Towards Benchmarking Cognitive Models: A Python Library for Modular Environment Specification and Partial Model Generation in ACT-R

Emmanuelle Dietz (emmanuelle.dietz@airbus.com) and Oliver W. Klaproth (oliver.klaproth@airbus.com)
Airbus Central Research & Technology, Hein-Sass-Weg 22
21129 Hamburg, Germany

Abstract

In this paper we present the cognitive modeling library *txt2actr*, which facilitates the specification of an ACT-R environment through simple text files and partially automates the construction of certain components within a cognitive model. Our general purpose goes beyond this library and aims at promoting the modular construction and evaluation of cognitive models. In particular, we suggest to establish benchmarks that allow (i) the competition among models with respect to classical tasks in experimental psychology, and (ii) the evaluation of possibly new or more applied tasks with respect to benchmark models. Such benchmarking proposals can be found in various other disciplines and usually serve as an incentive to improve existing theories and eventually converge towards a common language. Yet, *txt2actr* is far from providing a solution to the associated challenges. It rather serves as a proof of concept by illustrating how two model components for very specific cognitive phenomena in situation awareness can be applied in three different environments.

Introduction

After 50 years, Newell's critism that the scientific community does not *seem in the experimental literature to put the results of all the experiments together* (Newell, 1973, p. 298) still seems to hold. Interestingly, this problem persists independent of the *bands of cognition* (Newell, 1990), that is the proposed models that deal with *lower levels* such as biological processes or the ones which address *higher levels*, such as human reasoning and decision making. As Khemlani and Johnson-Laird (2012) observed for the particular case of modeling human syllogistic reasoning, *the existence of 12 theories of any scientific domain is a small disaster.* Or, as Taatgen and Anderson (2010) put it, *multiple possible models is not just a problem for cognitive architectures but for any scientific theory*.

One bottleneck might be that most cognitive theories are not formalized and therefore ambiguous: There is no commonly accepted language which allows to compare and thus evaluate their cognitive plausibility on a set of benchmark tasks. As noted by Marewski and Mehlhorn (2011), this leads, among others, to the specification problem (Lewandowsky, 1993), how to translate an under-specified hypothesis into a detailed model, and the identification problem (Anderson, 1976), what to do with many different models which are equally capable of reproducing and explaining data.

During the past decades, cognitive architectures (CA) have been proposed (e.g. ACT-R (Anderson 2007), SOAR (Laird, 2012)) to *address under-specified process hypotheses and provide a falsifiable methodology* (Thomson, Lebiere, Anderson, & Staszewski, 2015). These architectures contributed largely to the development of the field and the comparability of cognitive modeling. However, two main issues still persist: Firstly, it requires substantial intellectual commitment to learn, understand and construct models within these architectures (Taatgen & Anderson, 2010). Secondly, these architectures are highly parametrized, which on the one hand provides a great amount of modeling freedom, but on the other hand leads to models which rather capture *the intuitions of the designers* (Thomson et al., 2015). Laird, Lebiere, and Rosenbloom (2017) proposed a common model of cognition as an abstracted framework depicting the *best consensus given the community's current understanding of the mind* on the architectural level. Such a standard model could then be used as a common language for the community and guide researchers by enabling them to include or extend other components and evaluate or develop psychological experiments.

According to Taatgen and Anderson (2010), a good model is characterized as being applicable to various tasks, as simple as possible and able to predict outcomes of new tasks. That means that the metric for a good model can then be specified by its generalizabillty (Thomson et al., 2015) and its predictability, including predictions of yet untested cognitive phenomena (Ragni, 2020). These models should be built out of components (Taatgen & Anderson, 2010) and the applicable strategies and heuristics should be rather *selected by the model than by the designer* (Thomson et al., 2015).

According to Ragni (2020), another difficulty is how *to identify the relevant problems* (or tasks) that a model should account for. He suggests to establish generally accepted benchmarks, similar to the PRECORE Challenge (Ragni, Riesterer, & Khemlani, 2019) for human reasoning tasks. The evaluation of this challenge was done with the benchmarking tool Cognitive COmputation for Behavioral Reasoning Analysis (CCOBRA) framework,[1] which was thereafter again applied for new prediction mechanisms in individual human syllogistic reasoning (Dietz Saldanha & Schambach, 2020). The success of establishing benchmarks and developing competitions can be observed in other disciplines (e.g. SAT,[2]

[1] https://github.com/CognitiveComputationLab/ccobra
[2] https://satcompetition.github.io/2021/

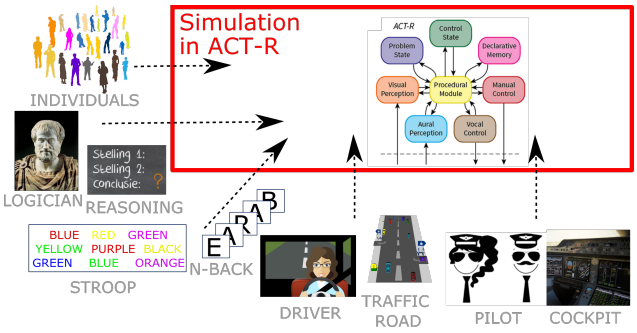

Figure 1: Currently, a majority of model and environment specifications in ACT-R are manually implemented.[3]

ASP (Gebser, Maratea, & Ricca, 2020)), which have the beneficial side effect to improve existing approaches, and more importantly, motivating the scientific community to agree on a common language. Furthermore, contrasting cognitive models with benchmark models (such as statistical baselines or data-driven neural networks) will determine the current empirical upper bound of the models' performance (Riesterer, Brand, & Ragni, 2020). At the same time these upper bounds can serve as new incentives for future models to outperform the benchmark models. However, the previously mentioned criteria for good models should be the main focus. In particular even if machine learning techniques might have better predictive overall performances, the generalization of models across a range of paradigms and conditions can be more powerful. As (Lebiere et al., 2013) stated, approaches based on cognitive modeling require less data and fewer domain-specific assumptions to be parametrized as they can be guided by cognitive constraints. Furthermore, they have the advantage to combine symbolic structures and statistical parameters Taking these proposals as a starting point, we suggest an environmental setting where cognitive models can be benchmarked according to their performance with a set of tasks. In the ideal case, if results are openly shared the ones who have the experimental data but are missing the best predicting models can benefit from the ones who have the cognitive models and vice versa. In this paper, we present the software library *txt2actr* in which the task, the environment, and the model can partially be specified through text files. With this library we aim at developing a modular task design through an ACT-R interface and the parametrization of ACT-R models by a modular and guided production and chunk engineering process. *txt2actr* serves as a proof of concept to address some of the challenges discussed above, and is far from providing a solution or a complete benchmark of tasks or models.

Related Work

The approaches we briefly discuss here are related in the sense that they emphasize the importance of generalizing or benchmarking models, which we consider highly relevant for the challenges we intend to address. Interestingly they all address this issue on a different level of cognition.

Salvucci (2013) proposes a single model of cognitive skill acquisition in ACT-R by reusing component skills across 7 different task domains. The results are a step towards a more unified account of skill learning and demonstrate that a model can reuse knowledge by transferring it to various tasks.

One of the goals suggested by Taatgen (2013) is to reuse cognitive processes and structure them in a way so that they can be applied in many different combinations, similarly to a construction kit that should be deployable throughout all tasks. As almost all cognitive models suffer from the problem of prior knowledge, transfer in cognitive control, that is the process by which goals or plans influence behavior, might be a a promising approach to address this issue, where processes of cognitive control are based on skills.

Marewski and Mehlhorn (2011) specify 39 different process models, which should not only predict what decision a person will make, but also how the information used to make the decision will be processed. In particular, they focus on a class of models that makes decisions by exploiting the accessibility of memory contents. For this purpose they choose to model a task for recognition heuristic. These 39 models either differ in very small aspects or very fundamental assumptions about processing. The main purpose of Marewski and Mehlhorn is not to advocate any particular process model for the task in consideration but rather using the debate as a case study to provide a methodological primer on how architectures like ACT-R can be used to lend precision to theorize decision processes. By implementing models of different levels of description and specificity in one architectural modeling framework, they make the models and their predictions comparable providing a basis for future model tests.

GOMS (Card, Moran, & Newell, 1983) contains hierarchical methods, visual and memory stores, and control constructs and aims at explaining expert routine behaviors and reduce the effort for detailed task analysis and cognitive modeling techniques. Amant and Ritter (2004) provide an automatic generation of GOMS models into ACT-R models. However, it suffers from under-constraints in many areas, for example visual processing (Amant, McBride, & Ritter, 2006). The extension SGOMS (West & Nagy, 2007; West & Somers, 2011) assumes that cognitive modeling at the level of psychological experiments (micro cognition) can scale up to higher level task, such as dealing with task interruptions, by an additional higher-level control structure and multi-tasking.

Cognitive Modeling with *txt2actr*

The python library *text to ACT-R*, abbreviated as *txt2actr*, provides (i) an interface between text files that describe and dynamically change the environment which the cognitive model interacts with during the ACT-R simulation and (ii) a partial and modular construction of the cognitive model. *txt2actr* is publicly available on github.[4]

[3] The image in the red box is from Anderson and Borst (2017).

[4] https://github.com/eadietz/txt2actr

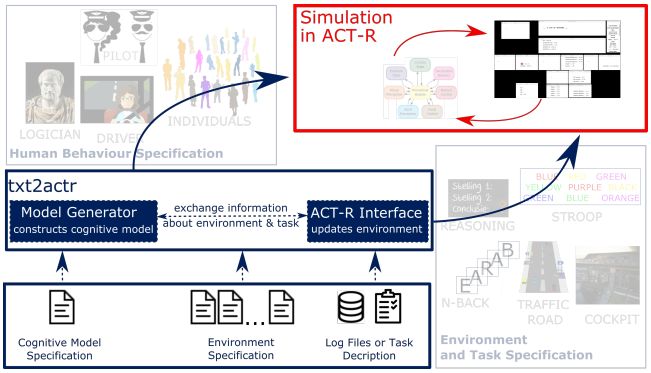

Figure 2: Overview of the two main components in *txt2actr*.

Figure 2 shows the two main components of *txt2actr* in the blue box (labeled txt2actr), where the *ACT-R interface* (on the right) and the *(partial) model generator* (on the left) interact and depend on each others specifications. The necessary environment specifications and cognitive model specifications can be defined in the respective text files that are in the blue box below the labeled txt2actr box. The task descriptions or the log files about the environment and the cognitive model file replace the specifications of the individual environment and tasks (on the lower right corner) and the individual human behavior (upper left corner). The decision to structure *txt2actr* this way is driven by the idea that ideally, a set of cognitive models could then be systematically evaluated with respect to a set of various tasks. Whether such a generic parametrization of tasks or models will eventually be possible needs to be further investigated. Additionally, we aim at a lower inhibition threshold for cognitive modeling, which should be usable by the ones with simulation needs but with little or no experience with cognitive modeling.

We chose the cognitive architecture ACT-R as basis our purpose as it provides a wide range of functionality, it is well established within the community and has a very well documented manual (Bothell, 2020) including an extensive tutorial. ACT-R (Anderson, 2007) is a theory about how human cognition works. It allows to get a better understanding human cognition by simulating different cognitive functions. Each function is represented by a particular (and independent) module that communicates with other modules through buffers. Knowledge in ACT-R is either encoded as declarative memory or procedural rules. Cognitive architectures are also used for tasks within the real world, such as aircraft cockpit or car driving environments (Salvucci, 2006).

We heavily rely on the already existing and publicly available python module actr.py which allows a direct interaction within the ACT-R environment in Lisp through python.[5]

ACT-R Environment Interface

We will briefly introduce the two most important aspects that affect the dynamics of the (ACT-R) environment and can be modified by *txt2actr* during the simulation:

[5]The tutorial and actr.py can be found here `http://act-r.psy.cmu.edu/software/`

vision	description	y position	x position	width	length	fontsize	x text position	y text position	header labels
leading_car	showing red when leading car brakes	255	410	550	250	12	25	25	car_normal,car_brake,car_hold
own	showing own behavior	590	410	250	150	12	50	50	clock,brake,react_emg,car_collision
clock	showing the time	590	710	160	80	12	15	4	time

image	description	y start	x start	y end	x end	color	window	label
backlight_1	should appear when leading car brakes	90	10	180	100	back-light.gif	leading_car	car_brake
backlight_2	should appear when leading car brakes	90	400	180	100	back-light.gif	leading_car	car_brake

button	y position	x position	height	width	action	window	label	
button_1	10	10	60	60		own_display	brakes	

Figure 3: The specifications in *txt2actr* for the positions of the windows, the images and the buttons in ACT-R for a driving environment use case. Further specifications such as which items should appear when and where (depending on the log files) are also possible.

Visual Scene Everything that can be perceived by the cognitive model through its vision module, such as numbers, text, geometric figures, buttons or images from external sources can be specified by their sizes, location and colors (if applicable).

Audio Scene Everything that can be perceived by the cognitive model through its audio module, such as tones, (spoken) words or digits can be specified by their volume, type of tone or duration.

In order to make sure that the information about the environment is displayed at the intended time, the ACT-R interface in *txt2actr* uses the schedule_ time function provided by actr.py. As illustration consider a cockpit environment in an aircraft: In flight, some of the values shown in the cockpit displays in the visual scene will permanently change according to the specifications of a given log file. Additionally, some tone or sound in the audio scene might occur as well. All of the above described components can be specified within the respective text files in *txt2actr*.

Figure 3 shows a screenshot of the environment specification files from a very simplified car driving use case. The full environment specification of this use case can be found in the use-cases/driving-task folder in *txt2actr*.

So far we have not discussed how the model perceives its environment. Thus, possibly even when a tone appears in the model's acoustic scene or a value is shown in one of the windows at a certain time, it might well be that the cognitive model does not perceive it. The model behavior depends on its specification which we will discuss in the next section.

Partial Model Generator

Before we lay out our understanding of the model generator, we introduce the concept of cognitive principles, as they form the foundation for the generation of models.

Cognitive Principles are *cognitively plausible explanations* for some episodes of human behavior, which can be anything from biases, heuristics, judgments or even decision making and reasoning. In particular, cognitive principles are

```
(add-dm
;; the location specification for each item (label) value
(car_normal-info isa display-info name car_normal screen-x 442 screen-y 334)
(car_brake-info isa display-info name car_brake screen-x 442 screen-y 310)
(car_hold-info isa display-info name car_hold screen-x 442 screen-y 286)
(backlight_1-info isa image-info name backlight_1 screen-x 470 screen-y 435)
(backlight_2-info isa image-info name backlight_2 screen-x 860 screen-y 435)
(car_collision-info isa display-info name car_collision screen-x 467 screen-y 694)
(react_emg-info isa display-info name react_emg screen-x 467 screen-y 670)
(time-info isa display-info name time screen-x 732 screen-y 648)
;; the list of items that are to be attended in a routine loop
;; in case of (visual) goal-driven information processing
(car_normal-0 ISA list-info current-on-list car_normal next-on-list car_brake)
(car_brake-1 ISA list-info current-on-list car_brake next-on-list car_normal)
)
```

Figure 4: The environment specifications of Figure 3 reappear as declarative knowledge in the cognitive model.

not necessarily in line with *rational* or *(classical) logical* reasoning, but rather demonstrate naturalistic thinking in everyday life. To some extent, the identification of these psychological phenomena are one of the main motivators in the field of Cognitive Science. As many decades of research show, these psychological phenomena are very insightful for the understanding of the human mind, but at the same time extremely difficult to specify unambiguously during the development of the corresponding model. Cognitive principles are modular and formalized approximations of these phenomena. It is likely that a psychological phenomena can be formalized in various ways, and that there is no agreement on their formal representations. In this case, each of the formalization is an instance of them. We intend to specify a catalogue of cognitive principles, each of them as a module, such that they apply independently of each other. We are aware that it various cognitive phenomena influence each other, or only have an effect when applied together. However, for the goal of benchmarking, i.e. construct a benchmark of cognitive models out of these principles, it is helpful to consider them separately, similarly has has been done for the case of human syllogistic reasoning in (Dietz Saldanha & Schambach, 2020).
Different to the approaches from the previous section, we do not try to identify new cognitive phenomena but rather better understand, formalize and classify already well-established ones. In the first step, cognitive principles are formalized as abstracted and modular entities, and in the second step, during the model construction, they are instantiated with respect to a given task or environment. In the ideal case, a model can be simply specified through the underlying cognitive principles that it assumes.

Modular chunk and production engineering takes place in two steps:

1. Modular specification of the model's properties through cognitive principles.
2. Model construction by instantiating the specification with respect to the environment and the task.

Fortunately, the ACT-R architecture itself consists of a set of modules, which allow us to naturally specify the different components in a modular way. As already mentioned, knowledge can be either represented declaratively, by means of chunks that belong to certain chunk types with a set of slot configurations. For instance,

(**chunk-type display-info name screen-x screen-y**)

is a *display-info* that might have the slots *name*, *screen-x*, and *screen-y*. A chunk can then be understood as an instantiation thereof, such as

(ALTITUDE-info isa **display-info**
name ALTITUDE **screen-x** 389 **screen-y** 514)

tells us at which coordinates we can find the current altitude. The automatic construction of the initial chunk types and chunks is based on our assumption that the model has some basic knowledge about the environment it interacts with. By default, there will be four different chunk types: display-info, button-info, image-info and sound-info. These chunks are derived from the task specification and can be used to model familiarity with a task environment such as knowledge about positions and functions of buttons in an aircraft cockpit. Consider again the environment specification files in Figure 3: If not specified otherwise in the cognitive model specification file, then *txt2actr* will automatically include these items and their coordinates into the declarative knowledge of the cognitive model.
Figure 4 shows a screenshot of the declarative knowledge from the very simple car driving use case.

Two Cognitive Processes in three Environments

txt2actr also allows to specify initial *procedural knowledge*, which in ACT-R is done by means of production rules. These production rules can be read as conditional statements, where, in case the condition holds, the consequence will be executed. The automated construction of an initial set of production rules is more complex, in particular if aiming at constructing them independent of the task. Therefore, we will illustrate the modularity of text2actr by implementing two cognitive processes from information processing and test them in three different environments.

Situation Awareness We will consider two essential processes for modeling situation awareness (Freiman, Myers, Caisse, Halverson, & Ball, 2019). Situation awareness describes to which extent someone has perceived and understood vital elements of a situation for completing the task at hand (Endsley, 2015). According to Endsley (2015) and Freiman et al (2019), a good model of situation awareness needs to account, among others, for the alternation between data-driven and goal-driven information processing, which both can be understood as two distinctive cognitive principles: In the case of *data-driven information processing*, visual attention should be guided by changes in the environment, while for *goal-driven information processing* the model actively engages in search of specific information by means of coordinates. In both cases, the model keeps an updated representation of the values of the attended items. Figure 5 illustrates this idea by the cockpit environment: Goal driven

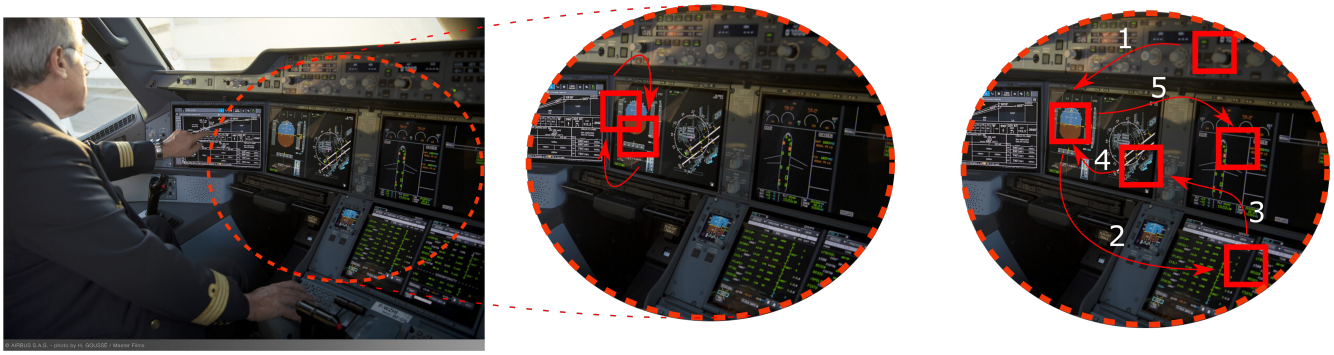

Figure 5: A real cockpit (left), and two ways on how visual information could be attended (middle and right), where the red squares and the arrows denote a possible sequence in which locations are attended.

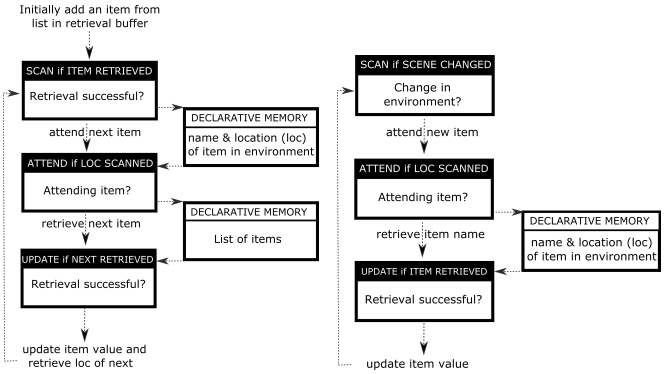

Figure 6: The goal-driven (left) and the data-driven information processing (right) components in ACT-R. Each box with a black header represents a production rule.

information processing (middle) could be where the visual attention alternates between two locations. An example of data-driven (or event-driven) information processing (right) would be where the visual attention is determined by any newly appearing item on the displays (the arrows denote a possible sequence of attention).

Both cases of information processing are implemented as abstracted modular components in *txt2actr*, each represented by a set of general production rules. Additionally, for the goal-driven case, the list of the items to be attended (based on the text files) is automatically generated and added to the declarative memory. An example of such a list for the car driving environment is shown in the last two lines of Figure 4. In case one or both components are chosen to be part of the model, *txt2actr* will create a model with these component(s) instantiated with respect to the specified environment.

The lisp files of both model components in ACT-R with the respective names (data-driven.lisp and goal-driven.lisp) are part of *txt2actr*.[6] Figure 6 shows a description of the processes: Initially, for the goal-driven information processing (left), an item from the list is set into the retrieval buffer. When the retrieval of the current item's location (first production rule) was successful, then this location is attended and the next item on the list is retrieved. Finally, the current item's value is updated, and process starts again by retrieving the location of the next item. In case of data-driven information processing (right), the first rule only fires when the scene changes, for example when new values appear in the visicon. The new item is attended and based on its location, its name is retrieved. Finally, similar to the previous case, this item's value is updated in the model's memory. These model components are not use case specific and thus generally applicable.

For testing whether both model components would behave as intended, both individually and together, we have chosen and specified three very simple but different environments. These three environments were specified exclusively through the provided text files in *txt2actr* (which are contained in the folder environment-specification of each use case). The first one is the *paired associates task* (Anderson, 1981) where the environment and model specification are adapted from the ACT-R tutorial.[5] Note that we intentionally chose for a task in which the model has a different purpose in order to observe how the model behaves. The second and third examples are about the simulation of real-world scenarios: A driving environment and a cockpit environment. Different than in the first example, the values in their environment continuously change according to the log files. Furthermore, the model does not have any other task to accomplish except of updating values in its own memory according to the two cases of the above described visual information processing. The driving environment is built on data from an empirical study originally by Haufe et al (2011) and takes processed extracts of datasets from BNCI Horizon as log file input.[7] The cockpit environment takes as log files extracts from Dashlink.[8,9]

[6] https://github.com/eadietz/txt2actr/tree/master/benchmarks/model-components

[7] The used dataset (VPae.mat) can be found here: http://bnci-horizon-2020.eu/database/data-sets (30.4.21)

[8] https://c3.nasa.gov/dashlink/projects/85/ (30.4.21)

[9] The dataset can be found here https://c3.nasa.gov/

Figure 7: Simulation of paired-associates task (left), driving environment (middle) and cockpit environment (right) in ACT-R. The red dot in the middle and right image shows the model's visual focus.

Observations Figure 7 shows a screenshot of the simulation of each environment. In the driving environment, different values change continuously, while in the paired-associates task, only one value changes, and this happens only occasionally. Therefore, naturally the data-driven information processing is fired more often in the driving environment than in the paired-associates task and even more often in the cockpit environment. Interestingly, in the compound model consisting of both visual processing components the production rules of the data-driven information processing do not apply anymore. Only by the specification of high utilities for this component the production rules of both components apply.

This leads us to the more general question of how such a compound model of visual processing should behave. On the one hand the model should be able to pursue goal-driven visual behavior while being sensitive to new stimuli that compete for visual attention. When goal-driven behavior does not occupy all buffers, these buffers are available to being used by salient, not goal-related stimuli which can lead to distractions and mind wandering (Taatgen et al., 2021). While utility functions can help modeling commitment to goals or susceptibility to distraction, we believe that modeling of attentional control should recognize the interplay of cognitive resources and the environmental factors such as the salience of stimuli (e.g., alert sounds in the cockpit).

It is very likely, that other (ACT-R) modelers would have implemented the above components differently in ACT-R or even diverge from the processes shown in Figure 6. The novelty of our approach is not to demonstrate that the proposed components are the most cognitively plausible ones, but rather that we can build abstracted modular entities of these components, which can be instantiated with respect to different environments.

Conclusions

This paper proposes a possible path for benchmarking cognitive models. Yet, we are far from providing a solution but rather show how a very specific cognitive phenomenon might be applicable to different environments. Already modeling concrete aspects of visual information processing in ACT-R leads to plenty of choice points on the implementation side that are not specified in the theories of situation awareness. However, we believe that taking the effort to approximate theories by implementing instances thereof as models could be beneficial to identify under specifications that might not be immediately obvious.

Our contributions with *txt2actr* are two-fold: First, the specification of an ACT-R environment can be done through text files. Second, we have shown that it is possible to formulate abstracted entities of cognitive phenomena from which model components can be automatically generated. However, we are very aware that this process needs to be built with care and based on more objective criteria for cognitive plausibility or consensus. Therefore, we also need to find a more accessible way of individually assembling cognitive models, possibly guided by a catalog of cognitive principles usable for modular and guided model construction. Currently, we only consider the (partial) automation of initial chunk types, initial chunks and initial productions. The general parameter settings, additional chunk types, chunks, productions or other commands can be specified manually via a text file. For the future, a systematic account on producing general parameter settings might be considered as well. We also argue that some choices (or strategies) do not need to be taken before modeling a task, but can be taken at a later stage (e.g. evaluation of different strategies by instantiating different models for the same task), should be able to include different theories and allow a systematic comparison between modeling approaches.

Finally, we believe that establishing benchmarks will promote (i) the competition among models with respect to the most typical tasks in cognitive psychology, and (ii) the evaluation of (possibly new) tasks with respect to benchmark models. This might help the community to address the previously mentioned issues and eventually unify the field.

`dashlink/static/media/dataset/Tail_687_9.zip` (30.4.21)

References

Amant, R. S., McBride, S. P., & Ritter, F. E. (2006). AI support for building cognitive models. In *Proceedings of the twenty-first national conf. on artificial intelligence and the eighteenth innovative applications of artificial intelligence conf.* (pp. 1663–1666). AAAI Press.

Amant, R. S., & Ritter, F. E. (2004). Automated goms–to–act-r model generation. In *Proceedings of the sixth international conf. on cognitive modeling (iccm)* (pp. 28–34).

Anderson, J. R. (1976). *Language, memory, and thought / john r. anderson*. L. Erlbaum Associates ; distributed by the Halsted Press Division of Wiley Hillsdale, N.J. : New York.

Anderson, J. R. (1981). Interference: The relationship between response latency and response accuracy. *Experimental Psychology: Human Learning and Memory*, *7*, 326-343.

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

Borst, J. P., & Anderson, J. R. (2017). A step-by-step tutorial on using the cognitive architecture act-r in combination with fmri data. *Mathematical Psychology*, *76*, 94–103.

Bothell, D. (2020). *ACT-R 7.21$^+$ Reference Manual.* (http://act-r.psy.cmu.edu/actr7.x/reference-manual.pdf, accessed on 22.4.21)

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. CRC.

Dietz Saldanha, E., & Schambach, R. (2020). A computational approach for predicting individuals' response patterns in human syllogistic reasoning. In S. Denison, M. Mack, Y. Xu, & B. C. Armstrong (Eds.), *Proceedings of the 42th annual meeting of the cognitive science society.* cognitivesciencesociety.org.

Endsley, M. R. (2015). Situation awareness misconceptions and misunderstandings. *Cognitive Engineering and Decision Making*, *9*(1), 4-32.

Freiman, M., Myers, C. W., Caisse, M., Halverson, T., & Ball, J. T. (2019). Assessing cognitive fidelity in a situation awareness process model. In G. L. Rogova, N. M. McGeorge, O. E. Gundersen, K. Rein, & M. D. Freiman (Eds.), *IEEE conf on cognitive and computational aspects of situation management, cogsima 2019* (pp. 100–106). IEEE.

Gebser, M., Maratea, M., & Ricca, F. (2020). The seventh answer set programming competition: Design and results. *Theory and Practice of Logic Programming*, *20*(2), 176–204.

Haufe, S., Treder, M. S., Gugler, M. F., Sagebaum, M., Curio, G., & Blankertz, B. (2011, jul). EEG potentials predict upcoming emergency brakings during simulated driving. *Neural Engineering*, *8*(5), 056001.

Khemlani, S., & Johnson-Laird, P. (2012). Theories of the syllogism: A meta-analysis. *Psychological bulletin*, *138 3*, 427-57.

Laird, J. E. (2012). *The soar cognitive architecture*. The MIT Press.

Laird, J. E., Lebiere, C., & Rosenbloom, P. S. (2017, December). A Standard Model of the Mind: Toward a Common Computational Framework across Artificial Intelligence, Cognitive Science, Neuroscience, and Robotics. *AI Magazine*, *38*(4), 13.

Lebiere, C., Pirolli, P., Thomson, R., Paik, J., Rutledge-Taylor, M., Staszewski, J., & Anderson, J. R. (2013). A functional model of sensemaking in a neurocognitive architecture. *Computational Intelligence and Neuroscience*, *2013*, 921695:1–921695:29.

Lewandowsky, S. (1993). The rewards and hazards of computer simulations. *Psychological Science*, *4*(4), 236-243.

Marewski, J. N., & Mehlhorn, K. (2011). Using the act-r architecture to specify 39 quantitative process models of decision making. *Judgment and Decision Making*, *6*(6), 439-519.

The nature and transfer of cognitive skills. (2013, 06). *Psychological review*, *120*, 439-471. doi: 10.1037/a0033138

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In *Visual information.* New York: Academic Press.

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

Ragni, M. (2020). Artificial intelligence and high-level cognition. In P. Marquis, O. Papini, & H. Prade (Eds.), *A guided tour of artificial intelligence research* (Vol. 3, pp. 457–486). Germany: Springer.

Ragni, M., Riesterer, N., & Khemlani, S. (2019). Predicting individual human reasoning: The precore-challenge. In A. K. Goel, C. M. Seifert, & C. Freksa (Eds.), *Proceedings of the 41th annual meeting of the cognitive science society, cogsci 2019: Creativity + cognition + computation, montreal, canada, july 24-27, 2019* (pp. 9–10). cognitivesciencesociety.org.

Riesterer, N., Brand, D., & Ragni, M. (2020). Predictive modeling of individual human cognition: Upper bounds and a new perspective on performance. *Topics in Cognitive Science*, *12*(3), 960-974.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, *48*(2), 362-380.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science*, *37*(5), 829–860.

Taatgen, N., & Anderson, J. R. (2010). The past, present, and future of cognitive architectures. *Topics in Cognitive Science*, *2*(4), 693-704.

Taatgen, N., van Vugt, M. K., Daamen, J., Katidioti, I., Huijser, S., & Borst, J. P. (2021). The resource-availability model of distraction and mind-wandering. *Cognitive Systems Research*, *68*, 84-104.

Thomson, R., Lebiere, C., Anderson, J. R., & Staszewski, J. (2015). A general instance-based learning framework for studying intuitive decision-making in a cognitive architecture. *Applied Research in Memory and Cognition*, *4*(3), 180-190. (Modeling and Aiding Intuition in Organizational Decision Making)

West, R., & Nagy, G. (2007). Using goms for modeling routine tasks within complex sociotechnical systems: Connecting macrocognitive models to microcognition. *Cognitive Engineering and Decision Making*, *1*(2), 186-211.

West, R., & Somers, S. (2011). Scaling up from micro cognition to macro cognition: Using sgoms to build macro cognitive models of sociotechnical work in act-r. *Cognitive Science*, *33*.