UC Merced

Title
Modeling Speed-up and Transfer of Declarative and Procedural Knowledge

Permalink
https://escholarship.org/uc/item/07d3w1v8

Journal

Authors
Johnson, Todd R.
Wang, Hongbin
Zhang, Jiajie

Publication Date

Modeling Speed-up and Transfer of Declarative and Procedural Knowledge

Todd R. Johnson (Johnson.25@osu.edu)
Department of Pathology; The Ohio State University, 2015 Neil Ave. Room 395
Columbus, OH 43210

Hongbin Wang (wang.190@osu.edu)
Department of Psychology; The Ohio State University
Columbus, OH 43210

Jiajie Zhang (zhang.52@osu.edu)
Department of Psychology; The Ohio State University
Columbus, OH 43210

Abstract

This paper addresses three hypotheses concerning the procedural/declarative distinction: 1) Procedural and declarative knowledge speed-up as separate, but parallel, power curves; 2) Procedural knowledge operates in one direction only—from condition to action—whereas declarative knowledge can be cued by any of its elements; and 3) Declarative knowledge is active—it can result in behavior independent of procedural knowledge. The paper presents a single Act-R model that closely fits the data of two learning and transfer experiments conducted by Rabinowitz and Goldberg (1995). The model provides a good fit to the data, further validating Act-R as a model of the human cognitive architecture. In addition, the model shows that the two experiments cannot be used to argue that declarative knowledge can be retrieved without any intervening procedural knowledge.

Introduction

Recent results suggest that the retrieval of declarative knowledge and the application of procedural knowledge speed up as separate power laws of practice (see VanLehn, 1996 for a review). In other words, the time to retrieve a declarative memory element speeds up as a power function of the number of retrievals, whereas the time to apply a procedure speeds up as a power function of the number of applications. Another set of results indicates that procedural knowledge operates in one direction only—from condition to action—whereas declarative knowledge can be cued by any of its elements (Anderson, 1993).

Rabinowitz and Goldberg (1995) conducted two experiments that nicely illustrate these phenomena. These experiments use a learning and transfer paradigm to examine learning of declarative and procedural knowledge, and their different retrieval characteristics. In addition, they used the results to argue that declarative knowledge is active—it can result in behavior independent of procedural knowledge. Furthermore they argued that Act-R (Anderson, 1993) could not account for these findings, because its declarative memory is inert and can be retrieved only in the service of a production rule.

This paper presents a single Act-R model that accounts for the data in the two Rabinowitz and Goldberg experiments. In addition, the paper presents protocol results from a new experiment designed to further test the assumptions of the experiments and the model. Trafton (1996) has described an Act-R model for one of Rabinowitz and Goldberg's experiments, but a bigger challenge is to construct a single Act-R model that can account for the results from both experiments.

Rabinowitz and Goldberg's Experiments

Both experiments used an alphabet arithmetic task, which consists of problems of the form *letter1 + number = letter2*, where *letter2* is *number* letters after *letter1*. For example, A+2=? is C, because C is 2 letters after A.

In Experiment 1, one group of participants (the consistent group) received training on 12 different alphabet addition problems that were repeated for 36 blocks. Another group of participants (the varied group) received training on 72 different problems that were repeated for 6 blocks. The problems used addends from 1 to 6. Consistent problems had two occurrences of each addend, whereas varied problems had 12 occurrences.

In the transfer phase, both groups received 12 new addition problems, repeated 3 times. Rabinowitz and Goldberg reasoned that during training the consistent group would quickly acquire declarative knowledge of the answers and switch to retrieval, whereas the varied group would continue to count up the alphabet. Thus the consistent group would get a lot of practice at retrieving the answers to the same 12 problems, but relatively little practice on the procedural knowledge needed to count up the alphabet. In contrast, the varied group would receive little or no practice retrieving declarative knowledge, but a great deal of practice counting up the alphabet. When transferred to the 12 new addition problems, the consistent group should revert to counting up the alphabet, resulting in a dramatic decrease in speed. However, the varied group should show perfect transfer from the training problems to the new problems.

The training results are shown in Figure 1. Each point on the graph is the mean of the median response times for all subjects on a block of 12 problems. The different asymptotes support the assertion that varied participants practiced procedural knowledge, whereas consistent participants practiced
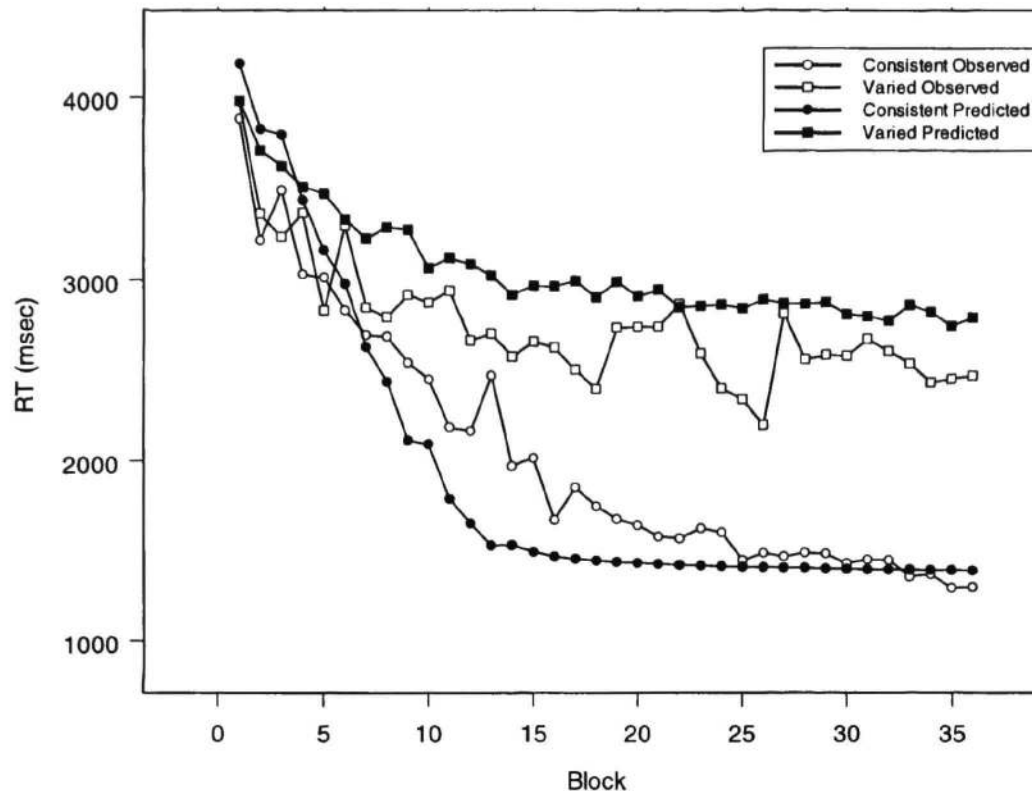
Figure 1: Observed and predicted mean response times during alphabet arithmetic training as a function of training group and practice block. Observed data replotted from Rabinowitz and Goldberg (1995).

retrieval.

The transfer results support the predictions (see Figure 2): the varied group showed perfect transfer (from 2463 ms to 2427 ms), but the consistent group showed considerable slow down (from 1294 ms to 2858 ms).

Although Experiment 1 supports the predictions, it is also consistent with a procedural-only long-term store. The consistent subjects might have acquired problem-specific procedural knowledge that directly produces the answer to each problem. For example, knowledge of the form "If problem is A+2, then type C." Since this knowledge is specific to the 12 training problems, it would not have helped the participants during the transfer phase. This issue is examined in Rabinowitz and Goldberg's second experiment.

The second experiment attempts to determine whether consistent training leads to specific procedural knowledge, or to declarative knowledge. It is based on the hypothesis that declarative and procedural knowledge have different retrieval characteristics. Declarative knowledge is thought to be subject to symmetric retrieval, meaning that any part of a declarative memory element can act as a cue for the retrieval of that element. Procedural knowledge is thought to be subject to symmetric access, meaning that a procedure operates in only one direction: from condition to action.

Training in Experiment 2 was identical to Experiment 1, however, in the transfer phase, both groups were given 12 subtraction problems repeated 3 times. A subtraction problem is of the form *letter1 number = letter2*. For example, C-2=A. The 12 subtraction problems were inverted versions of the addition problems that both groups had seen during training. If the consistent group acquires declarative knowl-

edge of the addition problems, the participants in this group should be able to solve the subtraction problems by retrieving and inverting addition problems. However, if this group has acquired problem-specific procedural knowledge, they will need to develop a new procedure for counting down the alphabet, as will the varied participants—who presumably strengthen their procedural knowledge during training.

Training results are similar to those for Experiment 1. The transfer results are consistent with the predictions: the varied group requires considerably more time than the consistent group (see Figure 3). The consistent group went from 1100 ms to 4557 ms, whereas the varied group went from 2500 ms to 7689 ms.

Taken together, Experiments 1 and 2 support the speed-up of both declarative knowledge retrieval and procedural knowledge application, as well as symmetric access to declarative knowledge and asymmetric access to procedural knowledge.

An Act-R Model

Act-R (Anderson, 1993) seems well suited for modeling these results, because it contains procedural and declarative long-term stores, along with learning mechanisms that alter the speed of elements in the two stores as a function of experience. Such a model will serve three purposes. First, it will act as an additional test for several of Act-R's theoretical assumptions. Second, although each of Act-R's mechanisms has been tested in isolation, this model will test the interaction of several mechanisms. Third, the model will provide an explicit account of declarative and procedural learning and
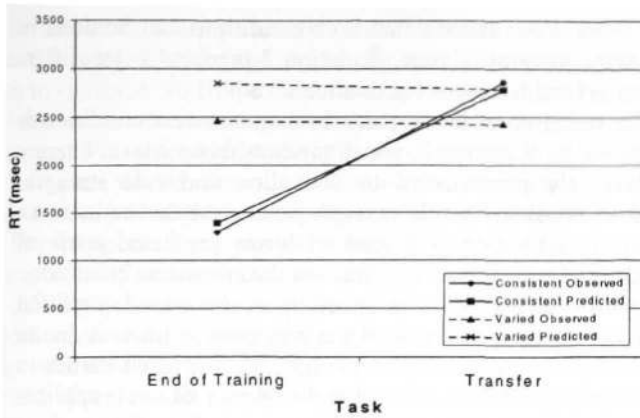
Figure 2: Mean response times for Experiment 1 as a function of task and group.

Figure 3: Mean response times for Experiment 2 as a function of task and group.

transfer that might then be used to analyze a wide range of more complex cognitive tasks. The model presented here uses Act-R 4.0 (Anderson & Lebiere, in press).

The alphabet arithmetic model has three main production rules for the main goal (see Table 1). RETRIEVE-PLUS-RESULT attempts to solve an addition problem by retrieving a fact from declarative memory that matches the problem, but also contains the answer. If successful, it uses the retrieved answer as the solution. RETRIEVE-MINUS-RESULT attempts to solve a subtraction problem by retrieving an addition declarative memory element (DME) that is the inverse of the subtraction problem. In other words, if the current problem is C-2=?, this rule will attempt to retrieve a fact of the form letter + 2 = C. SUBGOAL-COUNT creates a subgoal to solve the current problem by counting up or down the alphabet.

Table 1: The English version of the model's main production rules.

Retrieve-Plus-Result
IF the goal is to do an alphabet ADDITION arithmetic problem of the form letter1 + number =, but the answer has not been determined, and there is a fact in memory stating that letter1 + number = letter2 THEN note letter2 as the answer
Retrieve-Minus-Result
IF the goal is to do an alphabet SUBTRACTION arithmetic problem of the form letter1 - number =, but the answer has not been determined, and there is a fact in memory stating that letter2 + number = letter1 THEN note letter2 as the answer
Subgoal-Count
IF the goal is to do an alphabet arithmetic problem, but the answer has not been determined THEN set a subgoal to compute the answer by counting

The model first tries to retrieve an answer by using the appropriate retrieval rule. If the retrieval fails, then SUBGOAL-COUNT will fire to create the computation subgoal.

The model switches from computation to retrieval by acquiring declarative representations of problems that it has solved. When the model begins to solve problems it will not
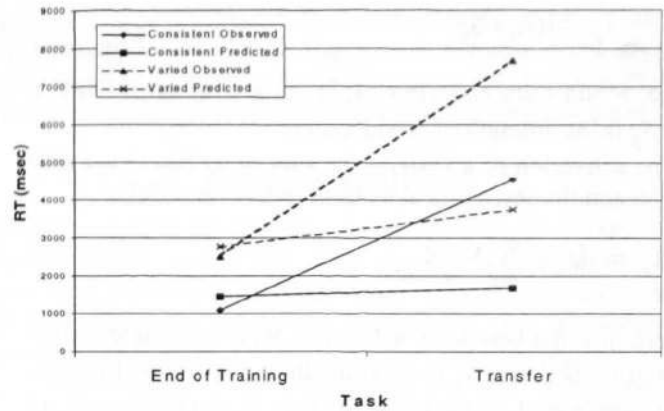
have any DMEs to retrieve, so it will always use SUB-GOAL-COUNT. However, each time it solves a problem, it automatically remembers the problem and solution as a DME. These DMEs are then available for recall in future trials. Details of this memorization process are given below.

The computation subgoal works by counting either up or down the alphabet. It uses a set of declarative memory elements that represent the alphabet using the chunks:

ABCD EFG HIJK LMNOP QRS TUV WXYZ

Each chunk is a DME containing up to five letters and a pointer to the next chunk.

The subgoal contains 26 rules that implement counting forward and backward through the alphabet. To do this, it must first retrieve the alphabet chunk that contains the starting letter (e.g., A for A+2=?). Next it steps forward along the chunk until it finds the starting letter. Finally, it counts along the alphabet (either forward or backward) the required number of letters. If it reaches a chunk boundary, it must retrieve either the next or previous chunk before continuing the count.

The subgoal automatically produces a declarative memory trace of the problem and its solution. Goals in Act-R are DMEs that have been pushed onto the goal stack. You can think of a goal as problem-specific working memory, because it encodes the problem, the solution, and possibly intermediate results. When the subgoal is achieved, a rule pops the goal off of Act-R's goal stack, but the goal remains in declarative memory as a DME representing the problem and its solution.

The model uses three of Act-R's mechanisms: base-level learning, which speeds up access to commonly retrieved DMEs, strength learning, which speeds up rules that are commonly used, and the memory retrieval threshold, which prevents the retrieval of DMEs below a specified activation.

To understand how these mechanisms produce the speed-up and transfer shown in the data, you must first understand how Act-R predicts latencies. The total time for a trial in Act-R is the sum of the times needed to fire each production rule during that trial. The time to fire a rule is the sum of the time needed to retrieve the DMEs it matches plus the time to execute the rule's action. *The time to retrieve a DME is inversely proportional to DME activation and production strength:*

533

$$t_i = Fe^{-f(A_i + S_p)} \qquad \text{Equation 1}$$

Here, F and f are constants. A_i is the activation of DME i, and S_p is the strength of production p.

The activation of a DME is the sum of its base level activation and the spreading activation from other DMEs:

$$A_i = B_i + \sum_j W_j S_{ji} \qquad \text{Equation 2}$$

where Bi is the base level activation, W_j is the source activation of DME j, and S_{ji} is the strength of association from j to i. A single unit of source activation is divided among all DMEs that fill slots of the current goal. For the present model, this means that elements of the current problem (i.e., the letter, operator, and number) will spread activation to DMEs representing past solutions. For example, if the current goal is to solve A+2, then A will spread activation to all DMEs of previous problems that contain A either as the first letter or as the answer. The same is true for the operator and the number. Hence, the DME that represents the past solution to the current problem will receive activation from all three elements and will, most likely, be the most active DME.

The base level activation of a DME reflects the log prior odds that the DME will be matched by a production rule. Act-R assumes that these odds increase as a function of use and decrease as a function of delay. This is given by the optimized base-level learning equation:

$$B_i = \ln\left(\frac{nL^{-d}}{1-d}\right) + \beta \qquad \text{Equation 3}$$

where β represents the initial base-level, d is the decay rate, L is the time since the DME was created, and n is the number of times the DME has been used.

The use count of a DME is incremented whenever the DME is matched by a rule or when a duplicate DME is created. As noted above, when a goal is popped from the stack it remains in declarative memory. However, if Act-R detects that a newly created DME is identical to an existing DME, then it destroys the new DME and increments the use count of the old DME. This increases the DME's chances of recall in future trials.

A DME that matches a rule's condition will be successfully retrieved whenever its activation exceeds the global retrieval threshold. When a DME is first created, its base-level activation is set to a base level constant plus a permanent activation noise.

We can now see how the model might learn to retrieve declarative traces in the consistent training condition, but not in the varied training condition. In the consistent condition, the model is exposed to each problem 36 times. These frequent exposures boost the base-level activation of the memory traces, allowing the retrieval rules to directly recall the solutions. In contrast, in the varied condition the model is exposed to each problem only six times. In addition, the varied condition takes longer because the first 72 trials can only be solved by counting. In the consistent condition there is a chance of recalling one or more answers after the first 12 tri-

als. Even if we assume that both conditions can be done in the same amount of time, Equation 3 predicts major differences in final base-level activations.

The speed-up of participants in the consistent condition is predicted by Equation 1, which predicts that retrieval latency is inversely proportional to activation and rule strength. Without considering rule strength we can see that an increase in DME activation will lead to lower predicted retrieval times and hence lower trial times in the consistent condition.

The model predicts that speed-up in the varied condition and part of the speed up in the consistent condition is due to speed-up of procedural knowledge. As discussed earlier in this section, Act-R assumes that the latency of a rule application is inversely proportional to its strength and the activation of the DMEs that it matches (see the discussion surrounding Equations 1 and 2). Rule strength is governed by the same equation that governs base-level learning (Equation 3) except that L is the time since the rule was created, d is a separate strength decay constant, and n is the number of times the rule has been fired.

Strength learning, combined with the latency equations (Equations 1 and 2), predict the speed-up in the varied condition and why varied training produces perfect transfer to new addition problems, whereas consistent training shows no transfer. In the varied condition, the model receives a lot of practice using the rules for counting up the alphabet and these same rules are also used in transfer. In contrast, when the model is given consistent training, it learns to retrieve the answers to the 12 problems, so it rarely uses the counting rules. Once the model reaches the transfer phase it must begin to use the counting rules again, but their strengths will be either at or below their initial values, producing the dramatic slowdown observed in the data.

The model also accounts for the subtraction transfer results. In the consistent condition, the model acquires and strengthens DMEs representing each problem and its solution. When transferred to subtraction, these DMEs have a high enough activation to be retrieved and inverted by RETRIEVE-MINUS-RESULT. The model predicts that performance will be slower than at the end of training, because it has not yet strengthened RETRIEVE-MINUS-RESULT. In contrast, when the model is in the varied training condition, the DMEs rarely become active enough to retrieve, so they are not available during transfer. Although the model has strengthened its rules for counting up the alphabet, very few of these rules are used to count down, so the model must use counting down rules that have not yet been used, and hence are much slower to fire.

The above discussion illustrates how the model can make the right qualitative predictions. The remainder of this section discusses the quantitative predictions and their fit to the observed data.

Four parameters were estimated to fit the model to the data. These were the base-level learning decay parameter (d in Equation 3), production strength decay parameter, retrieval threshold, and permanent activation noise. These four parameters are critical to fitting the data. The rule strength decay parameter affects the learning rate of procedural knowledge. The interaction of the retrieval threshold

with the three other parameters determines the amount of practice needed before the model can switch from computation to retrieval. To fit the data, these parameters must be set so that consistent training leads the model to retrieve the answers, whereas varied training leads the model to continue to compute the answers. In addition, the parameters must also produce the right learning curves for the two conditions.

The fit reported here was obtained with base-level learning decay set to .7, strength decay set to .5, retrieval threshold set to .55, and permanent activation noise variance set to .15. In addition, the total time to read the problem and type a letter was estimated at a constant 1.25 sec. This defines the lower bound of the model's response times. To reflect familiarity with the alphabet, all alphabet DMEs were given initial base-level activations of .974, reflecting 100 uses in the last 1000 seconds. Production rule strengths were initially set to .486, reflecting 25 uses in the past 1000 seconds. All other parameters used the default Act-R 4.0 values.

The model's predictions for the training phase in Experiment 1 are shown in Figure 1 along with the observed data. The model predictions were produced by simulating 15 subjects in each condition. The same model and parameter values were used for both conditions. The R^2 for the consistent condition was .89 and for the varied condition .78. This is pretty good considering that two different groups of subjects were modeled using the same parameters. In addition, the model captures the qualitative trends in the data—consistent simulations get much faster than varied simulations.

The transfer results are shown in Figures 2 and 3. The model closely fits the quantitative and qualitative results for alphabet addition transfer: consistent training leads to a large slow down in the transfer phase, whereas varied training results in perfect transfer. The subtraction transfer simulation matches the qualitative results, but not the quantitative ones: consistent training leads to better performance on subtraction than does varied training, but the model underestimates the latency of subtraction problems.

The poor fit of the model to the quantitative subtraction data for the varied condition is easy to fix. It is possible to increase the time to compute a subtraction problem answer by either decreasing the strength of the subtraction counting rules or by switching to a different technique to solve the problems. A decrease in the rules' strengths is justifiable because most people rarely need to recite the alphabet backwards. However, it is also possible that people use a different strategy, such as guessing an answer and then counting forward to see if it is the right one. The next section further explores this issue.

The poor match to the subtraction latency in the consistent condition is much more puzzling. Specifically, why do the participants need over 4 seconds to solve each problem? If they are really recalling an alphabet addition problem and inverting it, then they should be closer to the predicted times, but instead their times are more than double the predictions. One possibility is that only a subset of varied participants actually switched to retrieval, whereas the remainder used computation. This hypothesis is further explored in the next section.

The model's good fit to the data shows that active declarative knowledge is not needed to account for the results. Thus, the two experiments do not discriminate between declarative knowledge being inert or active. However, it is possible that protocol data might provide evidence concerning this issue.

Protocol Analysis

To better understand the strategies that people use for alphabet arithmetic, particularly with respect to subtraction, a variant of Rabinowitz and Goldberg's Experiment 2 was run at The Ohio State University. Participants were 42 undergraduate students who received course credit for their effort. The main change to the experiment was that participants were required to answer a questionnaire that asked them to describe the strategies they used to solve the problems during training and then during transfer.

The analysis considered the three issues discussed at the end of the last section: 1) strategies for computing subtraction answers; 2) whether only a subset of consistent training participants used retrieval in the subtraction transfer phase; and 3) inert versus active declarative knowledge.

Three main strategies were mentioned during the training phase: counting only, counting plus recall, and computing (in an unspecified way) plus recall. Many more strategies were mentioned in the transfer phase: counting backwards only, recall and inversion only, counting backwards or computing initially then switching to recall and inversion, generate and test, and a mixture of counting back and generate and test. Table 2 shows the percentage of participants in each category. The results clearly support the assumption that varied training leads to more counting, whereas consistent training leads to direct retrieval. 95% of the participants in the consistent group reported using recall during training, versus only 32% of those in the varied condition. Most participants in the varied group (68%) reported that they used only counting throughout the entire training phase, in contrast to only 5% of participants in the consistent group.

The transfer protocol results are consistent with the hypothesis that varied training leads to strengthened asymmetrically accessible procedural knowledge for counting up, whereas consistent training leads to symmetrically accessible declarative knowledge. 70% of the consistent group reported recalling and inverting the addition problems, versus only 5% of the varied group. Likewise, only 15% of the consistent group reported counting back only, versus 36% of the varied group. Another 18% of the varied group used the generate and test strategy.

These results help clarify the model's problems of underestimating the difficulty of subtraction. First, they show that at least 15% of the consistent group used computation instead of recall, offering a possible explanation for the higher than predicted response times for this group on the transfer task. Second, the results indicate that the model's strategy of counting backward is consistent with the majority of participants in the varied group, but that the model is simply underestimating the time required to count back. In fact, two participants who used generate and test, mentioned that they switched to this method because counting back was too difficult.

Table 2: Reported strategy use based on training group and task.

	Condition	
	Consistent (n = 20)	**Varied** (n = 22)
Training		
Counting only	5 % (1)	68% (15)
Count + Recall	80% (16)	32% (7)
Compute + Recall	15% (3)	0%
Transfer		
Counting back only	15% (3)	36% (8)
Recall and Invert	60% (12)	5% (1)
Count back then recall and invert	5% (1)	0%
Compute then Recall and Invert	5% (1)	0%
Generate and Test	5% (1)	18% (4)
Count back + Generate and Test	0%	9% (2)
Other	5% (1)	5% (1)
Not codable	5% (1)	27% (6)

The protocol data provides little evidence of whether declarative knowledge is inert or active. Only 10% of the consistent group mentioned computing the answers to a few subtraction problems before recognizing them as inverted addition problems. For these two subjects, it seems that switching to recall and inversion was a conscious activity that required an initial recognition step. If declarative memory is truly active, the answers to the subtraction problems should come to mind immediately, without a conscious recognition process. Although these results are inconclusive, they do suggest that a follow-up study using concurrent verbal protocols during the transfer phase might resolve this issue.

Conclusion

This paper has three main results. The first is that the successful fit of the model to the alphabet arithmetic results shows that the two experiments fail to discriminate between active or inert declarative memory. Declarative memory in Act-R is inert—it can only be retrieved in the service of a production rule. Although the protocol data provided little insight into this issue, it does suggest that some kind of recognition process is needed before some participants can switch to recall and inversion. Recent work on feeling-of-knowing (i.e., the feeling that you know an answer to a problem) provides some support for this view. Schunn, et al.

(1997) have shown that feeling-of-knowing is based on similarity of the problem to previously seen problems, not on the availability of an answer to the problem. Since subtraction problems are so different from the inverted addition problems, it seems likely that solving one or two subtraction problems might lead to a feeling of knowing based on similarity between the solved subtraction problem and previously seen addition problems. This feeling-of-knowing might then prompt a person to consciously explore the similarities.

Second, the model's successful fit to the data and the protocol results provide additional support for separate declarative and procedural long-term memory stores. In addition, the model also shows that the separate strengthening of procedural and declarative knowledge can produce the observed results.

Third, the paper shows that Act-R is sufficient to capture both the qualitative and quantitative details of the acquisition and transfer of procedural and declarative memory. More importantly, the model shows that several Act-R mechanisms working together can predict whether training will lead to procedural strengthening or the recall of declarative knowledge.

References

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum.

Anderson, J. R., & Lebiere, C. (in press). *The Atomic Components of Thought*. Hillsdale, NJ: Lawrence Erlbaum.

Rabinowitz, M., & Goldberg, N. (1995). Evaluating the structure-process hypothesis. In F. E. Weinert & W. Schneider (Eds.), *Memory Performance and Competencies: Issues in Growth and Development* (pp. 225-242). Hillsdale, NJ: Lawrence Erlbaum.

Schunn, C. D., Reder, L. M., Nhouyvanisvong, A., Richards, D. R., & Stroffolino, P. J. (1997). To calculate or not to calculate: A Source activation confusion model of problem familiarity's role in strategy selection. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 23(1), 3-29.

Trafton, J. G. (1996). Alphabet arithmetic and Act-R: A reply to Rabinowitz and Goldberg, *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*. Mahwah, NJ: Lawrence Erlbaum Associates.

VanLehn, K. (1996). Cognitive skill acquisition. *Annual Review of Psychology*, 47, 513-539.