Diminishing Return in Transfer: A PRIM Model of the Frensch (1991) Arithmetic Experiment

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

Abstract

It is generally believed that transfer decreases with expertise. Several existing models explain this by different forms of chunking, but each is specific to a particular task. The new PRIM theory of cognitive skill transfer allows a more generalized approach to study this phenomenon. To demonstrate this, I present a model of the Frensch (1991) experiment of transfer that explains why a short training period does not differentiate between three types of training in terms of the amount of transfer, but a longer training period does.

Keywords: Learning, Skill acquisition, Transfer, Production compilation.

Introduction

There is a general, but tentative, consensus that certain skills like language, mathematics and algebra have cognitive benefits beyond their own domain. For example, bilingualism has benefits far beyond direct language skills, and is associated with improvements in cognitive control and working memory. There is also evidence that as knowledge becomes more specialized it becomes less useful for transfer. There are several models of these phenomena, but they do not yet paint a consistent image.

To account for transfer among cognitive skills, Thorndike (Thorndike & Woodworth, 1901) introduced the theory of *identical elements*: skills only benefit from each other insofar the elements that represent the knowledge for the skills are the same. Thorndike used this theory to argue that transfer is fairly limited, and that only when knowledge elements are truly identical, transfer is possible. A modern version of identical elements is the *identical productions* theory by Singley and Anderson (1985). In their model, transfer between two tasks is defined as proportion of production rules that are identical among models of those tasks.

Both in Thorndike's and Singley and Anderson's account the identical elements are fairly task-specific. As a consequence, both predict no transfer between tasks that share no surface characteristics. More recent studies, for example Jaeggi et al. (2008), show evidence for such transfer (far transfer). To account for far transfer, and also to give more detailed accounts of earlier transfer studies, Taatgen (in press) introduced the PRIM theory of skill acquisition and transfer. The PRIM theory is also an identical elements theory, but the elements of knowledge are smaller than the typical task-specific rules, and are not specific to a task. Instead, they specify how information between different cognitive modules is matched and transported. Any specifics of the task are part of the information that is transported, and are therefore not part of the representation itself. I will explain the PRIM theory in some more detail later on, but one of its advantages over earlier theories is that it models the learning process in detail. This means that transfer aspects of the learning process can also be explained. More in particular, we will examine whether we can explain the diminishing return of increased expertise.

Diminishing return in transfer

There are a few modeling studies that have produced evidence for the diminishing return of expertise. Transfer in most of these studies is relatively limited in the sense the main task does not change.

Newell and Rosenbloom (1981) describe a model of the 1023-choice reaction task. In this task, subjects are presented with ten lights corresponding to their ten fingers. When a particular combination of lights comes on, they have to press the corresponding fingers. Training on this task produces the classical power law of practice. Newell and Rosenbloom explain this phenomenon through chunking. They assume that initially each light has an individual rule, so that the number of rules necessary for a response is equal to the number of lights. Learning produces combinations of these rules. The initial combinations are very useful (e.g., a combination of two rules is useful once every four trials), but as the new rules themselves are combined, the utility of the more specialized rules decreases, until specialization produces rules that are only useful 1 out of 1023 trials.

In series of analogical reasoning experiments by Anderson and Fincham (1994) subjects were first able to use examples in the analogy in two directions even if they were only trained in one direction. Later in training, though, the trained direction became more efficient than the untrained one. Taatgen and Wallach (2002) explain this with a model of proceduralization: initially examples are retrieved from declarative memory in which there is no directional preference, but later production rules are learned that only operate in the direction that they are trained in.

In a study by Frensch (1991), subjects had to solve a series of six equations (see Figure 1 for an example display). Training consisted of three possible regiments: fixed order, in which subject solved the equations from top to bottom, random order, in which the arrow indicated the next equation to solve, and blocked, in which subjects had to solve the same equation multiple times with different

numbers before moving on to the next. After training, all conditions were tested on series of fixed order trials in which one of the equations was changed. The main result was that if training was short (25 blocks), performance in the test phase was identical. However, after long training (75 blocks), there was a clear differentiation: fixed order training produced superior performance over random order training, which was better than blocked training.

Frensch (1991) modeled the experimental results with a production system model that used production composition. Equations were first solved in small, individual steps that were later combined into larger production that could carry out multiple steps in one cycle.

The three models discussed above are similar in the sense that they all use some form of composition or compilation. They are nevertheless all different in the details, and all start with an initial production system model of the task that just becomes more efficient through training. It is therefore not easy to generalize from these three models, and they are also not capable of explaining transfer beyond the task that they model. I will therefore present a new model of the Frensch (1991) experiment using the PRIM model. I will first explain the Frensch experiment in more detail, and then explain the PRIM approach to modeling the data.

Experiment

In the experiment, which I already informally described in the introduction, Frensch used a task developed by Elio (1986). Figure 1 shows an example of the screen that subjects worked on.

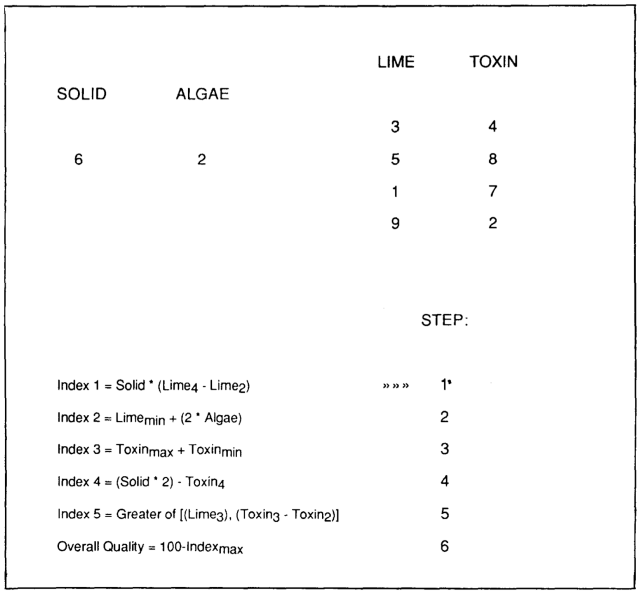

Figure 1. Example screen in the Frensch (1991) task. Copyright 1991 the American Psychological Association. Reprinted with permission.

The task was to solve the six equations (Index 1 through 5 and Overall quality) by substituting the variables with the numbers on the top part of the screen, calculate the outcome, and enter it using the keyboard. The arrows indicated which step had to be executed next.

The study had a 3x2 design with three training conditions and two training durations. In the fixed-order training condition, the arrow always moved from step 1 to 6 in order. In the random-order training condition, the arrow would go through the six steps in random order. In the blocked training condition, subjects would do each step multiple times before moving to the next step. In the short-duration condition, subjects received 25 trials of training in one of the three training conditions, followed by 50 trials of trials in the fixed-order condition in which one of the six equations was changed. The long-training condition was the same, except that subject received 75 trials of training.

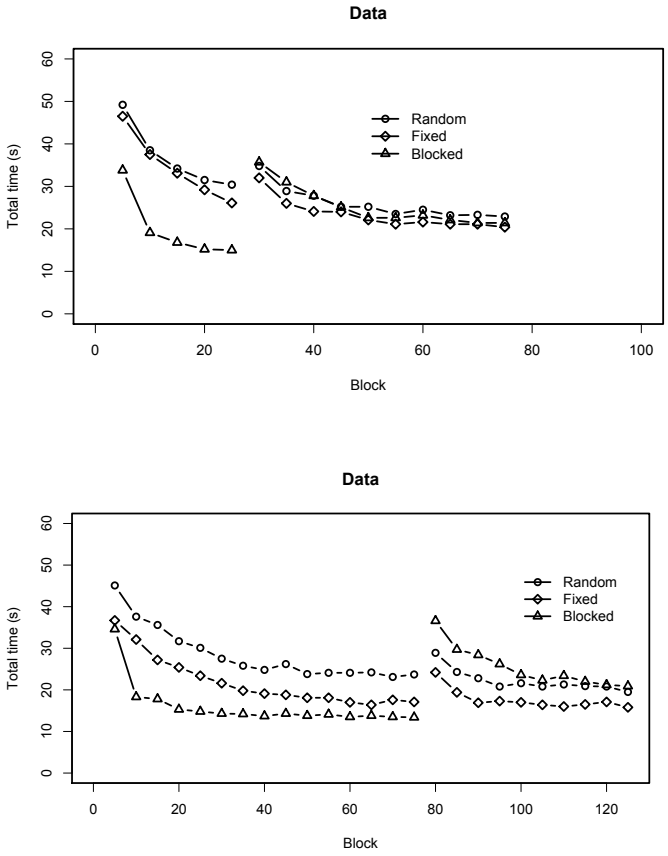

Figure 2. Results of the Frensch (1991) experiment. The top graph shows the short training condition and the bottom graph the long training condition. The left part of each figure is the training phase, with three different training types, and the right side is the transfer phase, where all subjects performed the fixed-order version of the task.

The results (Figure 2) clearly show that in the short training condition performance on the transfer task is the same, with a possible slight advantage for fixed-condition training. However, after long-training condition there is a clear difference: the fixed order has the best performance, followed by random order and the blocked condition. This difference persists until the end of the experiment.

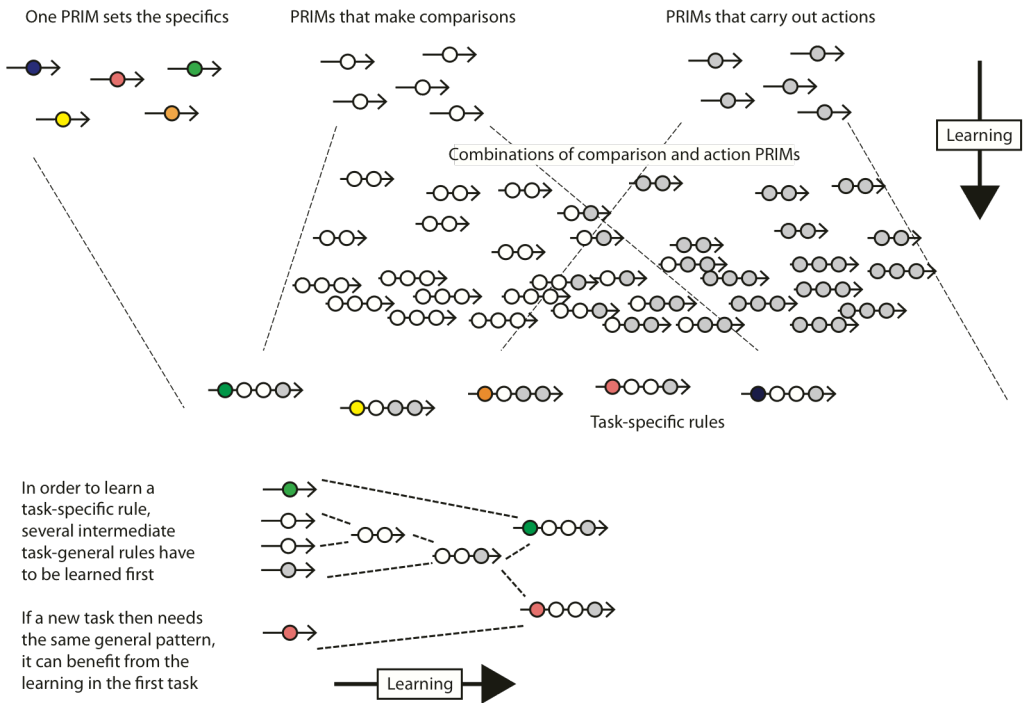

Figure 4. Impression of learning in the PRIMs theory. Circles represent PRIMs, and arrows show combinations of PRIMs. White circles are condition PRIMs, grey circles action PRIMs, and colored circles PRIMs that set specific values. The top figure illustrates how the initial rules with just one PRIM are 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.

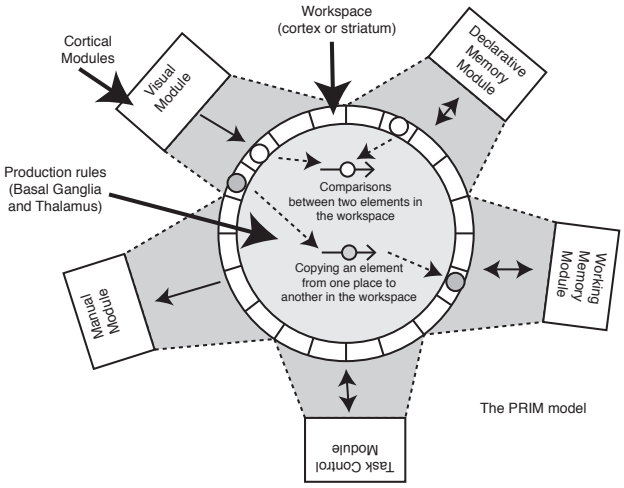

Figure 3. Global workspace/ACT-R buffer model of information processing. From Taatgen (in press). Copyright the American Psychological Association. Reprinted with permission

The PRIM theory

The PRIM (primitive information processing elements) theory (Taatgen, in press) is an extension of the ACT-R architecture (Anderson, 2007). The key characteristic of the theory is that production rules are broken down in elementary processing elements. 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 3 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 standard production rule:

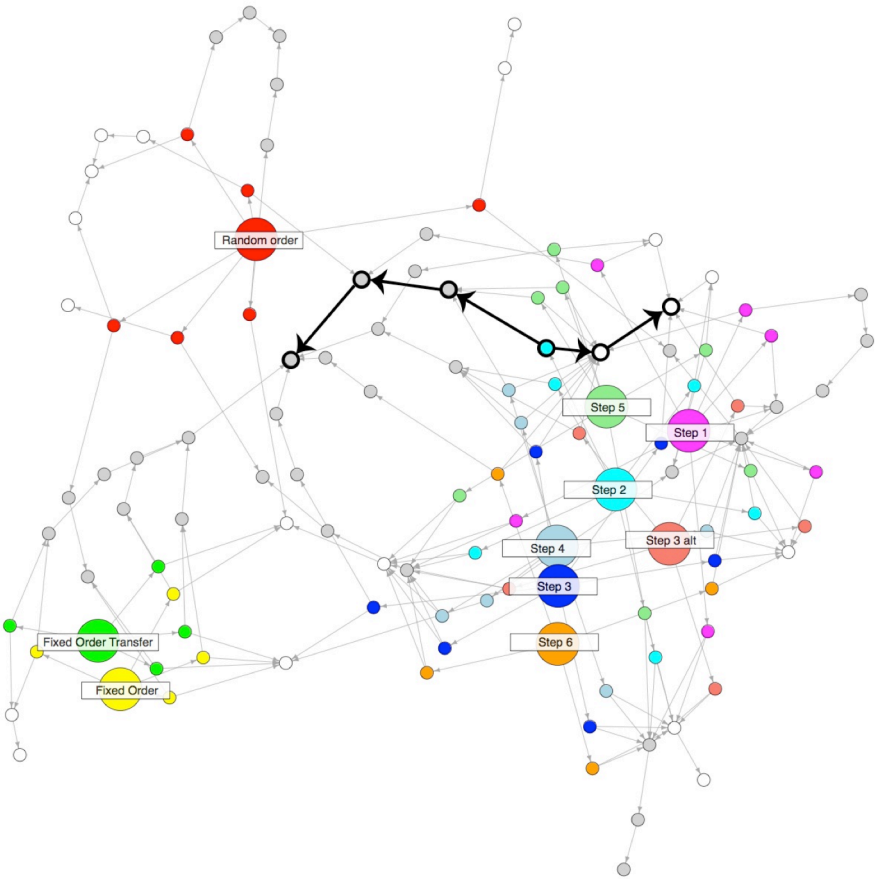

Figure 5. Representation in terms of PRIMs of the Frensch experiment. The same convention is used as in Figure 3: the small colored nodes represent the PRIM that sets tasks-specific items, white nodes are conditions and grey nodes are actions. The large colored and labeled nodes are used to group the PRIMs for specific tasks together, and have no meaning in themselves. Whenever there is a possibility for overlap, tasks share the same (white and grey) nodes. All the task representations that calculate the equations, Step 1-6 and Step 3 alt have many overlapping PRIMs. The Fixed Order and Fixed Order Transfer have identical PRIMs, while the Random Order PRIMs have only small amounts of overlap with other tasks. As an example, a single task-specific PRIM (for the step 2 task) with its condition and action lists is highlighted, comprising what traditionally would be a production rule. The condition and action lists of the example is shared by many other task-specific PRIMs.

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

This rule would be broken down into the following 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 for the ability to model

transfer. 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 4 illustrates both the process of compilation and the process of transfer. 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 (in press).

Model of the Frensch Experiment

The model is an adaptation of the model of the Elio (1986) task reported in Taatgen (in press). Each of the seven equations on the display (six during training plus one extra in the transfer phase) is modeled as a separate task. In the blocked condition they are treated as separate independent tasks, but in the fixed-order and random-order conditions they are subtasks of either a fixed-order or a random-order goal.

Each of the equation tasks is relatively straightforward: the appropriate numbers are looked up on the screen, and arithmetic facts are retrieved from memory to calculate the answer, which is then typed in. The overlap between these seven tasks is fairly large, because the pattern of alternating looking up information and arithmetic facts is slightly different for each equation, but similar enough for positive transfer among them.

The mappings between the step number and the goal to calculate its equation are stored in declarative memory. Both the fixed-order and the random-order task use these mappings. The fixed-order task starts with retrieving step 1, and setting this as a subtask. Once the task is completed the next step is retrieved, until all steps are done. The random-order task first looks at the screen to see what step needs to be done next, then retrieves the appropriate step, sets this as a subtask, and reiterates this until all steps are done. In the blocked condition, no main task is set but the equation subtasks themselves are used as main task.

Figure 5 shows the overall structure of the PRIMs necessary to do the experiment. The best way to think about it is to consider each of the small colored circles as a production rule. Each of the colored circles points to a linked list of white circles, the condition PRIMs, and a linked list of grey circles, the action PRIMs. Together they represent the knowledge for a single traditional production rule.

When the model is run, the learning process gradually compiles larger and larger rules according to Figure 3, eventually learning rules specific to the particular tasks. The amount of overlap among the PRIM structures determines how much mutual benefit there is from learning. What we can already see in Figure 5 is that the tasks that solve the equations (step 1-6) have a large mutual benefit, but the random-order and the fixed-order tasks largely have their own knowledge structures (the two fixed-order tasks, before and after transfer, have of course a perfect overlap, because they are identical in structure).

To test the model, it was run through the same procedure as the subjects in the experiment, leading to six different completion time predictions.

Model results

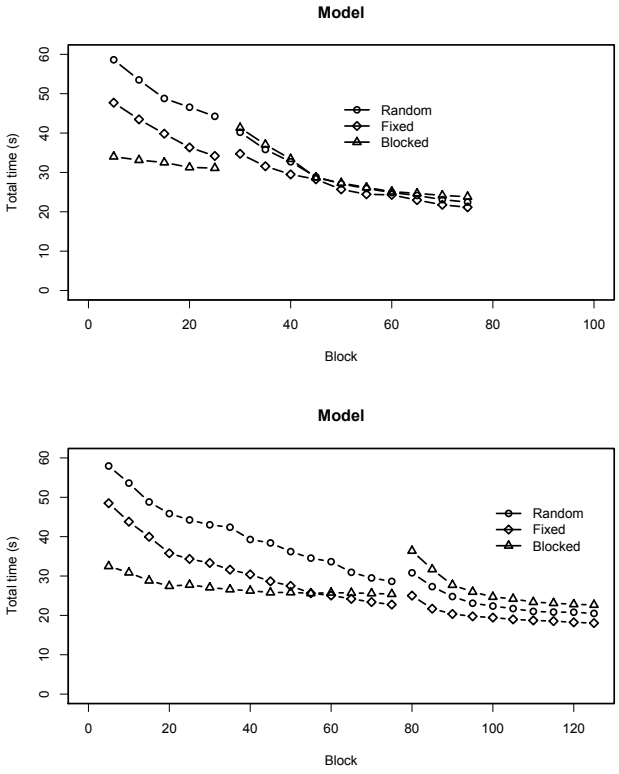

Figure 6. Model results: top figure is the short-training condition, and bottom figure the long-training condition

Figure 6 shows the results of the model. It matches several patterns in the data. The most important is the effect of transfer: in the short-training condition, the only difference between the training conditions is a slight initial edge for fixed-order training, but this difference does not persist. In the long-training condition, there is a clear differentiation in the three training conditions, just as in the data. The explanation for this fit is the fact that initial learning mainly focuses on the six arithmetic procedures. These procedures have a very strong overlap in the PRIMs (as can be seen in Figure 4), so each of these receives ample initial training. The components of the random-order and

fixed-order tasks receive much less training because they have hardly any overlap with anything else. As a consequence, in the short-training condition the model has become faster at calculating the equations, but not so much at overall control, and this shows in the transfer phase. In the long-training condition, on the other hand, the control procedures have had enough training to produce transfer. Fixed-order training produces the best transfer, because the procedure remains the same. Random-order training still edges out the blocked training because of the small amount of overlap between fixed-order and random-order training. Like in the data, these differences persist until the end of the experiment.

The model also fits some of the more mundane aspects of the data: in the training phase, blocked training is by far the fastest, and random-order training the slowest. Also, the speed-up due to training follows the power law of practice.

Discussion

What we have seen in the experiment and the model is that initially different types of training led to the same amount of transfer. But after more training transfer became more specific, and elements that were learned in the random-order and in the blocked training no longer fully transfer to the fixed-order condition.

So, what are the advantages of the PRIM model over the earlier models? Like its three predecessor models (the Newell & Rosenbloom Soar chunking model, the Frensch production composition model and the Taatgen and Wallach production compilation model), it uses the principle of combining two existing knowledge components to build a larger, more specialized component. The main difference is that the three existing models all start with task-specific production rules that become more specialized and therefore more efficient. Although this is sufficient to model individual tasks, it also means that every particular task needs its own particular model. The PRIM model on the other hand starts with a fixed set of production rules, and a representation of how to do the task in declarative memory, so it starts at a much lower level. As a consequence, even the final task-specific rules are still relatively simple, while the older production models allowed the size of rules to get out of hand, potentially solving a whole problem in one huge production.

The PRIM theory still has a number of open issues. The current model starts with a "blank slate", in the sense that it only has the PRIMs to start with and yet no rules that are combinations of PRIMs at all. This probably explains why the model is initially slower than the subjects. As already discussed in Taatgen (in press), it is probably worthwhile to explore what general patterns of reasoning are common in human reasoning, and prime the model with that. Note, however, that few current cognitive models take prior skills into account.

Unlike its predecessors, the PRIM model also allows to study far transfer, and the limits of specialized training for far transfer. Is there an explanation for the effect of bilingualism on cognitive control in general, and what kind of alternative training could be effective? What aspects of learning mathematics transfer best to non-mathematical domains? Hopefully the new theory can provide some answers to these questions.

Acknowledgements

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

References

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

Anderson, J. R., & Fincham, J. M. (1994). Acquisition of procedural skills from examples. *Journal of experimental psychology: learning, memory, and cognition, 20*(6), 1322-1340.

Elio, R. (1986). Representation of similar well-learned cognitive procedures. *Cognitive Science, 10*, 41-73.

Frensch, P. A. (1991). Transfer of composed knowledge in a multistep serial task. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 17*(5), 997-1016.

Jaeggi, S. M., Buschkuehl, M., Jonides, J., & Perrig, W. J. (2008). Improving fluid intelligence with training on working memory. Proceedings of the National Academy of Sciences, 105(19), 6829-6833. doi: 10.1073/pnas.0801268105

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 1-56). Hillsdale, NJ: Erlbaum.

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

Taatgen, N. A. (in press). The nature and transfer of cognitive skills. *Psychological Review*. Preview available at http://www.ai.rug.nl/~niels/actransfer.html

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition, 86*(2), 123-155.

Taatgen, N. A., & Wallach, D. (2002). Whether skill acquisition is rule or instance based is determined by the structure of the task. *Cognitive Science Quaterly, 2*(2), 163-204.

Thorndike, E. L., & Woodworth, R. S. (1901). The influence of improvement in one mental function upon the efficiency of other functions. *Psychological Review, 8*, 247-261.