures for accomplishing the instructed tasks. Each of the uAgent components are discussed in the following sections, followed by their integration as a single system. Finally, a case study on development times relative to current approaches to model development times is presented.

In the following sections, we will discuss this development in each of the discretized modules that together represent the uAgent, along with the general specifications for said modules to allow for future revision and expansion. Second, the uAgent will be specialized to desired levels of proficiency using AFRL's Autonomous Research System (ARES; Nikolaev et al., 2016).

Toward Undifferentiated Cognitive Models

The undifferentiated agent, or uAgent, is a system capable of learning new tasks through instruction. The process involves: (1) parsing the instructions in to a structure that can be (2) integrated with prior information within a declarative memory system through an ontology of instruction. Given an integrated declarative system, the uAgent (3) associates it newly acquired knowledge with existing interactive routines through a controlled vocabulary. Finally, the uAgent is ready to (4) perform the task based on its knowledge of instructions. Components associated with each step will be discussed in detail, below.

Instruction Parsing

Though many advances have been made in the field of natural language processing (NLP), it still remains a challenging problem to extract complex rules and meanings out of text. To make the problem of parsing text more tractable, we use a controlled natural language - Attempto Controlled English (ACE; Fuchs, Kaljurand, & Kuhn, 2008).

A controlled natural language is a language that permits only a subset of grammatical constructions available in natural language (in this case, only present perfect tense, no use of second person, and a very specific syntax for commands). These restrictions make it possible for software (e.g., the Attempto Parsing Engine, APE; Fuchs et al., 2008) to automatically parse sentences written in the controlled language into logical statements called discourse representation structures (DRS). These structures approximate first-order logic.

The requirements of the Instruction Parsing module are as follows. First, the language requirements of incoming instructions must be specified (here, we use the ACE controlled language). Second, these instructions must be processed into a form compatible with the target declarative memory system structure to be used by the acting uAgent.https://www.overleaf.com/project/607840726e9a778f9940dc4b

In the current system, we begin with plain English instructions of a task of interest. Two types of tasks we are currently working with include basic experimental psychology tasks - psychomotor vigilance (Dinges & Powell, 1985) and visual search (Treisman & Gelade, 1980) - and a material engineering task in which an individual guides a set of experiments with a 3D printer (Nikolaev et al., 2016). In both cases, instructions are re-written by hand into sentences that follow the rules of the ACE language. Then, the instructions are provided to APE [1] to translate the ACE sentences into DRS structures, which are then integrated into a declarative memory structure based on an ontology of instruction.

Ontology of Instruction

The primary function of the Ontology module is to directly formalize the structure of information necessary to complete the desired tasks and goals of the uAgent. This furthers the goal of the uAgent as a whole, as it provides the foundation for the structure and relationships within the declarative memory system used by the uAgent (see Figure 1).

In order to create a system that is both generalizable and able to correctly handle diverse types of instructions, an ontology was created that is capable of representing instructions for a cognitive agent task. This instruction ontology can be used to directly inform relationships among tokens of knowledge within a cognitive agent performing tasks. Further, it can be leveraged to derive a semantically-anchored declarative memory system for long-term storage for knowledge, such as a knowledge graph (Noy et al., 2019). It can also support experiment design, irrespective of any agent, by providing a structured basis for evaluating the content and design of similar tasks. Additionally, because an ontology contains a precise axiomatization of the knowledge it is supposed to represent, deductive reasoning techniques can be applied to detect possible gaps or errors in instructions. Further information regarding the ontology can be found in (Eberhart et al., 2020).

[1] http://attempto.ifi.uzh.ch/site/resources

English: You will be seated in front of a computer screen. A letter will appear in the middle of the screen. When you see the letter, press the spacebar.
ACE: p:psychomotorVigilance is a task. There is a screen. There is a letter. There is a subject. The n:spacebar is a button. If the task is active then the subject v:watchesFor the letter and the letter v:appearsOn the screen. If the letter v:appearsOn the screen then the subject presses the n:spacebar. The task is active.

Table 1: PVT instructions in English and ACE.

The ontology was developed to represent the relationships between steps, items, actions, instructions associated with tasks relying on a graphical user interface. To ensure a potentially high degree of complexity in instructions, the multi-stage Intelligence, Surveillance, & Reconnaissance Mutli-Attribute Task Battery (ISR-MATB) task (Frame et al., 2019) was used as an example task when developing the ontology. Because it has multiple interconnected cognitive tasks, using the ISR-MATB aids in the development of a undifferentiated representation of instruction knowledge. The ontology was produced by following the Modular Ontology Modeling (MOMo) methodology, outlined in (Krisnadhi & Hitzler, 2016; Hitzler & Krisnadhi, 2018; Shimizu, Hammar, & Hitzler, 2021), and is designed to ensure high quality and reusability of the ontology. The adaptability required to model the ISR-MATB task, together with the modular techniques used to create it, mean that the ontology can very easily be adapted for use in new tasks.

Currently, DRS items from instruction are obtained as input to the ontology whenever an agent begins learning through instruction (see Figure 1). The DRS structured information is then available to an agent during a task, and additional knowledge that the agent acquires can be added to supplement this. As new tasks are implemented and tested

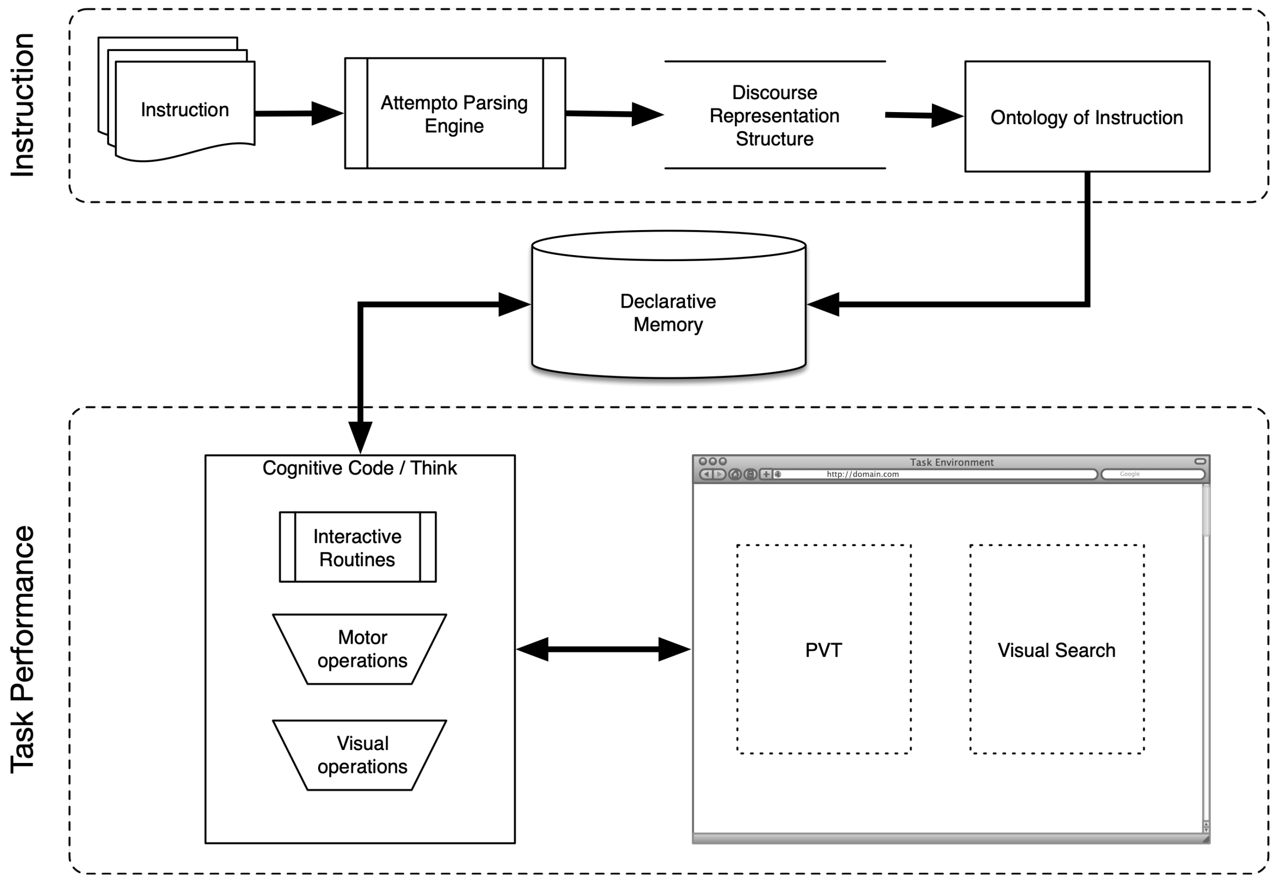

Figure 1: Architectural components of the undifferentiated cognitive agent (uAgent).

this process is simple to extend to encompass new types of knowledge, since the structure of the ontology and the format of input data is agnostic to the actual content of the knowledge represented.

To summarize, the Ontology leverages information theory and formal logical structures to ensure that pertinent information is assimilated in the most reasonable, orderly fashion possible from a theoretical standpoint. Importantly, this ensures that any future expansions of the uAgent into additional research fields and tasks will be expedited, as any additional pertinent information can be distilled directly into the most useful form through the ontology and into the uAgent's declarative memory system.

Declarative Memory System

The declarative memory system of the uAgent contains information associated with parsed instructions, prior knowledge, and a controlled vocabulary connecting verbs to known procedures in the procedural system. The approach taken to represent the declarative memory system was a knowledge graph (Noy et al., 2019), which describes facts, actions, objects of interest, and the relationships between them.

The uAgent here stores a knowledge graph built from declarative chunks. Specifically, it incorporates chunks that, using ACT-R-like slot-value pairs (Anderson, 2007), links knowledge together in a graph by having one chunk's slots include other chunks as values for those slots. As such, the representation is flexible enough to incorporate all the declarative knowledge needed in the instructions for our purposes, including not only basic actions but also conditionals and sequences of actions.

A declarative memory system structured as a knowledge graph requires connections to real action/observable behavior to ground the information in the actions available within the instructed task. Without grounding, the agent can have all of the available information about the task but no way to observe or interact with its task environment. To this end, a controlled vocabulary (CV) was introduced to map verbs onto concepts or actions. For example, a CV entry for "search" could map onto an interactive routine (Gray, 2008) instructing the agent to attend a location, locate an item there, and encode it. Within the ACT-R paradigm, this led to creating a new class of chunks for CV entries. These chunks contain the CV term and map to a production or set of procedures built into the agent prior to instruction, thereby grounding that term onto a known set of actions (Ji, van Rij, & Taatgen, 2019).

Novel task strategies can be constructed using these grounded interactive routines, thereby allowing the agent to interact with an environment for which it was not specifically designed and perform tasks without needing to have a whole set of bespoke procedures and knowledge built into it.

Altogether, the CV defines the set of verbs which are already grounded to behavior(s); in essence, it represents the agent's knowledge of general behaviors *a priori*. Accordingly, by relating the CV to task appropriate interactive routines, we ensure the knowledge is inherently grounded to the environment.

As a module, the knowledge graph serves as the basis of the uAgent memory: it contains information pertinent to the instructions given, but processed through the lens of overall task knowledge it should have before hand (i.e. the Ontology). It must follow the format of the structures provided within the formal ontology, and further, should use a defined controlled vocabulary to map those terms onto active agent behaviors where appropriate.

Procedural Memory System

Given the above declarative memory structures for representing instructions, the system needs to ground concepts to simulated actions via interactive routines (i.e., embedded or learned procedural knowledge). Models developed in cognitive architectures such as ACT-R (Anderson, 2007) or Soar (Laird, 2012) typically use production systems to represent this procedural knowledge. Here, we take a different approach, using *cognitive code* (Salvucci, 2016) to maintain and execute procedural knowledge. Cognitive code embeds procedural knowledge into a common programming language, facilitating the development of model code while maintaining the most important properties of human-like abilities and limitations inherent to any cognitive architecture. Specifically, we are using the *Think* architecture [2], which incorporates declarative and procedural concepts taken primarily from ACT-R and provides them for easy use via the Python programming language.

Several components of this project have led to important extensions of Think's code base. One extension involves the integration of traditional declarative memory with Think execution. The default Think code base includes a declarative memory module that embodies ACT-R's core theory of memory (Anderson, 2007). For this project, we bypass this traditional memory module, and instead use the ontology and knowledge graph described earlier as the model's primary long-term declarative storage. The Think procedures still maintain short-term declarative items, namely those that comprise the current "context" during execution (i.e., information that would traditionally be stored in ACT-R's *imaginal* buffer).

Besides this integration of a new type of declarative memory, the other critical extension of Think's code base relates to the realization of procedural learning. Although cognitive code can often be made to operate in ways very similar to traditional production systems, a critical difference is that cognitive code cannot (in most cases) be constructed during simulation as some architectures have done with procedural learning. For example, ACT-R's production compilation mechanism (Taatgen & Lee, 2003)) transforms declarative instructions into procedural form which eventually leads to gradual learn of new procedures; the most critical aspect of this learning is that, at first, a model must perform a declarative retrieval to remember the learned instruction before executing it, but later, the compiled instruction (in the form of a production rule) skips the retrieval and simply executes the associated action. Although Think does not create new code on the fly in the same way, we have augmented its capabilities by adding procedural learning that captures the essence of ACT-R's production compilation—specifically, in performing declarative retrievals early in learning (which take additional time and may fail), and then skipping these retrievals later in learning (leading to gradual speedup and eventually fast performance).

As a module, the cognitive code contained with Think serves as the "actual" uAgent, so to speak – it represents the system which is deciding and acting upon the best course of behavior during any task. In theory, this could be replaced with any number of cognitive architectures, provided they are capable of using the prespecified knowledge graph structures to serve as the basis of memory, and further, have a correctly specified controlled vocabulary to map that knowledge graph onto the behaviors known to the system *a-priori*.

System Integration

To develop the uAgent with an adaptable framework going forward, we used a modular design approach (Bryson, 2000). In particular, this capitalizes on the interdisciplinary nature of the researchers involved while simultaneously minimizing the overall burden of coordination. To that end, during development the fundamental uAgent capabilities were segregated into discrete modules. Overall integration of these modules was then assigned to a few individuals, with the entire research team meeting to discuss overall design strategies as appropriate. Of note, this approach also allowed for a degree of asynchronous development across the research teams involved, thereby reducing the project coordination burden significantly. In addition, the modular approach ensures that the uAgent will be adaptable to other fields of research and task performance, as future research can adapt the uAgent by focusing on a specific uAgent module where appropriate. We now move on to discuss the primary modules of interest in the uAgent.

Given the interdiscipinary nature of this research, we first settled on the use of the open source Python as the primary programming language, integrating each individaul uAgent module into one coherent system. In particular, this allows us to utilize the Think system (Salvucci, 2021) in order to simulate both the uAgent behavior, and the enviroment in which is it actively behaving. Further, whenever these separate modules are expected to interact directly, we worked to determine the best overall form of interface and information exchange

[2]https://github.com/salvucci/think

to facilitate ease of integration and future expansion. To that end, we now note the primary interface decisions we made during said development.

First, we concluded that the Ontology of instruction should serve as a form of blueprint for the knowledge graph. This ensures that the Instruction Parsing module will produce structures that can be assigned to knowledge graph structures where appropriate. Effectively we are leveraging the relations inherent to the Ontology in order to improve the capabilities of the instruction interpretation; in essence, the uAgent can make informed assumptions about the informational structure while processing any incoming instructions.

Similarly, to ensure the agent is capable of acting on those instructions, we concluded that the knowledge graph module should also consider a controlled vocabulary representing the behaviors found within the Think cognitive agent. This controlled vocabulary is essentially the actions that the think uAgent is capable of performing in the current environment. In essence, we ensure that the knowledge graph structures which serve as the basis of the Think agent memory also have a direct mapping onto Think behaviors where appropriate.

Altogether, we integrate each of the uAgent modules into a coherent end-to-end system, and explicitly define the interface requirements necessary to ensure the system can take instructions as input, and produce human behavior with high fidelity.

Case Study

As a proof-of-concept for the approach, we built an end-to-end system that takes ACE instructions of cognitive tasks commonly used in basic research - psychomotor vigilance and visual search - converts the instructions into a knowledge representation capable of performing the task, and then performs the task in a simulated environment.

As an exercise to determine if the current approach could save time with respect to building a traditional ACT-R model, we compared the amount of time it took to build a model of a set of cognitive tasks with the amount of time it took to write ACE instructions of the same task. The task we used was a novel task battery that includes a set of commonly used experimental psychology tasks (Frame et al., 2019; Eberhart et al., 2020). This battery includes four subtasks - psychomotor vigilance, visual search, auditory search, and multi-cue decision-making. We a built model of the task in a Java implementation of ACT-R 6 and wrote a set of ACE instructions for it.

It took approximately 120 hours to build the ACT-R model, but only approximately 30 hours to write the ACE instructions. This exercise suggests that the present method could potentially save a substantial amount of time in developing new models and agents. Moreover, writing the ACE instructions required only a brief reading of publicly available tutorials on the ACE language, and not training and experience in writing ACT-R models, the latter of which can be substantial. In our proof-of-concept system, we showed that the ACE instructions of the PVT and Visual Search subtasks could be successfully integrated into the ontology and the agent could use this resulting knowledge to perform the task. We are working toward end-to-end demonstrations of the other two subtasks.

Conclusions & Future Work

Progress toward a modeling framework capable of being taught new tasks through written instruction was presented. As evidenced in the uAgent case study, such an approach will likely significantly reduce model and agent development times. Further, the modular-based approached toward uAgent development will facilitate the integration of other cognitive architectures by using the uAgent declarative memory as its knowledge repository.

While the uAgent shows promise as a means for teaching models how to perform new tasks, multiple challenges remain. For example, it is unreasonable to assume that the union of instruction and prior knowledge is sufficient for completing an instructed task. As a result, we have begun developing approaches for detecting and resolving gaps in a uAgents knowledge base. This work will require multidisciplinary approaches to model development coupled with empirical investigations into when and how humans detect and resolve knowledge gaps.

In order to better understand how humans form representations from instructions and identify and resolve gaps in understanding from those instructions, we plan to conduct a human-subjects experiment using the task battery described above. We plan to teach participants to perform the tasks in the battery using either a complete set of instructions, or a set with ambiguities with respect to certain types of knowledge. We plan to use think-aloud protocols to track how participants extract knowledge from these instructions and how they detect and resolve uncertainty. We believe this will provide insights in how to improve the undifferentiated model's knowledge acquisition.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0386 to Kansas State University, 19RHCOR089 3003B & 3003C to AFRL, and FA9550-18-1-0371 to Drexel University.

References

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* Oxford University Press.

Ball, J., Myers, C., Heiberg, A., Cooke, N. J., Matessa, M., Freiman, M., & Rodgers, S. (2010, aug). The synthetic teammate project. *Computational and Mathematical Organization Theory*, *16*(3), 271–299. doi: 10.1007/s10588-010-9065-3

Bryson, J. (2000). Cross-paradigm analysis of autonomous agent architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, *12*(2), 165–189.

Dinges, D. F., & Powell, J. W. (1985). Microcomputer analyses of performance on a portable, simple visual rt task during sustained operations. *Behavior research methods, instruments, & computers*, *17*(6), 652–655.

Eberhart, A., Shimizu, C., Stevens, C., Hitzler, P., Myers, C. W., & Maruyama, B. (2020). A Domain Ontology for Task Instructions. In B. Villazón-Terrazas, F. Ortiz-Rodríguezm, S. M. Tiwari, & S. K. Shandilya (Eds.), *Knowledge graphs and semantic web. second iberoamerican conference and first indo-american conference, kgswc 2020* (pp. 1–13). Mérida, Mexico: Communications in Computer and Information Science, vol. 1232.

Frame, M., Lopez, J., Myers, C., Stevens, C., Estepp, J., & Boydstun, A. (2019). Development of an autonomous management system for human-machine teaming with multiple interdependent tasks. In *Presented to the annual meeting of the psychonomic society conference, montreal, qc, canada, november 2019.*

Fuchs, N. E., Kaljurand, K., & Kuhn, T. (2008). Attempto controlled english for knowledge representation. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 5224 LNCS, pp. 104–124). doi: 10.1007/978-3-540-85658-0_3

Gray, W. D. (2008). The Interactive Routine as Key Construct in Theories of Interactive Behavior. In V. Sloutsky, B. Love, & K. McRae (Eds.), *30th annual meeting of the cognitive science society* (p. 127). Austin, TX.

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*(3), 461–482.

Hitzler, P., & Krisnadhi, A. (2018). A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. *CoRR*, *abs/1808.08433*. Retrieved from `http://arxiv.org/abs/1808.08433`

Ji, M. Y., van Rij, J., & Taatgen, N. A. (2019). Discoveries of the algebraic mind: A PRIMS model. *Proceedings of ICCM 2019 - 17th International Conference on Cognitive Modeling*, 71–76.

Krisnadhi, A., & Hitzler, P. (2016). Modeling with ontology design patterns: Chess games as a worked example. In P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, & V. Presutti (Eds.), *Ontology engineering with ontology design patterns – foundations and applications* (Vol. 25, pp. 3–21). IOS Press.

Laird, J. E. (2012). *The Soar Cognitive Architecture*. MIT Press.

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., . . . Kirk, J. R. (2017). Interactive Task Learning. *IEEE Intelligent Systems*, *32*(4), 6–21. doi: 10.1109/MIS.2017.3121552

McNeese, N. J., Demir, M., Cooke, N. J., & Myers, C. (2017). Teaming With a Synthetic Teammate: Insights into Human-Autonomy Teaming. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 001872081774322. doi: 10.1177/0018720817743223

Myers, C., Ball, J., Cooke, N., Freiman, M., Caisse, M., Rodgers, S., . . . McNeese, N. (2019). Autonomous Intelligent Agents for Team Training. *IEEE Intelligent Systems*, *34*(2), 3–14. doi: 10.1109/MIS.2018.2886670

Newell, A., & Simon. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

Nikolaev, P., Hooper, D., Webber, F., Rao, R., Decker, K., Krein, M., . . . Maruyama, B. (2016). Autonomy in materials research: a case study in carbon nanotube growth. *Nature Partner Journal: Computational Materials*, *2*(1), 16031. Retrieved from `http://www.nature.com/articles/npjcompumats201631` doi: 10.1038/npjcompumats.2016.31

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., & Taylor, J. (2019). Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, *62*(8), 36–43. doi: 10.1145/3331166

Rodgers, S., Myers, C., Ball, J., & Freiman, M. (2013). Toward a situation model in a cognitive architecture. *Computational and Mathematical Organization Theory*, *19*, 313–345.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science*, *37*(5), 829–860. doi: 10.1111/cogs.12032

Salvucci, D. D. (2016). Cognitive Code : An Embedded Approach to Cognitive Modeling. In *Proceedings of the 14th international conference on cognitive modeling* (pp. 15–20).

Salvucci, D. D. (2021). Interactive Grounding and Inference in Instruction Following. *To appear in Topics in Cognitive Science*.

Shimizu, C., Hammar, K., & Hitzler, P. (2021). Modular ontology modeling. *Semantic Web*. (Under Review.)

Taatgen, N. a., & Lee, F. J. (2003). Production compilation: a simple mechanism to model complex skill acquisition. *Human factors*, *45*(1), 61–76. doi: 10.1518/hfes.45.1.61.27224

Treisman, A., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive psychology*, *12*, 97–136.