Obstacles to the Skill-Based Approach: Why is skill reuse so difficult for cognitive architectures?

Corné Hoekstra (c.hoekstra@rug.nl)
Bernoulli Institute for Mathematics, Computer Science, and Artificial Intelligence
University of Groningen, Groningen, the Netherlands

Sander Martens (s.martens@umcg.nl)
Department of Biomedical Sciences of Cells and Systems
University Medical Center Groningen, University of Groningen, Groningen, the Netherlands

Niels A. Taatgen (n.a.taatgen@rug.nl)
Bernoulli Institute for Mathematics, Computer Science, and Artificial Intelligence
University of Groningen, Groningen, the Netherlands

Abstract

Skill reuse is a commonly accepted aspect of human cognition but it has been difficult to translate to cognitive architectures. We developed the skill-based approach which enables modelers to create models composed of skills created for other tasks but it does not (yet) support fully *reusable* skills. We will discuss three factors that prevent full *reusability*: inflexible WM, rigid goal selection and all-or-nothing condition checking. The factors are discussed in the context of the architecture PRIMs but they also apply to many other cognitive architectures. Finally, we discuss possible solutions to alleviate these issues.

Keywords: cognitive modeling, PRIMs, ACT-R, skill reuse, generalizability, cognitive architecture

Many tasks share considerable overlap in the cognitive elements required to complete it (Lee & Anderson, 2001). This cognitive overlap is one of the key fundamental principles underneath the attempt of cognitive architectures to arrive at a unified theory of cognition. In cognitive architectures the overlap in cognitive elements is put into practice by defining the blank slate cognitive system (i.e., the architecture) consisting of modules and buffers that underlies all behavior (Anderson et al., 2004). This approach has led to successful modeling of a wide range of tasks and paradigms; however, a crucial additional consequence of the cognitive overlap between tasks has never received much attention. Not only can the same architecture be used to complete many tasks, this architecture can also very often be used in the same way (i.e., with the same procedural knowledge). Incorporating this into cognitive architectures would take into account the fact that huge proportions of our capabilities have been acquired through a long process of development and learning while currently only the innate aspects of cognition are considered. In order to bring this idea into practice, we have developed the skill-based approach to cognitive modeling.

This approach can be valuable for multiple reasons. Firstly, models will mirror human behavior more closely which will improve model fit (Stearns & Laird, 2018). Secondly, reusing procedural knowledge is a large contributor to the flexibility people possess in executing various tasks. Incorporating it into cognitive modeling and AI could strongly improve flexibility and robustness (Taatgen, Huss, Dickison, & Anderson, 2008). Finally, the large range of models created in the different fields of cognitive science can be integrated more easily if they all draw from one pool of basic building blocks.

PRIMs

We have explored the idea of skill reuse in the cognitive architecture PRIMs (Taatgen, 2013). We will give a short introduction to PRIMs here and in the relevant sections further down the paper. (See Taatgen (2013) for a complete introduction). PRIMs is based on ACT-R and inherits many of its properties. It is a cognitive architecture built up from distinct cognitive modules whose actions are controlled by "production-rules" (operators in PRIMs) and it contains a similarly functioning declarative memory system. An important distinction between the two architectures is that the operators in PRIMs are built up from smaller units than ACT-R's production rules. These smaller units are the **prim**itive information processing element**s** (PRIMs). PRIMs are considered the basic elements of cognition and are only capable of either moving or comparing pieces of information in the workspace. Although a single PRIM is not very powerful, combinations of PRIMs (i.e., operators) are able to execute complex cognition on the same level as ACT-R. These primitive operations are assumed to be universally applicable to any task and therefore can provide low-level mechanisms of transfer. They are also relatively easy to implement in neural architectures (Stocco, Lebiere, & Anderson, 2010). The central concept of the skill-based approach, a skill, is one level above an operator. A skill is a reusable collection of operators that perform a part of a task. Although a skill is larger than an operator, carrying out a skill still only takes a small amount of time in the order of one second or less.

The low-level transfer combined with the higher-level concept of a skill make PRIMs well-suited for

exploring the skill-based approach although (most of) its principles can be implemented in other cognitive architectures as well.

The Skill-Based Approach

The central idea of the skill-based approach is to construct models of tasks in the same way humans would approach a new task. When people are confronted with a new task, they do not need to figure out from scratch how to complete this task but instead can rely on previously learned knowledge which has proven successful (Salvucci, 2013). A good example of this are the experimental tasks typical of cognitive psychology. Participants have usually never encountered these tasks before, yet they are quickly able to figure out what to do. Since they do not have time to learn new procedural knowledge specific to this task, it suggests that they reuse existing procedural knowledge. Concretely, the skill-based approach assumes that learning (almost) any new task merely means composing it from already existing skills.

A fundamental challenge to emulating this human-like flexible behavior in cognitive models is balancing generalizability with accuracy. Different tasks come with different contexts and the model needs to be general enough to function in all these contexts but also specific enough to produce the same result regardless of that context. The common solution to this challenge is to allow for dynamic variable binding (Greff, van Steenkiste, & Schmidhuber, 2020); that is, allow variables to take on different values depending on the context. Although this solution is commonly adopted across different types of AI, there is no consensus on how it should be implemented (Feldman, 2013). The solution adopted by PRIMs is variable instantiation; a skill is created with general variable names which are only defined (instantiated) when the skill is used in a new context. However, there is no principled way in which this mechanism is implemented in the architecture.

More exact details can be found in our previous publications on the skill-based approach in which we propose the method (Hoekstra, Martens, & Taatgen, 2020) and test the validity of its predictions (Hoekstra, Martens, & Taatgen, 2022), but in short the skill-based approach works as follows. The first step of the skill-based approach is determining which basic skills are responsible for performing the modeled task based on previous literature. This step comes forth out of the fundamental principle of the skill-based approach that every task is a composition of basic processing steps that have been done (many times) before. For example, in the attentional blink (Martens & Wyble, 2010) model we have constructed (Hoekstra et al., 2020), the four basic skills we included were 'visual search', 'consolidation', 'retrieval', and 'response'. Skills that were reused from other models. This first step increases the generalizability of a model because the ubiquity of its basic building blocks allows it to be easily linked to other models and theories. The second step involves creating and testing the validity of the basic skills. In this step, other models which include (some of) the basic skills are built and these models are compared with human data. In our attentional blink model, we completed this step by creating a model of a simple visual discrimination task and two working memory tasks (a simple working memory task and a complex working memory task). This step is necessary to create the basic skills and it provides evidence for the accuracy of these skills. The final step involves adapting the basic skills to the context of the task of interest. In PRIMs, the cognitive architecture we used, this is done by instantiating the skills.

Following this method, we succeeded in constructing a model of the attentional blink (AB) that consisted of elements (skills) that worked in both the original task (e.g., the complex working memory task) as well as the AB task. This shows that it is possible to create cognitive models out of elements created for other tasks and that models can be created by merely assembling already existing procedural knowledge. However, the process of creating these skills was quite laborious and it often required making modifications to the basic skills that seemed too "AB-specific" to be part of general basic skills (Hoekstra et al., 2020). In short, we succeeded in creating a model with *reused* skills but not with fully *reusable* skills. That is, we managed to create an AB model out of skills that are also parts of other models (and are therefore *reused*) but these skills cannot be freely reused in every other task that includes the same basic skill (i.e., they are not fully *reusable*). However, this is crucial; making the step from *reused* skills to *reusable* skills would realize the full potential of the skill-based approach. It would standardize the knowledge used in cognitive models as well as increasing the ease with which skill-reuse can be implemented during model building.

Current paper

In the current paper, we will discuss which factors cause the difficulties in creating fully *reusable* skills. We will describe three open questions that complicate the implementation of the skill-based approach, specifically in PRIMs but some also apply to ACT-R. Although these open questions demonstrate practical problems in implementing the skill-based approach, they also point to fundamental unanswered questions about how flexibility should be balanced with cognitive plausibility as well as learnability. The questions will be illustrated by challenges we encountered while using the skill-based approach to model the updating tasks described by Miyake and colleagues (Miyake et al., 2000).

Inflexible Working Memory

In PRIMs and ACT-R the main purpose of working memory (WM) is to keep relevant information quickly available and to support the building of new chunks. WM in ACT-R does not consist of one dedicated system but instead consists of two modules that

together function as WM: declarative memory and the problem state (Nijboer, Borst, van Rijn, & Taatgen, 2016). Declarative memory is responsible for storing chunks while the problem state takes care of keeping the chunks immediately available and is capable of creating new chunks.

In PRIMs, WM does consist of a single dedicated module responsible for keeping information readily available and for creating new (long-term) memory chunks. This module is called the imaginal buffer; however, it is often referred to as the WM-buffer and, for clarity, we will follow that convention. The WM-buffer in PRIMs works as any other buffer in the architecture in the sense that it has slots in which information can be placed and retrieved without any penalty. The slots function independently of one another and are numbered starting with one. Information is placed in and withdrawn from WM by a PRIM. For example, placing information presented on the screen in WM can be done by the PRIM *V1 -> WM1* and information can be taken out from WM by a PRIM such as *WM1 -> AC2*. Information can also be moved around within WM, for example *WM4 -> WM1*. The use of numbered slots in WM makes it much easier to reuse skills and operators compared to using named slots such as in ACT-R. However, it is not flexible enough to facilitate full reusability because the numbered slots are often still too rigid.

The inflexible working memory causes two main issues. The first is that the slots that will be used by the skills in the separate tasks need to be calibrated to work together. This requires a lot of effort from the modeler and although it is manageable for smaller and homogenous models, it quickly becomes unwieldy when the model involves many skills and different types of tasks. This is not a fundamental limitation, but it does present an obstacle to the adoption of the skill-based approach, especially when skill reuse is only a secondary interest. The second issue is more fundamental. Reusability of skills depends on the availability of the WM slots used in the original task. When these slots are not available in a different task, the skill cannot be reused. For example, the 'read' skill in our updating model stores the newly presented item in WM5 because the first four slots are used to keep track of the previously presented items. This might become problematic if the model would move on to a five-item memory task because the WM5 slot will be used to keep track of the fifth item. This illustrates that WM is not flexible enough unless a skill is designed while keeping every possible combination of tasks in mind and that full *reusability* is not yet possible.

Besides causing practical difficulties in using the skill-based approach, the issues with WM also point to a more fundamental question of how WM should be implemented in a cognitive architecture. The challenge is that WM needs to be extremely flexible on the one hand, but also consistent with the limitations that have been identified in the literature on working memory.

The buffer-based design of PRIMs' WM has the advantage of being relatively flexible. It can be used in many types of tasks and it can store many types of information, additionally it provides a means of keeping information readily available. However, it lacks some plausibility because it assumes perfect (decay-free) storage of its contents which is not fully in line with the WM literature.

The alternative to using a buffer for WM is to store items in declarative memory. This is an attractive option, because it puts no hard limit on the number of items, but it still imposes a soft limit through memory decay. However, using declarative memory as WM also has a strong limitation in the sense that the information is not readily available, and has to be retrieved first. Given that items can only be retrieved one at a time, it is impossible to interrelate two or more items, which is a necessity for almost all tasks.

In conclusion, the practical issues we encountered while exploring the skill-based approach not only point to implementation issues but also to fundamental questions of how flexibility and plausibility should be balanced in WM.

Rigid Goal Selection

The goal module plays a central role in determining which production will fire in both ACT-R and PRIMs. Although the goal buffer plays a similar role in both architectures it does not work in the same way. In ACT-R the goal buffer influences production selection through the goal-state chunk present in the goal buffer and exerts its influence in a very explicit manner. Only production-rules which condition side matches the pattern in the goal-state chunk will be considered for selection. This way, the goal module is largely responsible for guiding the model towards firing the right productions at the right time.

The goal module in PRIMs has the same general role and also is responsible for the broad strokes 'supervision' of the model through a task. However, the goal module in PRIMs executes its role in a different and less explicit way. Operator selection in PRIMs is determined by the activity of the operators in memory. The most active operator gets selected first and its conditions are compared to the current context, if the conditions match the context the operator will fire. If the conditions do not match, the next most active operator will be retrieved and its conditions tested. This process repeats until an operator with matching conditions is found which will then fire. The goal buffer has a large influence on this process by spreading activation to operators that are associated with the current goal. This biases the selection process towards selecting operators that match the goal without guaranteeing that such operators will fire (noise or non-matching conditions can still prevent it). The subtle but forceful influence the PRIMs goal module exerts allows for organized behavior while still allowing for flexibility within a task and, importantly, between tasks. The limitation related to the goal

module is not how the goal module impacts operator selection but instead in how the goal itself is selected.

As is the case with all exchanges of information in PRIMs, goals are also determined by a PRIM. A new goal becomes active by a PRIM updating the value in G1 (the first slot of the goal-buffer). Although it is also possible to create situations in which multiple goals are active, for simplicity sake we will focus on a situation with one active goal. Goals are defined by symbols (similar to ACT-R) and therefore setting 'respond' as the goal can be done by the PRIM *respond -> G1*, if there is a skill with that same name. There are no rules about when or how the goal-determining PRIM needs to fire, however the architecture is designed in such a way that the most logical place for such a PRIM is in the final operator of a skill. This is very useful for simple models because it allows for an easy to understand (and flexible) way in which the model moves from one goal to the next. However, it becomes limiting in more complex models, especially in tasks in which the order of the goals is not always the same.

Determining the next skill within the previous skill essentially means that the next goal is decided by the previous goal. This severely limits full reusability of a skill because the role of a skill differs depending on the task. In some tasks, a certain skill might only be used at the end of a task (and therefore would not even require a *next-skill* operator) while in a different task the same skill might be a central part of the task and be used multiple times within a single trial. Switching skills gets further complicated by condition checking (which will be discussed in the next section) because different conditions might require the same skill to be performed next and, therefore, require separate operators. Often these limitations lead to a large array of different operators whose only function is switching to the next skill in different situations. For example, the 'update-WM' skill required four different operators only for switching between skills in the three tasks we modeled due to its centrality in those tasks. Extending the 'update-WM' skill to more tasks would only introduce more of such operators even though the basic procedural knowledge of updating WM would remain the same. This puts the cognitive plausibility of this way of switching skills into question, because it implies that every skill includes many operators that are only responsible for switching to the next skill.

This exposes two core limitations that are present in the current conception of PRIMs (and also ACT-R). Firstly, skills take care of two separate aspects of cognition: they perform the cognitive processing steps and are responsible for goal selection. That is, they are responsible for both selecting the goals and ensuring that they are achieved. This makes skill reuse difficult because, as our example shows, the basic procedural knowledge (which takes care of achieving a certain goal) might remain stable in most situations but the goal selection process might be different. Separating goal selection from goal execution will make skill reuse much easier. The second limitation is related to the type of information on which goal selection is based. Currently, goals are purely selected based on declarative knowledge. At the start of a task, by creating the goal-switching operators a 'plan' for the task is laid out and the model is practically incapable of deviating from this path. This way of goal selection is too rigid and overlooks the fact that people select goals based on a plan combined with their perception of the current situation (Altmann & Trafton, 2002).

Our modeling suggests that goal selection should be separated from execution and be made more flexible. However, this is not an easy task. The basic assumptions of PRIMs do not consider goal selection a special case of cognition and posit that it should be accomplished by a PRIM. Furthermore, increasing the flexibility of goal selection leads to questions of how this flexibility can be balanced with reliability since a more flexible model will also be more unpredictable.

Condition checking

The final factor limiting the creation of fully reusable skills we will discuss here is related to a fundamental aspect of both ACT-R and PRIMs, namely condition checking. In both architectures, productions consist of a condition side (left-hand side) and an action side (right-hand side). The conditions are compared to the content of the buffers before the action side is executed. In ACT-R, the conditions of all productions are evaluated in parallel and when multiple productions match the current contents of the buffer the production with the highest utility factor will be chosen. In PRIMs, condition checking occurs serially starting with the first condition of the most active operator. When one of the conditions does not match, the next active operator will be tested until a matching operator is found. This takes a certain amount of time at first, but after a while most conditions will be compiled into one execution cycle and the most active matching operator will usually be picked without any time cost (comparable to ACT-R).

Conditions are thought to be a fundamental part of procedural knowledge in both architectures. Therefore, full skill reusability means that both the action as well as the condition side need to be reused. Although the action side usually works in both tasks, the condition side is more problematic. After all, a different task usually means a different context to which the conditions will be matched. This often means that the condition side of an operator needs to be adapted to the new task which hinders reusability. Conditions that are especially challenging are those that are related to specific situations in a certain task. For example, in one of the updating models WM needed to be updated based on information in the visual buffer while in a different model it had to be updated based on information in WM itself. In this situation the action PRIMs (the right-hand side) were identical, but a different operator still needed to be created to accommodate the difference in conditions.

This leads to the question to which extent conditions are reused. The quick learning displayed by humans suggests that some previously learned condition-action associations are retained when a new task is performed, however our modeling implies that this does not apply to all of them. Take for example the operator depicted below.

```
operator respond-value-WM1 {
V1 = *report-instructions
WM1 <> nil
==>
*action -> AC1
WM1 -> AC2
}
```

This operator gives the response (stored in WM1) at the end of a trial by performing an action (e.g., pressing a key on a keyboard). In this case, the second condition can be retained without problem because reporting WM1 would always require WM1 to not be empty. However, the other condition which tests whether the report instructions are currently on-screen should probably not be retained because it depends on the task.

The example suggests that not all conditions are created equal and that some conditions should not be reused. Especially conditions aimed at representing a task-specific situation hinder skill reuse suggesting that conditions might not be the best way to represent task-specific context.

Potential solutions

The three limitations we discussed impede the practical usefulness of the skill-based approach but we believe that they will not present a fundamental roadblock to fully *reusable* skills. The limitations we discussed are largely consequences of the reliance of cognitive models on the input of task-specific details from the modeler. Therefore, these issues might be alleviated by implementing learning mechanisms with which the model can figure out task-specific details independently or by providing more principled ways in which the modeler can specify such details.

The first limitation we discussed involved WM. The key issue here is that the inflexible WM demands a lot of coordination from the modeler because the model is not aware of the identity of the WM contents. A possible way to alleviate this would be to store the to-be remembered value together with its meaning (e.g., store the value "four" together with "current-stimulus"). This cannot be done in the current conception of the WM; however, the DM module does possess the required properties. By storing chunks in the DM (such as depicted below) the model would be aware of the value as well as the identity.

```
ISA fact
SLOT1 binding-fact
SLOT2 current-stimulus
SLOT3 four
```

In this situation, the current PRIMs imaginal buffer (i.e., the WM buffer) would be used almost exclusively to facilitate the creation of new chunks and as a problem-state (Borst, Taatgen, & van Rijn, 2010). Importantly, in order to keep the high flexibility of a buffer-based WM, these chunks should be accessible without the need of an explicit retrieval request but instead through means of a PRIM. For example, by allowing a PRIM to directly create bindings (e.g., *four -> *current-stimulus*).

This way of organizing WM provides a better balance of flexibility and plausibility, because chunks are subject to decay and retrieval times, however the information in WM is still easily accessible because it can be directly done by a PRIM. Furthermore, this design of the short-term memory would also provide a mechanism for the variable binding problem discussed earlier in the introduction. The dynamic bindings required to facilitate flexible model behavior could be stored in this same manner. Ideally, the model would create these flexible binding chunks independently (e.g., when 'reading' the instructions) which would tremendously improve skill reusability as well as model autonomy.

The second limitation we discussed involved the manner in which the next skill is selected in PRIMs. This issue boils down to how the next goal is placed in the goal buffer. In the current situation, the previous skill usually places the next skill in the goal buffer but this method creates a large amount of procedural knowledge only aimed at switching between skills.

There is a possible solution that fits the PRIMs philosophy. Instead of having one active skill, two skills can be active: one skill for execution, and one skill for planning. The execution skill carries out the actions required to achieve a particular subtask, and then terminates itself. The planning skill is then responsible for selecting a next skill.

This would be a big improvement over the current situation because it allows for goal switching separate from goal execution based on both a pre-made plan as well as the current context. Additionally, it allows for a flexible representation of task-specific information without the need to include such information in the general skills.

The final limitation we discussed concerned condition-checking. The limitation to skill reusability associated with condition-checking is that every task has a different context which makes it likely that the original conditions will not apply. Additionally, our modeling showed an important distinction between generally applicable conditions and task-specific conditions and raised the question whether conditions are the best way to represent task-specific context.

Testing conditions is one way to establish a mapping between the current state of the cognitive system and the action to be taken, but not the only one. Neural network approaches to modeling operators often use inspiration from the basal ganglia. The basal ganglia are considered to be central to forming

context-action mappings and recent modeling efforts have created models capable of creating such mappings. These mappings provided reusable context-action associations while retaining flexibility by means of small changes to the connection weights in the network (Stewart, Bekolay, & Eliasmith, 2012; Taatgen, 2020).

Such functionality could be incorporated in production-based architectures by specifying (or learning) connections between certain items in the workspace and operators. For example, the first condition of the previously mentioned example could be replaced by specifying a positive connection (through spreading activation) between the report-instructions and this operator. This would make it more likely that it gets picked when such instructions are on the screen but it does not prevent the operator from firing when they are absent. This functionality is already possible in PRIMs but it might be helpful to explicitly make it part of an operator definition (in addition to conditions) which is not only practical but also highlights that these connections are reused.

Conclusion

The skill-based approach is a promising addition to the arsenal of a cognitive modeler; however, the previous discussion has shown that there are still some important limitations. The inflexible WM demands a lot of coordination from this modeler, the unnatural goal selection requires a large amount of inefficient procedural knowledge and the all-or-nothing condition checking severely hampers the versatility of operators. Resolving these issues will require some substantial modifications to the cognitive architecture we employed and to production-system architectures in general. We proposed some solutions in this paper which we will explore in a subsequent study.

The current paper resulted from attempting to apply the skill-based approach to a series of basic tasks that make use of skills that are widely used. The difficulties we experienced show that current cognitive architectures do not support the creation of fully *reusable* skills. This does not mean, however, that the skill-based approach is completely ineffective, current architectures do support the use of *reused* skills and capitalizing on this characteristic will already result in more valid and generalizable models.

References

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: an activation-based model. *Cognitive Science*, *26*(1), 39–83.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–1060. https://doi.org/10.1037/0033-295X.111.4.1036

Borst, J. P., Taatgen, N. A., & van Rijn, H. (2010). The problem state: a cognitive bottleneck in multitasking. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, *36*(2), 363–382.

Feldman, J. (2013). The neural binding problem(s). *Cognitive Neurodynamics*, *7*(1), 1–11.

Greff, K., van Steenkiste, S., & Schmidhuber, J. (2020). On the binding problem in artificial neural networks. *ArXiv Preprint ArXiv:2012.05208*.

Hoekstra, C., Martens, S., & Taatgen, N. A. (2020). A Skill-Based Approach to Modeling the Attentional Blink. *Topics in Cognitive Science*, *12*(3), 1030–1045. https://doi.org/10.1111/tops.12514

Hoekstra, C., Martens, S., & Taatgen, N. A. (2022). Testing the skill-based approach: Consolidation strategy impacts attentional blink performance. *Plos One*, *17*(1), e0262350.

Lee, F. J., & Anderson, J. R. (2001). Does Learning a Complex Task Have to Be Complex?: A Study in Learning Decomposition. *Cognitive Psychology*, *42*(3), 267–316.

Martens, S., & Wyble, B. (2010). The attentional blink: past, present, and future of a blind spot in perceptual awareness. *Neuroscience and Biobehavioral Reviews*, *34*(6), 947–957. https://doi.org/10.1016/j.neubiorev.2009.12.005

Miyake, A., Friedman, N. P., Emerson, M. J., Witzki, A. H., Howerter, A., & Wager, T. D. (2000). The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: A latent variable analysis. *Cognitive Psychology*, *41*(1), 49–100.

Nijboer, M., Borst, J., van Rijn, H., & Taatgen, N. (2016). Contrasting single and multi-component working-memory systems in dual tasking. *Cognitive Psychology*, *86*, 1–26.

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

Stearns, B., & Laird, J. (2018). Modeling Instruction Fetch in Procedural Learning. In *Proceedings of the 16th international conference on cognitive modeling*.

Stewart, T. C., Bekolay, T., & Eliasmith, C. (2012). Learning to select actions with spiking neurons in the Basal Ganglia. *Frontiers in Neuroscience*, *6*, 2.

Stocco, A., Lebiere, C., & Anderson, J. R. (2010). Conditional routing of information to the cortex: a model of the basal ganglia's role in cognitive coordination. *Psychological Review*, *117*(2), 541–574.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471. https://doi.org/10.1037/a0033138

Taatgen, N. A. (2020). A spiking neural architecture that learns tasks. In *Proceedings of ICCM 2019 - 17th International Conference on Cognitive Modeling* (pp. 253–258).

Taatgen, N. A., Huss, D., Dickison, D., & Anderson, J. R. (2008). The acquisition of robust and flexible cognitive skills. *Journal of Experimental Psychology. General*, *137*(3), 548–565.