Integrated Cognition
AAAI Technical Report FS-13-03

The Gap between Architecture and Model: Strategies for Executive Control

Niels A. Taatgen
Institute of Artificial Intelligence, University of Groningen
Nijenborgh 9, 9747 AG Groningen, Netherlands
n.a.taatgen@rug.nl

Abstract

One major limitation of current cognitive architectures is that models are typically constructed in an "empty" architecture, and that the knowledge specifications (typically production rules) are specific to the particular task. This means that general executive control strategies have to be implemented for each specific model, which means a lack of consistency and constraint. Alternatively, they are implemented as part of the architecture itself, which is often implausible, because strategies are learned and differ among individuals. The alternative is to assume executive control consists of strategies that can transfer from one task to another. The PRIMs theory (Taatgen 2013) provides a modeling framework for this transfer. The approach is discussed using the example of working memory control.

The Goals of Cognitive Architectures

The enterprise of cognitive architectures is connected to grand ambitions. Both Newell (1990) and Anderson (1983) proposed the cognitive architecture as the great unifier in cognitive science and cognitive psychology. To underline these ambitions, Newell proposed a list with 13 items in his 1990 book that cognitive architectures should strife to accomplish. I will not reiterate the list here, but it contains items such as "Behave robustly in the face of error", "Use vast amounts of knowledge", up to "Arise through evolution".

By the end of the book Newell revisits the list, and concludes 6 of the goals have been satisfied (at least in some fashion). Anderson and Lebiere (2003) revisit the list, calling it the *Newell Test*, and conclude their ACT-R architecture satisfies 5 out of 12 criteria, pitting it against connectionism, which they also rate at 5 out of 12 (although on different items). Despite the seeming lack of progress on the Newell test scale, cognitive architectures have made great strides in providing explanations for a wide variety of cognitive phenomena. Just looking at the huge body of publications on the ACT-R website (act-r.psy.cmu.edu) shows that almost any area of experimental psychology has been covered. As a consequence, cognitive architectures have been successful in being published in mainstream cognitive science and psychology journals, and play a role in the development of cognitive theory. However, because of the alignment with standard practice some of the original goals have been neglected, which may be why progress on the Newell Test has been limited, because it has more ambitious goals than cognitive psychology generally pursues.

A typical modeling paper discusses a particular phenomenon using experimental data and a model of these data, and pushes a particular explanation or theoretical position. Although this is much better when it is compared to verbal explanations, the general research strategy is one that at least Newell argued against strongly (Newell 1973). According to him, psychology would never make progress in understanding cognition if it would persist in studying single phenomena using a research paradigm that he mockingly called "The Twenty Questions Game". He pointed out that pursuing this "Game" would be a fine choice to advance your career as a scientist, but would, in the end, not advance science. Unfortunately, it seems even cognitive modelers have partially fallen into this trap. The main problem is that models are constructed for particular experimental tasks or paradigms, but that generalization from one model to another is often very limited. Part of this problem lies in the current cognitive architectures themselves, because they promote thinking about cognition in terms of separable tasks.

Task-specific Models

The construction of a cognitive model for a particular task starts with an "empty" cognitive architecture. The modeler adds knowledge components that are specific to that task, or, in the case of learning models, trains the model using items specific to that task. The underlying assumption is that a real cognitive system already has a large body of knowledge about other tasks, but this knowledge has no or limited influence on the new task, and can therefore safely be omitted.

As a consequence, the common element between different models within a particular cognitive architecture is just the architecture itself: its representations, it mechanisms to handle knowledge, and its modules to interact with the outside world. The assumption of these architectural elements is that they are the same, no matter what the task is. Even stronger, there is the assumption that all architectural components are innate, because they are properties of the brain itself.

This leaves a gap for a third component: knowledge that is not part of the architecture, but also not task-specific. In particular I want to focus here on procedural knowledge, because the importance of more factual general knowledge has already been acknowledged and discussed elsewhere (e.g., Salvucci 2012). A first question concerning general procedural knowledge is one of transfer: if we already know a particular text editor, is it easier to learn a new one? The answer is yes (Singley & Anderson 1985). It is also quite plausible that skills build upon each other. For example, it is easier to learn multicolumn subtraction after learning multicolumn addition first, even though the two differ enough to require separate specific rules.

But is there reuse of procedural knowledge beyond tasks that resemble each other? One domain to look at in more detail is what is generally referred to as *executive control*. This includes the handling of goals and tasks, organization of working memory, activating relevant information and suppressing irrelevant information, interleaving multiple tasks, and handling interruptions, among others. There is always a certain awkwardness in how cognitive architectures handle executive control. It is a challenge to make control part of the architecture, because this implies it operates in the same way in every possible situation, which it often does not. On the other hand, if control is part of the (task-specific) model, control feels ad-hoc, and "programmed" by the modeler. Let me give two examples.

Organization of Goals

In production system models, the organization of goals is often handled by a goal stack. Although it is a convenient mechanism that works very well in many models, it has several problems. One type of problem is functional. Sometimes problems are not suitable for goals stacks, in tasks where goals are created and abandoned often, requiring great efforts in breaking down and rebuilding the goal stack. A second problem is behavioral plausibility: the human cognitive system does not have a perfect goals stack. For example, both Altmann and Trafton (2002) and Anderson and Douglass (2001) have shown that errors people make in the iconic goal-stack task, the Towers of Hanoi, are not consistent with an architectural goal stack. Instead, the handling of goals in their alternative models uses a strategy that partially resembles the goal stack, but that is part of the task model. Although this solution is more satisfactory in the sense that it accounts much better for the empirical data, it is also implausible that the handling of goals has to be reinvented for every single new task.

Working Memory Control

A second example is working memory control. There is an ongoing debate in cognitive psychology about the nature of working memory, but researchers almost all take an architectural stance. Nevertheless, many aspects of working memory are not part of the architecture. For example, the use of rehearsal to maintain elements in memory is unlikely to be architectural, because young children do not use rehearsal. Rehearsal is a strategy that can be used in many different situations, so it is definitely not specific to a particular model. There are several theories that propose that the actual working memory capacity is quite limited (Borst, Taatgen & van Rijn 2010; Oberauer 2002). Such a limited capacity requires strategies to have the right information available at the right time. For example, Borst er al. (2010) assume information in working memory is swapped out strategically with long term (declarative) memory, a process that is not architectural, but also not specific to the particular task, even though current models treat it as such (see also Altmann & Trafton 2002).

To summarize, many aspects of executive control consists of strategies rather than mechanisms. Research that further underlines this are studies of far transfer. For example, in a study by Karbach and Kray (2009) subjects are trained on task switching for several days. This not only improves their skill at task switching, but also reduces Stroop interference, increases working memory capacity, and improves the score on the Raven progressive matrices test. Apparently, something is trained that is not architectural (otherwise it wouldn't change), but also not particular to task switching.

To be able to model general task strategies, including executive control strategies, I have developed the primitive elements of information processing (PRIM) theory.

The PRIM theory

The PRIM (primitive information processing elements) theory (Taatgen 2013) is an extension of the ACT-R architecture (Anderson 2007). The key characteristic of the theory is that production rules are broken down into elementary processing elements, separating task-specific constants from task-general mappings. There are three types of elements (PRIMs), and they all operate on slots in ACT-R's buffers. The first type of PRIM is a condition consisting of a comparison between two slots (either equality or inequality). Given the number of slots in all the buffers this produces quite a few combinations, which means there are 1188 condition PRIMs. The second type of PRIM copies the contents of one slot to another slot. There are 504 PRIMs of this type. Finally, there is one PRIM that sets specific values (by retrieving them from declarative memory).

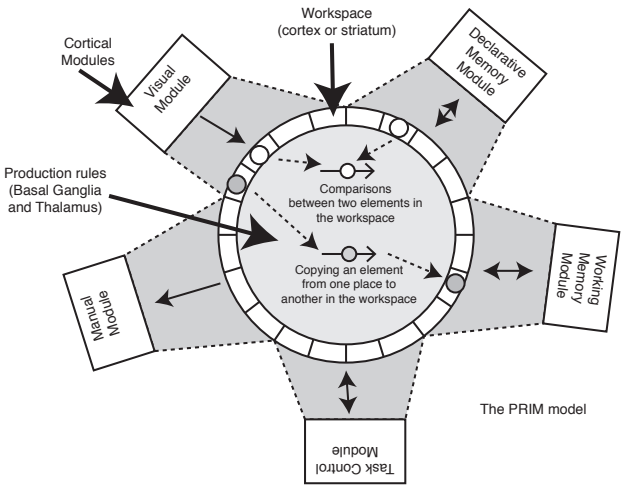

Figure 1. Global workspace/ACT-R buffer model of information processing. From Taatgen (2013). Copyright the American Psychological Association. Reprinted with permission

Figure 1 gives an impression of the role of PRIMs in an ACT-R framework. All the ACT-R buffers together produce a large vector of slots, and all production rules do is compare values in these slots, and move or copy information from one slot to another. The particular modules that are connected to the slots then carry out specific operations on the contents of particular subsets of slots, for example declarative memory and perception and motor slots. PRIMs are like the machine language of cognitive processing: they move around information in a certain way without incorporating any particular aspects of the task involved.

For example, take the following ACT-R production rule:

```
(p simple-count-rule
   =goal>
      isa count-goal
      state retrieve
   =imaginal>
      isa counter
      count =count
==>
   +retrieval>
      isa count-fact
      num1 =count
   =goal>
      state waiting)
```

If the goal is a count-goal in a state where we need to retrieve the next count fact
AND the counter has value *count*
THEN
Start retrieving a count-fact that specifies the number after *count*
AND set the goal state to waiting

This rule is part of a counting model, which initiates a retrieval of a count-fact from memory.
This rule can be broken down into the following five PRIMs:

Specific Value PRIM

- Set *item1* to retrieve, *item2* to count-fact, and *item3* to waiting, by retrieving these values from declarative memory

Condition PRIM

- The first slot in the goal should be equal to *item1*

Action PRIMs

- Copy *item2* to the first retrieval slot
- Copy the first slot in the imaginal to the second retrieval slot
- Copy *item3* to the first slot in the goal

In other words, even a simple count rule can already be broken down into five primitive elements. Note that the PRIMs do not refer to slot names, but to the position of a slot in the buffer. This is important, because the general knowledge should be applicable to many different types of knowledge, and specific slot names would be a hindrance. It also means that, contrary to ACT-R, the number of slots in a buffer is fixed.

Before learning, PRIMs are carried out one at a time. However, production compilation (Taatgen & Anderson 2002) combines PRIMs into larger units (production rules). As long as these units do not incorporate any specifics (and the implementation tries to postpone this for as long as possible), the combined PRIMs are task-general, and can therefore be reused by any other task that uses the same patterns of information processing. Figure 2 illustrates both the process of compilation and the process of transfer. Arrows with a single circle in it represent PRIMs that only carry out one elementary step, white circles are conditions, grey circles are actions and colored circles are task-specific

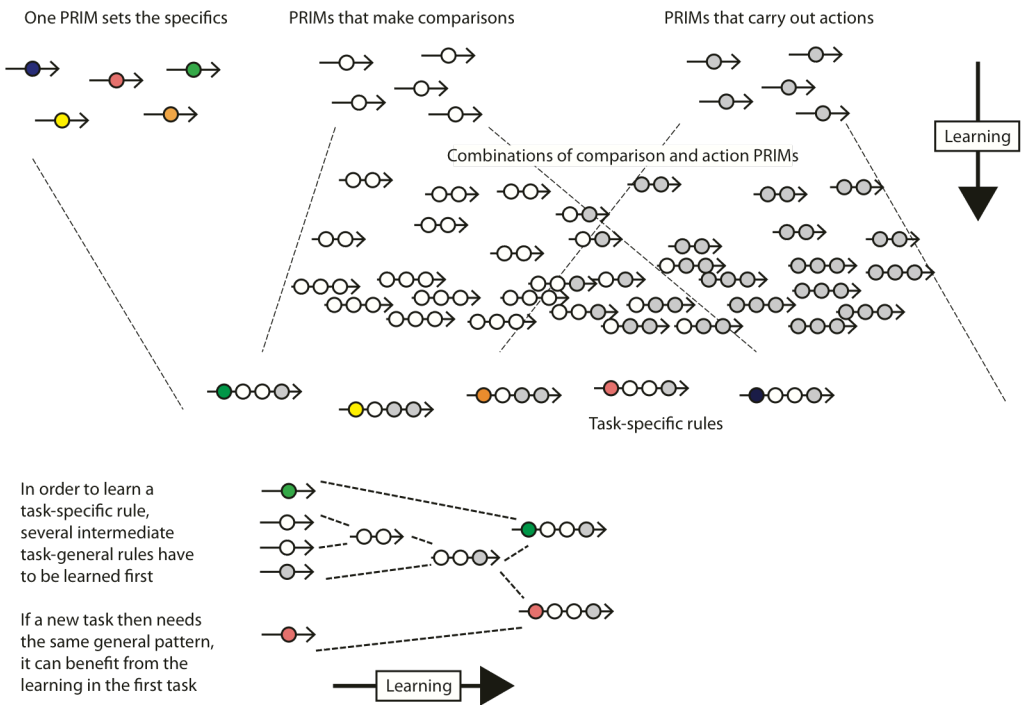

Figure 2. Impression of learning in the PRIM theory. Circles represent PRIMs, and arrows combine PRIMs into larger units. White circles are condition PRIMs, grey circles action PRIMs, and colored circles PRIMs that set specific values. The top figure illustrates how the initial rules with just one PRIM are combined into increasingly larger combinations, eventually leading to task-specific rules (i.e., the rules with a colored circle in them). The bottom figure illustrates transfer: if a number of PRIMs have already been combined in one particular task (the "green" task), it becomes easier to learn a rules for a new task that uses the same PRIMs (the "red" task), because only one compilation step is necessary instead of several.

instantiations of constants. As these PRIMs are used to carry out tasks, the production compilation process combines them into larger units, resulting in rules that carry out multiple PRIM steps in one combined step, as illustrated by arrows with multiple circles. As long as there is not task-specific (colored) component in a rule, it can be used by other tasks. This is illustrated in the bottom part of the Figure: the rule that has two white and one gray PRIM that is learned in the context of the green task can be used by the red task to shorten its learning trajectory.

In order to carry out all the PRIMs in the right order, task knowledge is initially stored in declarative memory. Initial novice behavior is characterized by retrieving references to PRIMs from declarative memory, and carrying them out one at a time. The production compilation process gradually combines the PRIMs, and eliminates the need to retrieve PRIMs from declarative memory.

Details about the PRIM theory can be found in Taatgen (2013).

Transfer in Working Memory Control

In Taatgen (2013), I discuss how anticipation of future events (proactive control) can transfer from one task to another. Here I will present an example of working memory control.

If we assume that actual working memory is limited to one chunk, maintaining multiple items has to involve a strategy in which items are stored in long-term declarative memory in such a way that they can be rehearsed and recalled. In the case in which the material to be remembered is a list of items in which the order is important, it makes sense to also structure these items as a list in the memory representation.

Figure 3 illustrates the process that builds such a list. The strategy consists of three components, with an optional fourth. The first is to initialize the list, the second is to add an item to it, and the third is to recall the items in the list.

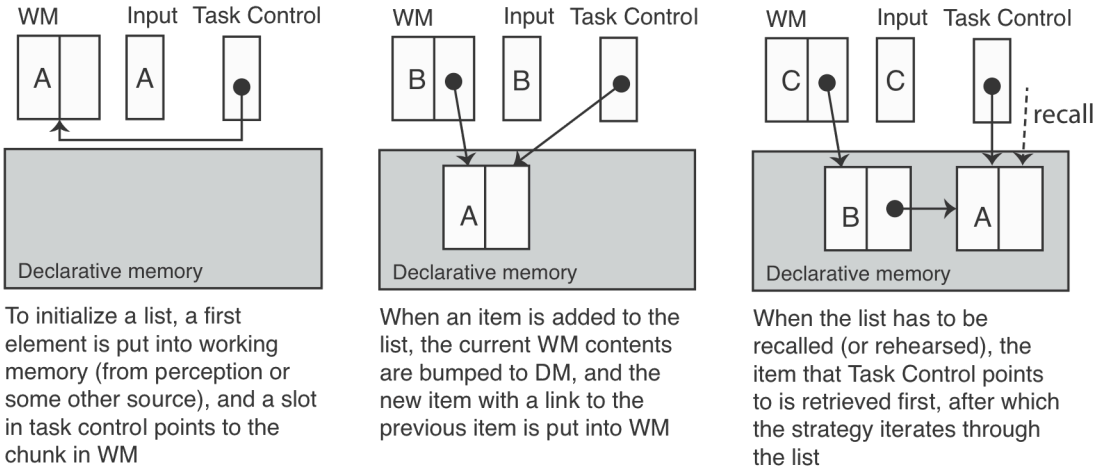

Figure 3. How the list memory strategy builds a representation in memory

The optional fourth is to rehearse the list, but this is more or less the same as recalling it, so I will not discuss it here.

If we would implement models using this strategy in an architecture like ACT-R, the result would be that each model would have their own task-specific version of the strategy above, and there would be no interaction among these models. When the models would be implemented using PRIMs, however, the parts of the model that involve list memory would overlap. Therefore, learning one task that uses list building and rehearsal would also benefit other tasks that use it.

To illustrate this, I implemented three such tasks using PRIMs. The first two tasks are both complex working memory tasks from Kane et al. (2004), and were already discussed in Taatgen (2013). In the Countspan task, subjects have to count objects in several consecutive screens, but then have to remember these counts because they have to be reported as a list after the last screen. In the Verbal-CWM task, subjects have to do a lexical decision task (deciding whether a string of letters is a word or a non-word), but sometimes a single letter appears. After a number of these presentations, subjects have to recall the sequence of presented single letters.

The third task is an Attentional Blink task (implemented in a regular ACT-R model by Taatgen, Juvina, Schipper, Borst & Martens 2009), in which a rapid stream of characters is presented on the screen (at a pace of 100 ms/character). Characters can be either letters or digits, and subjects are instructed to remember the letters.

Figure 4 illustrates the PRIMs involved in storing and recalling items from memory for the three tasks. Each of the nodes in the graph corresponds to a PRIM: colored nodes to task-specific PRIMs, and white and grey nodes to task-general PRIMs. Each colored node (with the exception of the three nodes with the task names in them) has an arrow to a list of conditions and to a list of actions, which together correspond to operations that are normally carried out by a traditional production rule. However, contrary to production rules, there is substantial overlap between the condition and action lists, in particular in the action lists. The figure illustrates the overlap in the action lists for the three components of list maintenance: initialization, storage and recall. The consequence of this setup is that the three tasks benefit from each other: once one of the three tasks has been mastered, it becomes easier to learn any of the other two.

Discussion

The list-learning example shows that a general skill can emerge out of individual models that all use the same strategy. The advantage is that it does not require any explicit learning of strategies or even explicit generalization. The implication is that once the list-learning strategy is part of the set of strategies the model has mastered, it becomes easier to use this particular strategy than a different list-learning strategy. The advantage over an architectural solution is that it doesn't force particular strategies: using this particular list strategy is just easier than using a different strategy. In other words: existing strategies offer soft constraints for models in terms of learnability.

To fully benefit from this intermediate level between architecture and tasks-specific model, it would be desirable

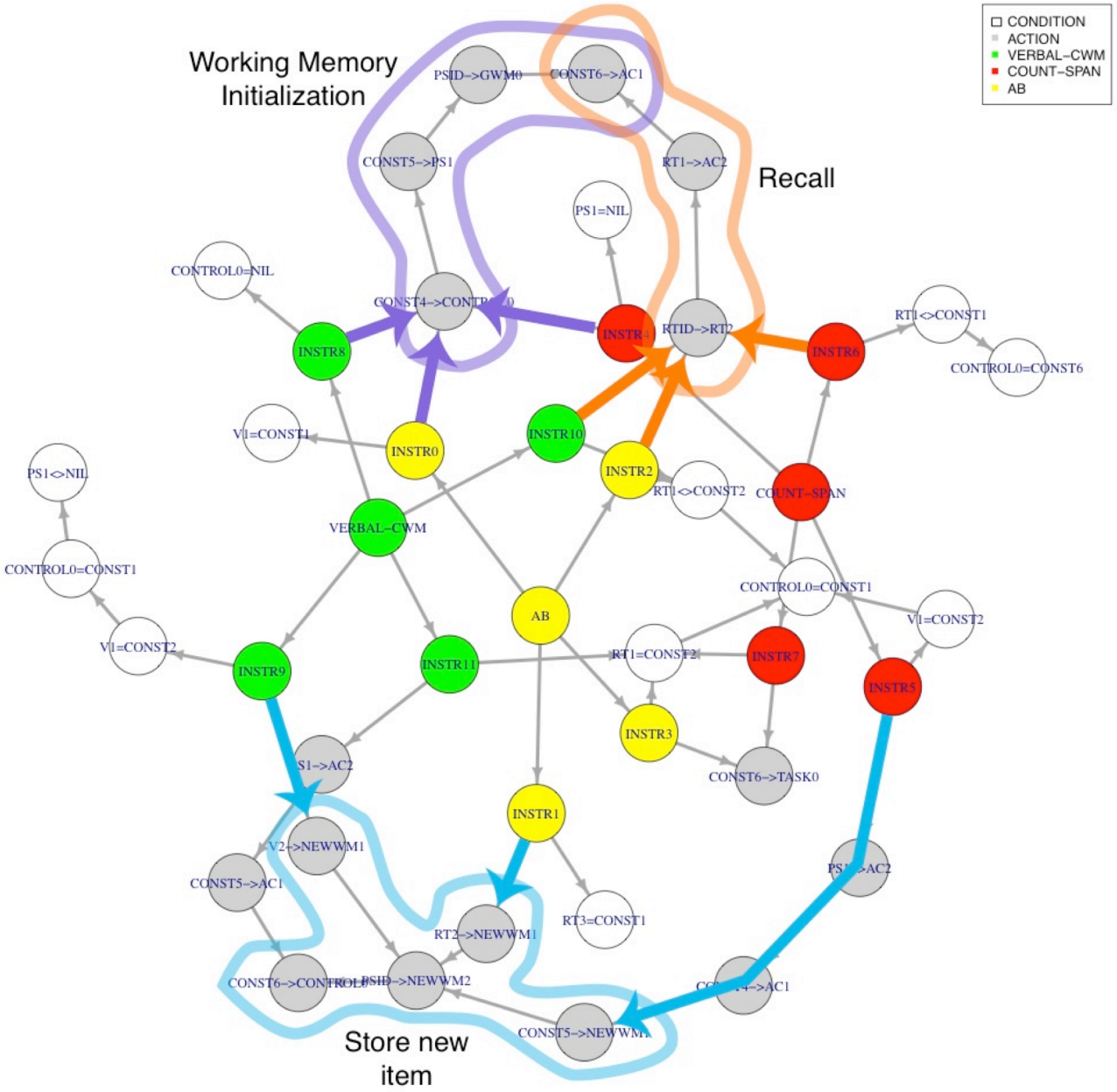

Figure 4. Structure of PRIMs in declarative memory in three tasks that use a list representation for storing and recalling items. Each of the colored nodes is a Specific Value PRIM that is specific for the task (Verbal-CWM, Countspan and Attentional Blink). Only PRIMs that are part of the storing and recalling of items are in the figure to reduce clutter. Specific Value PRIMs each have an associated list of condition PRIMs (white nodes), and action PRIMs (grey nodes) that overlap if they are the same. In this case, there is a strong overlap in action PRIMs in all three aspects of handling lists: initialization, storage and recall.

to train architectures with initial task strategies. This can be accomplished by priming models with existing models of tasks of which it is plausible that they are part of most people's cognitive repertoire. Each additional model would contribute to a better network of general cognitive skills, and increase the plausibility of subsequent models.

Although the PRIM model is based on ACT-R, the general approach is not limited to ACT-R. The core ideas,

breaking down procedural knowledge into primitives and separating task-general from task-specific elements, can be implemented in different ways and different architectures. The PRIM model in particular does require use a number of ACT-R assumptions and mechanisms: production rules that can only inspect information in a fixed set of buffers, the production compilation process and both a declarative and a procedural memory.

Acknowledgements

This research was supported by ERC StG 283597 MULTITASK from the European Research Council.

References

Altmann, E. M., and Trafton, J. G. 2002. Memory for goals: an activation-based model. *Cognitive Science* 26:39-83.

Anderson, J. R. 1983. *The architecture of cognition*. Cambridge, MA: Harvard university press.

Anderson, J. R. 2007. *How can the human mind occur in the physical universe?* New York: Oxford university press.

Anderson, J. R., and Douglass, S. 2001. Tower of Hanoi: evidence for the cost of goal retrieval. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 27:1331-1346.

Anderson, J. R., and Lebiere, C. 2003. The Newell Test for a theory of cognition. *Behavioral and Brain Sciences* 26:587-637.

Borst, J. P., Taatgen, N. A., and van Rijn, H. 2010. The Problem State: A Cognitive Bottleneck in Multitasking. *Journal of Experimental Psychology-Learning Memory and Cognition* 36:363-382.

Kane, M. J., Hambrick, D. Z., Tuholski, S. W., Wilhelm, O., Payne, T. W., and Engle, R. W. 2004. The Generality of Working Memory Capacity: A Latent-Variable Approach to Verbal and Visuospatial Memory Span and Reasoning. *Journal of Experimental Psychology: General* 133:189-217.

Karbach, J., and Kray, J. 2009. How useful is executive control training? Age differences in near and far transfer of task-switching training. *Developmental Science* 12:978-990.

Newell, A. 1973. You can't play 20 questions with nature and win. In W. G. Chase, ed., *Visual information processing*. New York: Academic Press.

Newell, A. 1990. *Unified theories of cognition*. Cambridge, MA: Harvard university press.

Oberauer, K. 2002. Access to information in working memory: exploring the focus of attention. *Journal of Experimental Psychology: Learning, Memory and Cognition* 28:411-421.

Salvucci, D.D. 2012. A large-scale knowledge base for ACT-R. Presentation at the nineteenth annual ACT-R workshop.

Singley, M. K., and Anderson, J. R. 1985. The transfer of text-editing skill. *Journal of Man-Machine Studies* 22:403-423.

Taatgen, N. A. 2013. The nature and transfer of cognitive skills. *Psychological Review* 120:439-471.

Taatgen, N. A., and Anderson, J. R. 2002. Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition* 86:123-155.

Taatgen, N. A., Juvina, I., Schipper, M., Borst, J., and Martens, S. 2009. Too much control can hurt: a threaded cognition model of the attentional blink. *Cognitive Psychology* 59:1-29.