Whether Skill Acquisition is Rule or Instance Based is determined by the Structure of the Task

Niels A. Taatgen[1]

University of Groningen, Netherlands

Dieter Wallach

University of Basel, Switzerland

The traditional view of skill acquisition is that it can be explained by a gradual transition from behavior based on declarative rules in the form of examples or instructions towards general knowledge represented by procedural rules. This view is challenged by Logan's instance theory, which specifies that skill acquisition can be explained by the accumulation of examples or instances of the skill. The position defended in this paper is that both types of learning can occur — but their success will depend on the respective task. In the Sugar Factory task, it is very hard to determine the rule guiding the system, rule learning will thus fail while instance learning dominates. In the Anderson-Fincham task, mainly rule learning occurs, but variations in the task show evidence for some instance learning as well. Experiments involving both tasks are modeled using ACT-R, a hybrid cognitive architecture whose adaptive learning mechanisms seem to be well suited for modeling two very different tasks using the same methods.

Keywords: Skill acquisition, instance-based learning, knowledge compilation, cognitive architectures

Introduction

The question whether skills are represented as abstract rule-like entities or as sets of concrete instances taps one of the central distinctions in cognitive science, spreading across fields as diverse as research on memory, problem

[1] Correspondence concerning this article should be addresses to Niels A. Taatgen, Department of Artificial Intelligence, University of Groningen, Grote Kruisstraat 2/1, 9712 TS Groningen, Netherlands. Electronic mail may be sent via Internet to niels@ai.rug.nl

solving, categorization or language learning (Logan, 1988; Hahn & Chater, 1998; Redington & Chater, 1996; Plunkett & Marchman, 1991; Lebiere, Wallach & Taatgen, 1998; Taatgen & Anderson, submitted). Recently, Hahn and Chater (1998) proposed that the distinction between instance- and rule-based learning mechanisms cannot be based on different types of representations, but must involve a framework of their use in problem solving. In this paper we extend their argument and emphasize the necessity of an integrative investigation of human skill acquisition using a comprehensive theory of cognition.

The view of skill acquisition as learning and following abstract rules has dominated theories of skill acquisition over the last decades, whether instantiated as production systems (Newell & Simon, 1972; Anderson, 1993), stored as logical implications (Rips, 1994) or represented in classifier systems (Holland, Holyoak, Nisbett & Thagard, 1986). While these approaches differ on various dimensions, they share the common assumption that cognitive skills are realized as abstract rules that are applied to specific facts when solving problems. In ACT-R, a comprehensive architecture of human cognition (Anderson & Lebiere, 1998), it is assumed that people start out with specific examples or instances of previous problem solving episodes that can potentially be generalized to abstract rules. These rules can be applied in subsequent problem solving and can thus account for increased performance. Discontinuous improvements in cognitive performance as reported by Blessing and Anderson (1996) or Haider (1997) can be taken as evidence for the acquisition of new rules that allow for an increase in observed performance. While Anderson (1993) describes the view that cognitive skills are realized as (production) rules as “one of the most important discoveries” in cognitive psychology, Logan (1988) argues for domain-specific instances as the representational basis for cognitive skills. According to Logan’s instance theory, general-purpose procedures or methods are applied to solve novel problems. Each time such a procedure is used in problem solving, its result is encoded as a separate instance. For new problems, the solution can be calculated using general procedures, or the solution to a previous problem can be retrieved and applied to the current task. Retrieved episodes can be used as a whole, in part, or in adapted versions to obtain solutions of new problems.

One source of empirical support for the instance-based approach is the fact that repeating a certain specific example of a problem increases performance on this example, but not on others (Wallach, submitted). The fact that participants frequently cannot verbalize abstract knowledge about problems solved also seems to provide evidence against some form of generalization as implied by rule-based skill theories. The ACT-R theory that forms the basis of this article, however, assumes that rules themselves cannot consciously be inspected, so this phenomenon is at least not without an

alternative explanation. Further evidence for the fact that knowledge is represented as rules, more specifically production rules, comes from research on the *directional asymmetry* of rules. A production rule consists of two parts, a condition part and an action part, which we informally denote as 'IF condition THEN action'. In a production system, control always flows from the condition to the action, i.e. if the specified condition is met, then the action can potentially be executed. In many practical cases, the condition and the action are both part of a pattern. If we assume, for example, the pattern AB, a rule like 'IF A THEN B' can be used to complete the pattern given A. In an instance approach, the pattern AB can be stored as an instance, and retrieved given either A or B. If participants are trained to complete some pattern AB on the basis of A, a rule approach predicts that they learn the rule 'IF A THEN B', and the instance approach predicts that they learn the instance AB. If participants are consequently asked to complete AB on the basis of B, the instance approach would not predict a significant decrease in performance. In the rule-based approach, with its assumption that control always flows from condition to action, however, the rule 'IF A THEN B' would be useless to complete AB on the basis of B. In that case a new rule 'IF B THEN A' would have to be learned, resulting in severe performance degradations (Kessler, 1988; Rabinowitz & Goldberg, 1995).

Another apparent source of evidence stems from the fact that rules are more general than instances, which are assumed to be represented in a relatively unprocessed form (Redington & Chater, 1996). If participants show increased performance on problems they have not encountered before, some generalized knowledge can be postulated as the basis of the observed performance. If we assume that stored instances can only be applied when the new problem at hand is literally identical to encoded examples, no performance increase on new problems is to be expected. However, if one or more old examples (or fragments of them) can be used to improve performance on a new example in a less direct fashion (Lebiere, Wallach & Taatgen, 1998; Lebiere, 1999), generalization is also possible in an instance-based setting. Consequently, if generalization in transfer experiments is used as evidence against instance theory, it must be ruled out that the answer to a certain problem can easily be derived from stored answers to previous problems. As Redington and Chater (1996) have pointed out, surprisingly simple models, relying on represented fragments of observed stimuli, can perform exceedingly well in transfer tasks without acquiring any abstract knowledge. An example of such a model will be discussed in a later section, when we demonstrate the scope of a purely instance-based approach in accounting for data that Broadbent (1989) has interpreted as evidence against the claim that general rules are learned on the basis of previously stored examples. The results of Broadbent and colleagues on dissociations between knowledge and performance seem to imply that participants can acquire knowledge to

successfully operate complex systems without showing increased scores when answering questions about the system's behavior. The instance-based ACT-R model proposed in this paper will provide a very simple alternative explanation for this dissociation result.

Before presenting models of rule- and instance based skill acquisition in a unified theory of cognition, the next section provides the theoretical framework underlying our approach.

Unified theories of Cognition

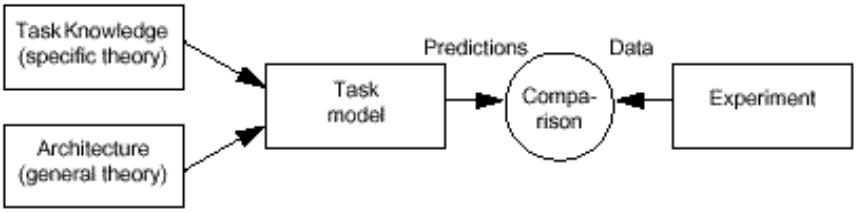


Figure 1. Prototypical research design used in cognitive modeling

In 1990 Newell published the book "Unified Theories of Cognition", in which he elaborated the ambitious vision of a comprehensive theory of cognition, an approach he first outlined in 1973 (Newell, 1973). According to Newell, instead of developing micro-theories for every separate phenomenon, psychology should aim at unification. The ultimate goal, following Newell, is an integrative theory that encompasses a broad set of theoretical approaches: a truly Unified Theory of Cognition. Emphasizing the capability to make detailed predictions, Newell imagined a computational theory that is based on a *cognitive architecture*. A cognitive architecture, in analogy to the architecture of a computer, conceptualizes and implements the structures and mechanisms that are postulated to form the basis of human cognition. To be able to explain cognitive phenomena and to make predictions, such an architecture has to be supplied with a model of a specific task. A task model takes the form of a set of initial knowledge that is encoded in the representational structures provided by the architecture. In the case of a model of expert behavior this may be an extensive body of task-specific knowledge. The task model of novice behavior, on the other hand, might only contain very general knowledge and some specific knowledge that is supposed to be acquired from instructions. Figure 1 illustrates the described research paradigm for cognitive modeling based on cognitive architectures.

In his 1990 book Newell acknowledged the fact that psychology is not ready yet for a single integrated theory. He presented Soar, a candidate ar-

chitecture of his own, but also called upon his fellow researchers to design alternative architectures. Since the development of Soar and Newell's challenge, three other well-known architectures have been implemented on the basis of production systems: EPIC (Meyer & Kieras, 1997), CAPS (Just & Carpenter, 1992) and ACT-R (Anderson, 1993; Anderson & Lebiere, 1998). The models that we discuss in this article rely on the ACT-R architecture. The other two architectures, EPIC and CAPS, do not incorporate learning (yet), and are therefore not suitable for modeling knowledge acquisition processes. Although the Soar architecture does incorporate learning, its pure symbolic nature makes it hard to model the subtle effects of gradual learning and forgetting that characterize the experiments modeled in this paper (see also Taatgen, 1999).

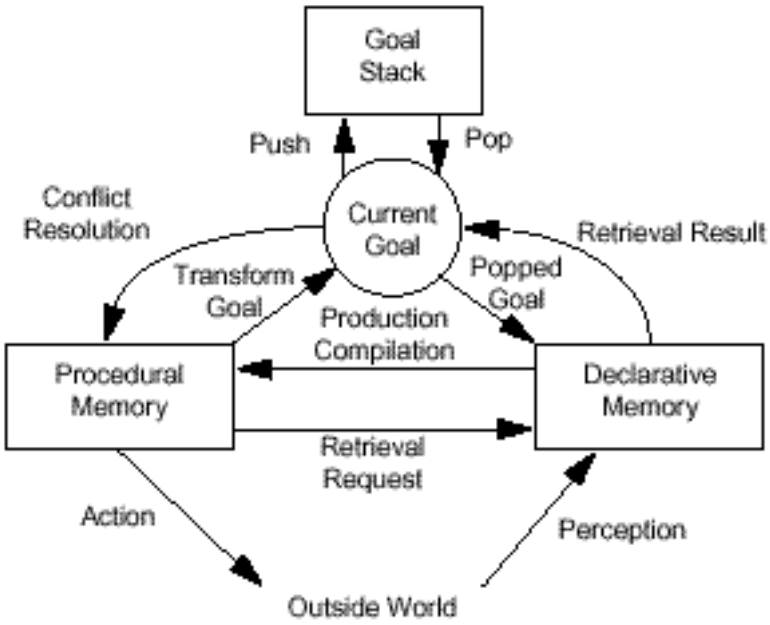


Figure 2. A schematic diagram of ACT-R. Adapted from Anderson & Lebiere (1998).

The ACT-R theory rests upon two important components: rational analysis (Anderson, 1990) and the distinction between procedural and declarative memory (Anderson, 1976). According to rational analysis, each component of the human cognitive architecture is optimized with respect to demands from the environment, given its computational limitations. Rational analysis

assumes that the functioning of an architectural component can be derived by considering how the component in question can work as optimally as possible in a given environment. Anderson (1990) relates this optimality claim to evolution that shapes the architecture. An example of this principle is the way choice is implemented in ACT-R. Whenever there is a choice between strategies to use or memory elements to retrieve, the architecture will take the one that has the highest expected gain, i.e. the choice that has the lowest expected cost while having the highest expected outcome.

The principle of rational analysis can also be applied to task knowledge. While evolution is shaping the architecture, learning shapes the knowledge and parts of the knowledge acquisition processes. Instead of only being focused on acquiring knowledge per se, learning should also aim at finding its right representation. This may imply that learning has to attempt several different ways to represent knowledge, so that the optimal one can be sorted out. This principle will underlie the models we will present later on. Figure 2 gives an overview of the theory, while further details on ACT-R can be found in the appendix to this paper.

Learning strategies

The goal of this paper is to explore instance-based and rule-based learning in an integrative theory of cognition. We assume that a participant in an experiment has some initial method or algorithm to solve a new problem. In most cases this method will initially be time-consuming or inaccurate. Each time an attempt at solving an example of the problem is made, an instance is learned that encodes the respective result. In ACT-R terms an instance is just a goal chunk that is popped from the goal stack and stored in declarative memory. This means that instance-based learning is an automatic process: no conscious effort is needed. Successful use of this knowledge depends on the activation of this knowledge, so it may be necessary to learn the same instance a number of times before it can reliably be retrieved.

Other types of learning require a more active attitude from the participant. If the initial method to solve a problem is too time consuming or produces many errors, the participant may try to derive some sort of rule to increase efficiency. This rule is not a compiled production rule, but is a form of representation that is open to conscious inspection and reasoning. It is rather a declarative structure, which generalizes experience and may state some sort of explicit hypothesis of how things work.

In order to propose and evaluate declarative rules, procedural knowledge is needed. This procedural knowledge has to be general enough to be used in multiple contexts. Once a declarative rule is well established it can be compiled into a production rule.

Production rules that propose new declarative rules to solve a certain type of problem actually put a new layer on architectural learning. The learning that is part of the architecture, ACT-R's built-in mechanisms, is fixed and not open to change. Learning strategies that pose new hypotheses and attempts of generalization are not part of the architecture, but result from acquired knowledge. As a consequence, one can expect large individual differences in learning strategies (Taatgen, 1997).

Our discussion of mechanisms for instance- and rule-based learning raises the following question: if there are different ways of learning, what type of learning will be witnessed in a particular experiment? To answer this question we revisit the principle of rational analysis. According to this principle, we will observe the type of learning that will lead to the largest increase in performance. In a task in which it is very hard to discover valid generalizations and where the set of possible instances is not too large, learning can be characterized primarily by instance learning because this form of learning produces efficient improvements. The next section will discuss an example of a model for a task in which the discovery of such relationships between variables is very difficult and which should — according to the arguments above — be accomplished by instance-based learning. Tasks in which there are too many instances to learn, but in which relationships are more obvious or salient, will probably be better explainable by postulating a form of rule learning (Broadbent & Hayes, 1988). In many tasks, however, both instance and rule learning will be successful to some degree. The model of the Anderson-Fincham task will be an example of this case.

Sugar Factory

In contrast to rule-based approaches that conceptualize skill acquisition as learning of abstract rules, theories of instance-based learning argue that the formation of skills can be understood in terms of the storage and deployment of specific episodes or instances (Logan, 1988; 1990). According to this view, abstraction is not an active process that results in the acquisition of generalized rules, but that rule-like behavior emerges from the way specific instances are encoded, retrieved and deployed in problem solving. While ACT-R has traditionally been associated with a view of learning as the acquisition of abstract production rules (Anderson, 1983; 1987; 1993), we present a simple ACT-R model that learns to operate a dynamic system by retrieving and deploying specific instances (i.e. chunks) that encode episodes experienced during system control. Our ACT-R model will first be compared to a model proposed by Dienes and Fahey (1995) before we apply it to data from a new experiment. Our comparison with the Dienes and Fahey model encompasses both the accuracy of the predictions and the assumptions made by each of the models.

The task

Berry and Broadbent (1984) used the computer-simulated scenario Sugar Factory to investigate how participants learn to operate complex systems. Sugar Factory is a dynamic system in which participants are supposed to control the sugar production *sp* by determining the number of workers *w* employed in a fictional factory. The behavior of Sugar Factory is governed by the following equation:

$$sp_t = 2w_t - sp_{t-1} + random\ component\ (-1,\ 0,\ or\ 1) \qquad [1]$$

The number entered for the workers *w* can be varied in 12 discrete steps $1 = w = 12$, while the sugar production changes discretely between $1 = sp = 12$. If the result according to the equation is less than 1, *sp* is set to 1. Similarly, a result greater than 12 is set to an output of 12. Finally, a random component of ± 1 is added in 2/3 of all trials to the result that follows from the equation stated above. To allow for a more realistic interpretation of *w* as the number of workers and *sp* as tons of sugar, these values are multiplied in the actual computer simulation by 100 and 1000, respectively. Participants are given the goal to produce a target value of 9000 tons of sugar (i.e. $sp=9$) on each of a number of trials. Participants are not informed about the relationship between the number of workers and the sugar production.

The models

Based on Logan's instance theory (1988; 1990), Dienes and Fahey (1995) developed a computational model (coined D&F model in the remainder of this section) to account for the data they gathered in an experiment using the Sugar Factory scenario. According to the instance theory, encoding and retrieval are intimately linked through attention: encoding a stimulus is an unavoidable consequence of attention, and retrieving what is known about a stimulus is also an obligatory consequence of attention. Logan's theory postulates that each encounter of a stimulus is encoded, stored and retrieved using a separate memory trace. These separate memory traces accumulate with experience and lead to a "gradual transition from algorithmic processing to memory-based processing" (Logan, 1988, p. 493).

Both models, the D&F model and our ACT-R model, assume some algorithmic knowledge prior to the availability of instances that could be retrieved to solve a problem. Dienes and Fahey (1995, p. 862) observed that 86% of the first ten input values that participants enter into Sugar Factory can be explained by the following rules:

1. If the sugar production is below (above) target, then increase (decrease) the amount of workers with 0, 100, or 200.
2. For the very first trial, enter a work force of 700, 800 or 900.

3. If the sugar production is on target, then respond with a workforce that is different from the previous one by an amount of -100, 0, or +100 with equal probability.

While this algorithmic knowledge is encoded in the D&F model by a constant number of prior instances that can be retrieved in any situation, ACT-R uses simple production rules to represent this rule-like knowledge. The number of prior instances encoded is a free parameter in the D&F model that was fixed to give a good fit to the data reported below.

Logan's instance theory predicts that every encounter of a stimulus is stored. The D&F model, however, deviates from this assumption in that it only stores instances for those situations in which an action successfully leads to the target. All other situations are postulated as being ignored by the model — an assumption which not only lacks plausibility, but also violates Logan's instance theory that supposedly forms the theoretical foundation of the D&F model. Complicating the modeling basis further, the D&F model uses a "loose" definition of what a successful action is. Due to the random component in the Sugar Factory equation, the outcome calculated by the Sugar Factory formula may vary by ± 1000. Unbeknownst to the participants, a sugar production of 8000 or 10000 is therefore considered to be still on target. Although the information about the loose criterion was not available to participants, the D&F model stored only instances about actions that were successful according to this loose scheme. The ACT-R model, in contrast, encodes every situation that the model encounters, irrespective of its result. The following chunk is an example of an instance stored by the ACT-R model:

Transition1239
    ISA transition
    STATE 3000
    WORKER 800
    PRODUCTION 12000

The above chunk TRANSITION123 encodes a situation in which an input of 800 workers, given a current production of 3000 tons, led to subsequent sugar production of 12000 tons.

Retrieving instances

In the D&F model each stored instance "relevant" to a current situation races against others and against prior instances representing algorithmic knowledge. The fastest instance determines the action of the model. An instance encoding a situation is regarded as "relevant" by the D&F model, if it either

matches the current situation exactly, or if it does not differ from it by more than 1000 tons of sugar in either the current output or the desired output, analogous to the "loose" range discussed above. Again, the D&F model makes use of information unavailable to participants. Retrieval in the ACT-R model, on the other hand, is governed by similarity matches between a situation currently present and encodings of others experienced in the past. On each trial, a memory search is initiated based on the current situation and the target state '9000 tons' as cues to retrieve an appropriate intervention or an intervention that is associated with a similar situation. The following production rule is used to model the memory retrieval of chunks:

IF the goal is to find a transition from the current state with
output *current* to a state with new output *desired*
AND there is a transition chunk in declarative memory, with
output *current* and new output *desired* and a number
of workers equal to number
THEN set the number of workers in the goal to *number*

As outlined in the previous section, chunk retrieval in ACT-R is governed by the activation level of memory elements. Generally, productions try to retrieve an instance that matches the current situation exactly. However, if such a chunk does not exist in memory, or if the activation level of an exact matching chunk is too low, ACT-R can retrieve a memory element that only partially matches the condition pattern of the retrieving production. Chunks that do not exactly match the current situation will be penalized by lowering their activation for each mismatching slot proportional to the degree of mismatch using the following equation:

$$Activation\ \ Penalty = MP \sum_{s=slots\ \ in\ \ matched\ chunk} (1 - sim(required\ s,\ \ actual\ \ s)) \qquad [2]$$

For each slot in the goal that is matched to a slot in the chunk that is a candidate for retrieval, the similarity between their respective contents is calculated. If this similarity is perfect (i.e. *sim*=1), no penalty is subtracted. When the similarity is lower than 1, a corresponding proportion of the fixed parameter *MP* is subtracted from the activation[2].

Figure 3 shows an example for this process of partial matching. In Figure 3 the chunk GOALCHUNK represents a situation in which the current sugar output ("2000") is encoded in the slot STATE, while the target state ("9000")

[2] Following Lebiere (1999), the following function is used to calculate the similarity of two numbers *a* and *b*, representing the sugar production in respective instance chunks: $sim(a, b) = \frac{\min(a, b)}{\max(a, b, 1)}$. Note that the similarity ranges from 0 (no similarity) to 1 (equality).

is encoded in slot PRODUCTION. Chunk EPISODE007 is retrieved by production RETRIEVE-EPISODE using partial matching of the fillers “1000” vs. “2000” in the state slot and the values “8000” vs. “9000” in the PRODUCTION slot. In the action part of production RETRIEVE-EPISODE the number of the WORKER slot of EPISODE007 (“5”) is then used to enrich the current GOALCHUNK by the retrieved value for its WORKER slot.

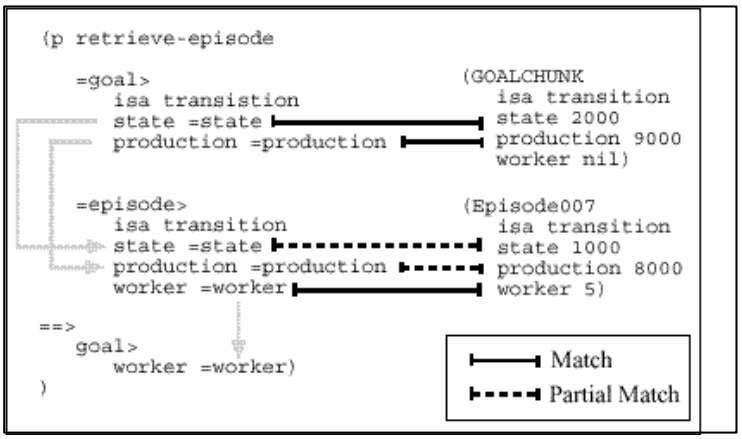


Figure 3. Partial matching in the Sugar Factory model

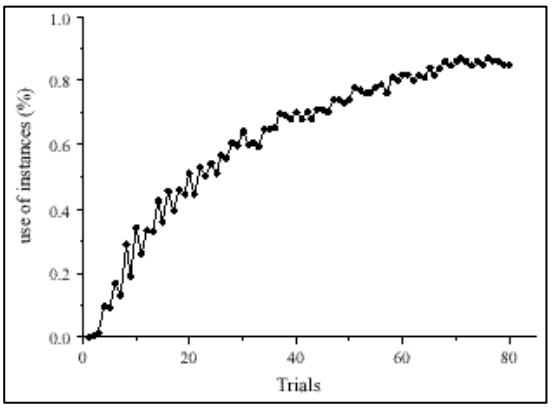


Figure 4. Relative use of instance retrieval per trial by the ACT-R model

As Figure 4 shows, ACT-R's use of instances instead of the initial algorithmic knowledge increases over time, resulting in the gradual transition from algorithmic to memory-based processing as postulated by Logan (1988, p. 493).

While both models of instance-based learning obviously share some striking similarities, the D&F-model makes assumptions regarding the storage and the retrieval of instances that can hardly be justified on either a theoretical or an empirical basis. Dienes and Fahey (1995) point out that these critical assumptions are essential to the performance of the D&F model:

> "The importance to the modeling of assuming that only correct situations were stored was tested by determining the performance of the model when it stored all instances. (…) This model could not perform the task as well as participants: the irrelevant workforce situations provided too much noise by proscribing responses that were in fact inappropriate (…) If instances entered the race only if they exactly matched the current situation, then for the same level of learning as participants, concordances were significantly greater than those of participants."(p. 865).

Since the ACT-R model does not postulate these assumptions, it can be regarded as more parsimonious than the D&F model, demonstrating how instance-based learning can be captured by the elementary mechanisms provided by a unified theory of cognition.

While our theoretical argumentation of the assumptions underlying the two models has favored the ACT-R approach, we will briefly discuss the empirical success of the models with respect to empirical data reported by Dienes and Fahey (1995) before we compare the predictions of the ACT-R model to our own data. Figure 5 shows the trials on target as reported by Dienes and Fahey (1995) when participants controlled Sugar Factory over two phases, consisting of 40 trials each. ACT-R slightly overpredicts the performance found in the first phase, while the D&F model slightly underpredicts the performance of the participants in the second phase. Since both models seem to explain the data equally well, we cannot favor one over the other on empirical grounds.

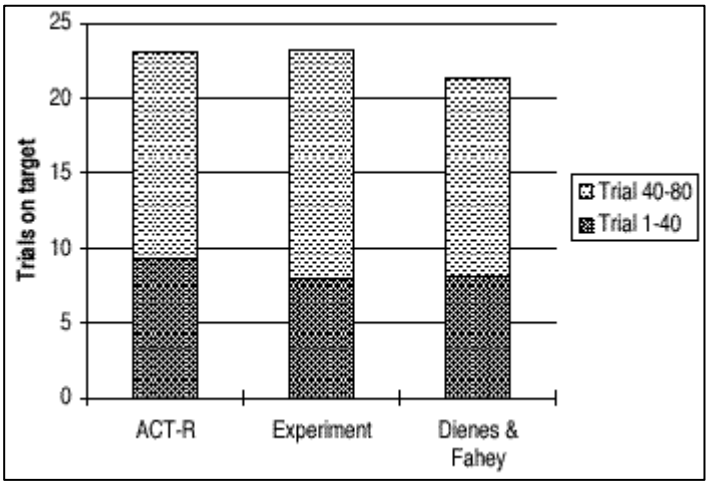


Figure 5. Number of trials on target in the experiment, the ACT-R model and the D&F model for the first and second half of the experiment conducted by Dienes & Fahey (1995)

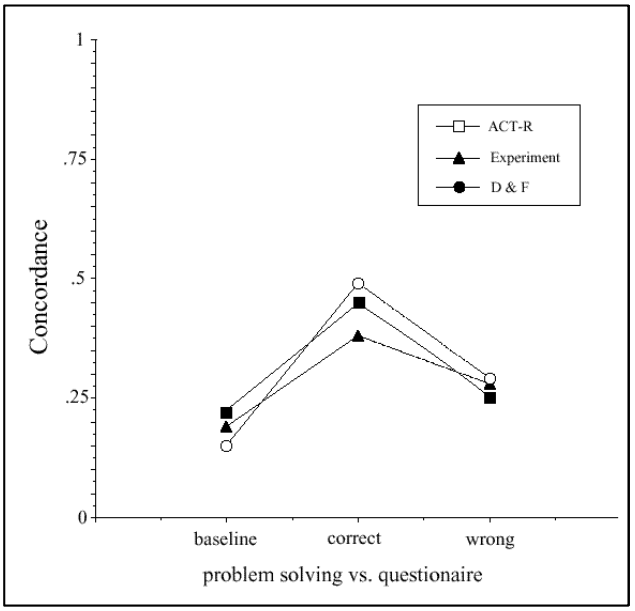


Figure 6. Concordances for the experiment and both models

After the participants had controlled the Sugar Factory in the experiment of Dienes and Fahey (1995), they were required to do a variant of a questionnaire. Again they had to determine the work force in a total of 80 situations, but this time they did not receive feedback, but just moved on to a new, unrelated situation. The situations presented were the last 40 situations from the first part of the experiment mixed with 40 new situations, i.e. situations which participant did not encounter in controlling the system.

Figure 6 shows how the percentage of times (concordance) participants chose the same work force in this second task as they did in the first. The baseline level represents the chance that both choices are equal due to random choice. This chance is higher than 1/12, because some choices are made more often during the experiment than others. The column labeled “correct” shows how often the same work force was chosen if this lead to a correct output, the “wrong” column shows the same for the incorrect outputs. Choices are close to base level for “wrong” answers, while they are significantly higher for “correct” answers, indicating a better memorization of “correct” answers. While this is a trivial consequence from not storing “wrong” instances in the D&F model, no special mechanisms to arrive at this

result are required in the ACT-R model. Generally, both models seem to do similarly well in modeling the concordance data, with no model being clearly superior.

Conclusion

We discussed a simple ACT-R model of instance-based learning and compared it to the Sugar Factory model of Dienes and Fahey (1995) with respect to their respective ability to model the control of a dynamic system. While both models were similarly successful in their empirical predictions, the ACT-R model was found to require fewer assumptions and was thus preferred over the model proposed by Dienes and Fahey (1995). Generally, ACT-R's integration of an activation-based retrieval process with the concept of partial matching seems to be a very promising starting point for the development of an ACT-R theory of instance-based learning and problem solving. The use of partially matching instances allows the architecture to implicitly generalize available examples for application to similar problems. In this view, the reported dissociations between knowledge and performance (Berry & Broadbent, 1984) do not seem surprising at all but are implied by instance-based learning: Participants who are good in controlling Sugar Factory should experience fewer different system transitions (since they are by definition often in the target state) and should thus have encoded fewer instances that would allow them to answer successfully the probes in the questionnaire. Participants who are bad in controlling Sugar Factory, on the other hand, experience a broader set of different system transitions and thus have a better basis of instances for answering questions that probe for system transitions (Funke, Buchner & Berry, 1997).

In the next section we explore the ACT-R model in its ability to account for the data of a new experiment.

Experiment

The goal of the experiment is to explore whether different learning conditions result in different learning strategies and thus in the acquisition of different types of knowledge (i.e. instance vs. rule knowledge). The success of the instance-based ACT-R model on the Sugar Factory task as described in the previous section supports our conjecture stated at the beginning of the paper: instance-based learning seems to be evoked by situations where the problem to be solved involves non-salient relationships between system variables that are very hard to discover: "looking for rules will not work when you cannot find them" (Reber, 1989, p. 232; Broadbent, 1989). The equation underlying the Sugar Factory can be characterized as *non-salient* (Berry, 1991) since it involves a negative feedback component and a random factor in determining the output.

In the experiment reported in this section we manipulate the activity demands when learning to control Sugar Factory with the intention to induce different knowledge acquisition strategies: *Learning by observation* results in the acquisition (and possible elaboration) of examples for demonstrated control behavior while *learning by doing* allows participants to generate, evaluate and test hypotheses about underlying structural relationships by choosing appropriate system interventions. Berry (1991) found that observing a person who interacts with the Sugar Factory in an initial learning phase had no effect on subsequent control performance. In Berry's study, participants watched the experimenter interacting with the task in a learning phase while they had to cope with the scenario themselves in a subsequent knowledge application phase. To make the observed control performance of the experimenter directly comparable with earlier studies conducted by Berry, the experimenter simply typed in responses taken from these studies. Participants in Berry's study were exposed to the very same sequence of input-output-pairs as were the participants in her previous studies (who generated these), but did not have the opportunity to actively provide input values and thus to explore their own hypotheses about underlying system relations in the learning phase. Interestingly, Berry found that participants were only able to learn successfully by observation when there was a salient relationship between input and output variables of the system to be controlled.

Further studies that investigated the effect of system observation on control performance and system knowledge report results that are partially inconsistent with the findings of Berry (1991). Funke and Müller (1988) found that observers were significantly worse in subsequent system control than participants who learned to control the system by doing. However, observers were better in constructing a causal model that expresses the underlying relationships of the system variables and which can be used as an indicator of rule-knowledge acquired. Wallach (1998) also found observers to be significantly better with respect to verbalizable knowledge about the rules that underlie a system's behavior. Control performance showed no differences between the two groups, indicating that exploratory learning and learning by observation led to comparable system control. It should be noted, however, that the systems used in the studies by Funke and Müller (1988) and Wallach (1998) can both be classified as scenarios which are governed by salient relationships between system variables.

One further difference between the studies cited above and the study conducted by Berry (1991) is that in her study the experimenter typed in the input value while the participants watched her interacting with the task. In the experiments conducted by Funke and Müller (1988) and Wallach (1998) observers were directly interacting with the task interface to view input values and the resulting system states. Instead of providing input values on

their own, observers watched the respective values of a "matched" exploratory learning participant whose interaction trace was read from a file and displayed on the interface. In contrast to the setting of Berry (1991), observers were thus in control of the interface — albeit without the opportunity to actively provide input values themselves. This procedure at least partially solves one problem with the setting of Berry: "One problem with studying learning while observing is that it is difficult to have control over what people actually do while observing. Although participants are instructed to watch closely when the experimenter is interacting with the task and do in fact appear to be doing so, there is no real control" (Berry, 1991, p. 905).

In our experiment we applied the exploration/observation procedure used by Funke and Müller (1988) and Wallach (1998) to investigate knowledge acquisition in the non-salient control scenario Sugar Factory for which Berry (1991) found no evidence of learning by observation. If learning by observation is indeed not successful in the case of controlling the non-salient dynamic system Sugar Factory (as argued by Berry, 1991), this can be taken as evidence for important boundary preconditions of successful instance-based learning.

Method

Participants: Participants were 40 (17 female, 23 male) students from the University of the Saarland aged 19 to 31 years (mean = 23.4) who were paid for participating in the experiment.

Materials

Control task: Participants were given the task to control the Sugar Factory. They used the numerical keypad of a computer to provide input values for the number of workers. After pressing a RUN-button on the interface, the result of the respective intervention was displayed. Participants were not given an opportunity to view past system transitions and — in contrast to the setting used by Berry (1991) or Dienes and Fahey (1995) — no graph was displayed that showed the sugar output over past trials.

Post-task questionnaire: In order to access the knowledge that participants acquired while controlling the scenario, 40 questions were provided that probed for state transitions. Questions involved state transition situations that participants experienced when controlling the system (old questions) and questions that comprised state transitions that participants have not experienced before (new questions). In the questionnaire, participants were given the value for the sugar production sp_{t-1} in the preceding trial as well as the current sugar production sp_t and they were asked to determine the intervention w_t that led to the production sp_t. No feedback was given about the correctness of the answer in this phase and participants were informed that each situation described in a question was unrelated to the previous ques-

tion. Because of the random component involved in computing the output signal of the Sugar Factory, the loose scoring scheme of counting a state one off the target was used to judge the correctness of the participants' responses. To ensure a maximal overlap of problem solving context and test context, questions were presented on the computer screen using the same interface for both tasks.

Procedure

Participants were tested individually in single sessions. Their task was to reach the target state, a sugar production of 9000 tons, as often as possible. Participants were not informed about the loose scoring scheme. The scenario was presented to the participants with an initial sugar production of 6000 tons. Participants interacted with the Sugar Factory in two blocks consisting of 40 trials each. They were introduced to block 1 as *a knowledge acquisition* phase while the experimenter referred to block 2 as a *knowledge application* phase. Participants were not informed about the application of a post-task questionnaire in advance. No time restrictions for the fixation of control interventions were imposed. To separate effects of active system control from those of system observation, the following yoked control design was used:

- A system exploration group (SE) was given the opportunity to actively explore the Sugar Factory in block 1, e.g. SE participants could freely select interventions, gather data, test hypotheses about the internal working of the Sugar Factory by changing the values for the workers hired and analyzing the resulting effects on the sugar production.
- In contrast, participants in a system observation group (SO) were restricted to the observation of the control behavior of yoked participants from the SE group that controlled Sugar Factory in block 1. In this block, each SO participant was assigned to a yoked participant and observed exactly the system transitions that the yoked "twin" from the SE group generated. This method of *experimental twins* (Funke & Müller, 1988) ensures that the respective twins are provided with identical data about the behavior of Sugar Factory.

The Sugar Factory interface for the SO condition was exactly the same as for the SE participants, albeit it was not possible to enter values for the work force. Instead, the intervention of a yoked SE participant and the resulting sugar production was displayed when SO participants pressed the RUN-button. The SO participants were informed that what they observed was not necessarily an optimal way to control the Sugar Factory but that they would watch a protocol of what a previous participant had done in an earlier experiment. During the knowledge application phase (block 2), all participants (SE and SO participants) controlled the Sugar Factory scenario.

Results

Control Performance

In contrast to results reported by Berry (1991), both experimental groups, SE and SO participants, were successfully able to learn to control Sugar Factory. No statistically significant difference was found in the control performance of both groups in the knowledge application phase ($t(38)=.563$, $p>.5$; see Figure 7): SE participants achieved a mean performance of 10.2 trials on target in block 2 (SD: 4.17), SO participants scored 9.5 trials on target (SD= 3.66) in this phase. In block 1, SE participants showed a mean performance of 9.55 trials on target (SD= 3.82), SO participants did not control Sugar Factory in this phase. To aid interpretation we ran a simulation study with 100.000 simulated participants to determine chance performance (i.e. each of the 12 possible responses is chosen with equal probability) on the 40 trials with the Sugar Factory task, resulting in an average performance of 4.99 trials on target (SD=2.08).

There was no significant improvement in performance from block 1 to block 2 for the SE participants ($t(19)=.48$). However, a split-half analysis of block 1 indicated that they improved significantly from the first half of block 1 to the second half ($t(19)=2.19$, $p<.05$). Clear evidence for successful observational learning was found by comparing the performance of SE participants in the first half of block 1 with the performance of SO in the first half of block 2. The average number of trials for the SE group was 3.65 (SD=1.93) in the first 20 trials of block 1, SO participant scored an average of 5.0 (SD= 1.55) trials on target in the first half of block 2. The difference is statistically significant ($t=2.25$, $p<.05$). This result is in sharp contrast to the findings of Berry (1991), who reports no evidence for learning by observation when comparing the performance of controlling vs. observing participants in block 1 and block 2, respectively.

To model the empirical data observed in block 2, we ran the instance-based ACT-R model that was introduced in the previous section after individually initializing it for each participant by providing the model with the control episodes that the respective participant generated (or observed in the case of SO participants) in block 1. We thus generated a set of ACT-R models in which each model started with the very same sample of system transitions that the respective SE participant generated in block 1 (and which was observed by the yoked SO participant). This procedure was chosen to insure that the model starts to control Sugar Factory in block 2 with the same information that the respective participants had about the system behavior. The model's single parameter, activation noise, was set to .5 for all runs of the model. As Figure 7 shows, the performance of the resulting sample of model runs is well within the range of performance of the SE and SO group

in block 2 and thus provides evidence in favor of the instance-based learning mechanism of the model.

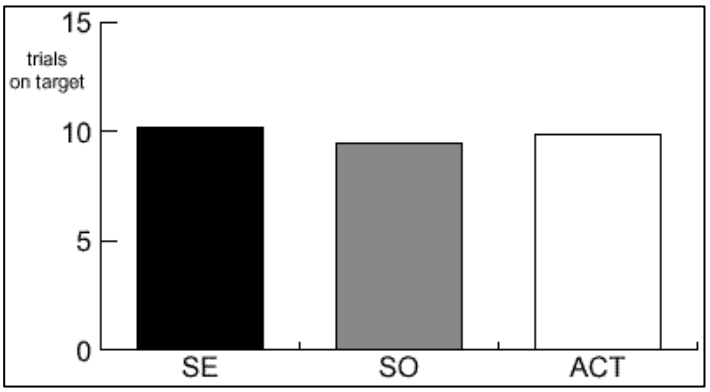


Figure 7. Performance of SE and SO participants and the model in the knowledge application phase

The mean time per intervention in block 1 was 8.06 seconds (SD=4.07) for SE participants, SO participants spend an average of 5.94 seconds (SD=2.02) for the observation of each trial in this phase. The difference between the two experimental groups is statistically significant, ($t(38)=2.08$, $p<.05$). An obvious explanation for this finding is that SE participants had not only to decide for the number of workers to hire but also to enter the respective input values using the keyboard. No significant difference in time per trial was found in block 2: SE participants spend an average of 7.12 seconds per intervention (SD=4.95); the respective mean time for SO participants was 8.07 (SD= 3.56).

Questionnaire data

Participants were generally significantly better in answering questions involving old system transitions in contrast to new transitions ($t(38)=2.48$, $p<.02$). The mean score for old items that were answered correct was 31.75% (SD=18.87) for the SE group and 31.25 (SD=17.0) for the SO group (see Figure 8). 22.20% of new questions were answered correctly by the SE group (SD= 13.53), the respective percentage correct in the SO group was 20.19 (SD=21.81). The different activity demands (system control vs. system observation) in the two experimental conditions thus did not seem to have evoked different knowledge acquisition strategies, i.e. rule vs. instance knowledge) since both groups are equally good in answering new and old questions about system transitions.

To model the questionnaire data, the same individual initialization procedure was used as when modeling the performance of participants in block 2. As Figure 8 illustrates, our simple instance-based ACT-R model reproduces the difference in incorrectness for new vs. old questions quite well,

although it is generally slightly better in answering the questions than participants.

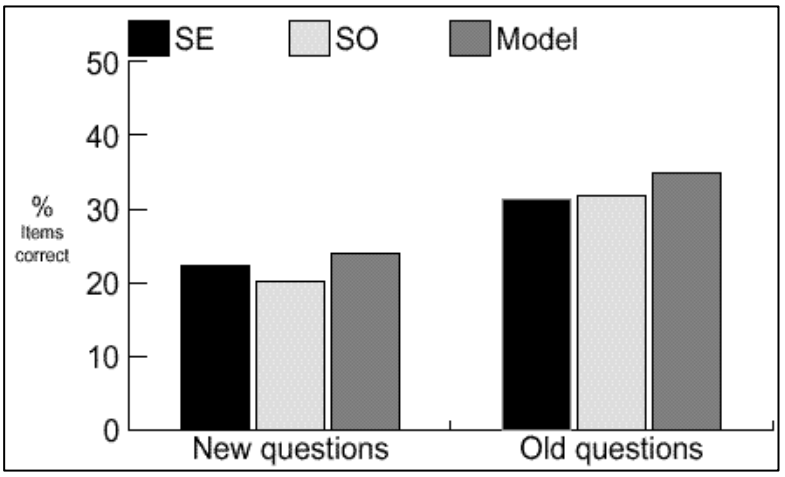


Figure 8. Questionnaire results for new and old questions, for the SE and SO group, and the model

In contrast to results reported in the implicit learning literature (Berry & Broadbent, 1984), we did not find a dissociation between knowledge (as accessed by the questionnaire) and performance (in terms of trials on target in the knowledge application phase): The overall correlation between knowledge of old items and performance was .42 (p<.01). There was no significant correlation between control performance and the ability to answer new items correctly.

Conclusion

Overall, the results from the experiment can be taken as evidence that (1) learning by observation does not seem to be limited in scope to systems for which the underlying relationship between variables is salient; (2) participants seem to be able to answer questions about old system transitions significantly better than questions about new ones — a result that is clearly compatible with an instance-based learning approach, but which provides a challenge for a rule-based view of skill acquisition in this task; (3) It is therefore not surprising that the performance of the simple instance-based ACT-R model is in good correspondence with the empirical data reported.

While instance-based learning as a strategy of knowledge acquisition turned out to be sufficient in the case of Sugar Factory, the next section demonstrates a task for which we claim that it requires a model which integrates instance-based learning with other learning strategies.

The Anderson-Fincham task

An example of a task in which both rule learning and instance learning are viable strategies is described by Anderson & Fincham (1994). In this task, participants first have to memorize a number of facts. These facts are like

"Hockey was played on Saturday at 3 and then on Monday at 1."

We will refer to these facts as *sport-facts* to prevent confusion with facts and rules in the model. A sport-fact contains a unique sport and two events, each of which consists of a day of the week and a time. After having memorized these sport-facts, participants were told that these actually represent rules about the time relationships between the two events. In this case "Hockey" means you have to add two to the day, and subtract two from the time. In the subsequent experiment, participants were asked to predict the second event, given a sport and a first event, or predict the first event, given the sport and the second event. So participants had to answer questions like: "If the first game of hockey was Wednesday at 8, when was the second game?" but also "If the second game of hockey was Thursday at 4, when was the first game?"

Using this paradigm, it is possible to investigate evidence for both rule-based learning and instance-based learning. Directional asymmetry, evidence for rule-based learning, can be tested for by first training participants to predict events in one direction for a certain sport-fact and by then reversing the direction to look how performance in the reverse direction relates to performance on the trained direction. Instance theory would predict no effect between both directions: an instance "Hockey-Saturday-Monday" can be used to both infer Saturday from Hockey and Monday and Monday from Hockey and Saturday. A rule on the other hand has to be reversed first before it can be used in the opposite direction than the direction on which it was trained.

Evidence for instance learning can be gained by presenting specific examples more often than other examples. Better performance on these specific examples would indicate instance learning. Anderson & Fincham (1994), and later Anderson, Fincham & Douglass (1997) performed five variations on this basic experiment, three of which we will discuss here. Before discussing the specific experiments, we will first outline the ACT-R model we have developed.

The ACT-R model

The central assumption of our model of the Anderson-Fincham task is, that the experimental data can only be explained by multiple strategies. In the model we will use the four strategies discussed by Anderson, Fincham & Douglass (1997): *analogy*, *abstraction*, *rule* and *instance*. Analogy is the strat-

egy that participants all have to use at the start of the experiment: they have to recall the original sport-fact, determine the relations between the two days and the two times, and apply these relations to the new case. The abstraction strategy is to memorize the relationships in the examples, for example "hockey day plus two". This saves some time, as it is no longer necessary to determine the relationship in the sport-fact. The rule strategy is similar to the abstraction strategy, except that the knowledge is now part of a production rule instead of a chunk in declarative memory. Finally, the instance strategy tries to recall past instances from memory to solve the problem, for example, if a participant has solved "Hockey Tuesday Thursday" in an earlier trial, they might be able to recall this fact and use it for another trial where Hockey and Tuesday occur (or Hockey and Thursday, if the first event is required). Abstraction corresponds to what we referred to as declarative rules; rule strategy corresponds to production rules. We will therefore refer to the abstraction strategy as the declarative-rule strategy and the rule strategy as the production-rule strategy. The strategies have different cost-success profiles (summarized in Table 1), which will determine at what stage of the learning process they will be most prominent.

To illustrate the respective strategies, Figure 9 shows schematic representations of each of them. Arrows in the figure denote rule firings in the model. In the case of an arrow pointing down, a sub-goal is pushed on the goal-stack, while an arrow pointing up stands for popping a goal. A horizontal arrow symbolizes a modification of the current goal. Since each problem involves calculating a day and a time, two separate sub-problems have to be solved. Each of these sub-problems has to be solved using one of the schemas in Figure 9.

Table 1 .Cost-success profiles of strategies

	Cost (in terms of time)	Additional knowledge needed for each rule	Uses knowledge gained from
Analogy	High	None	-
Declarative rule	Medium	1 declarative rule	Analogy
Production rule	Low	2 rules for each direction	Declarative rule
Instance	Very low	7-9 instances	Any other strategy

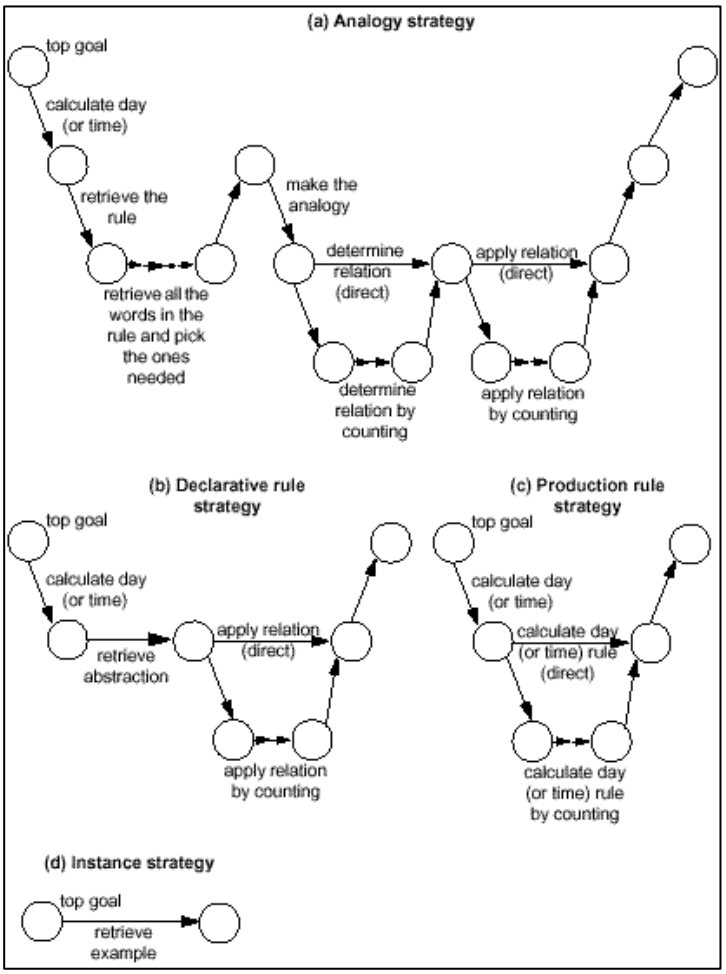


Figure 9. Schematic representation of the four possible strategies used in the model. Note that two strategies (possibly the same) are needed to solve the whole problem: one for the day of the week and one for the time.

The *analogy strategy* (Figure 9a) has the highest cost, but only needs the sport-facts learned initially. Starting at the top goal, a sub-goal is pushed on the goal stack to either find the day or the time. To be able to do this, the original example must first be retrieved, and the appropriate elements (days or times) must be extracted. Another sub-goal takes care of this stage. After retrieving the example, this second sub-goal is popped, and a new sub-goal is pushed to make an analogy between the example and the current problem. First the relation in the example is determined, for example the fact that two has to be subtracted from the day. This relationship can usually be de-

termined by direct retrieval, for example the relationship between four and six. But sometimes, in the case of days of the week, this has to be done by counting. To determine the relationship between Sunday and Friday, a participant has to count two steps back from Sunday. Counting is taken care of by an additional sub-goal, with the advantage that this sub-goal is added to declarative memory and can be retrieved on later trials to determine the relation directly. After determining the relationship in the example, this relation is applied to the current problem. This can again be achieved directly, or through a counting sub-goal.

The analogy strategy requires prior knowledge. The model assumes that people already know how to make simple analogies, how to memorize and recall strings of words, do know relationships between numbers and days of the week, and are able to calculate these relations if they cannot be retrieved from memory. The rest of the necessary knowledge, mainly involving perceptual-motor operations like reading the information on the screen and entering the answers, has to be learned by the participants through the instructions. This aspect of the task is not modeled.

The *declarative-rule strategy* (Figure 9b) assumes knowledge about the relation between the two days or two times for a certain sport. For example, "Hockey" means "add two to the days". A declarative rule in the model is a chunk that stores this information, for example:

RULE34
ISA DEC-RULE
SPORT HOCKEY
TYPE DAY
RELATION PLUS2

Using a declarative rule to find the answer only requires two steps: retrieve the rule and apply it to the current problem. The second step, application, may involve another counting sub-goal, similar to the analogy strategy. Although the declarative-rule strategy is more efficient than the analogy strategy, it requires knowledge participants initially do not have. Declarative rules can be used in both directions: the fact that there is a plus2 relation between the two events can be used to calculate the second event from the first event and vice versa (assuming appropriate plus2 facts are available).

The *production-rule strategy* (Figure 9c) uses production rules to find the answer. Each of the rules has two versions, one that retrieves the answer, and one that calculates the answer. An example of a retrieval rule is:

IF the goal is to find the day of the second event, the sport
is hockey and the day of the first event is *day1*
AND *day1* plus two days equals *day2*

THEN put *day2* in the second event slot of the goal

The calculate production pushes this calculation as a sub-goal, which is handled by the same production rules that determine and apply the relations in the analogy strategy:

IF the goal is to find the day of the second event, the sport is hockey and the day of the first event is *day1*
THEN push as a sub-goal to find the answer to *day1* plus two days
AND put the answer in the second event slot of the goal

The advantage of the production-rule strategy is that its costs are much lower than those of the analogy strategy, and also slightly lower than the costs of the declarative-rule strategy, since the answer can be found in a single step. However, in order to use it, the necessary production rules must be learned. Furthermore, the two example rules given only calculate the second event given the first. To calculate the first event given the second, two additional rules are needed. If we train the rules in only one direction, we can expect that only the rules for that direction will be learned. If the direction is subsequently reversed, productions for the opposite direction are not present yet, so the learner has revert to less effective strategies, producing directional asymmetry.

The strategy with the lowest costs is the *instance strategy* (Figure 9d). It can be applied to the top-goal, since it retrieves the answer from past sub-goals directly. This strategy will only work if the appropriate instance is available. An example of an instance is:

ITEM434
ISA ITEM
SPORT HOCKEY
TYPE DAY
LEFT SUNDAY
RIGHT TUESDAY

To be able to fully depend on this strategy, all possible examples have to be learned. For each sport-fact, seven to nine examples are needed. A partial match strategy, which was very successful in the case of the Sugar Factory, is not very helpful here, as the exact answer is needed.

The declarative-rule, production-rule and instance strategies are actually short cuts for the original analogy strategy. Both rule strategies make short cuts at the sub-goal level of the analogy strategy, and the instance strategy directly at the top level. The knowledge needed for the instance short cut is gained automatically, since the popped sub-goals serve as examples. To be

able to use an example, its activation must be high enough, so it has to be repeated a number of times before it can reliably be retrieved. Declarative rules, on the other hand, have to be learned more explicitly. Once learned, the process of compiling declarative rules into production rules is again automatic.

To create a declarative rule for use in later problems, information from different levels of the goal stack needs to be used. The relation is determined in the analogy sub-goal, while the name of the sport is stored higher in the goal stack. As a consequence, old goals created by the analogy strategy cannot be used as declarative rules. An explicit goal is necessary to assemble it. An appropriate moment to do this is at the end of the analogy strategy. The goal is not popped, but is replaced by a goal to build a declarative rule. Alternatively, the declarative rule could be derived first and be consequently applied. Since this alternative will result in the same empirical predictions, it is not further investigated.

For declarative rule learning additional steps in the reasoning process are necessary that are irrelevant for the immediate solution. The production rule that proposes to create an additional declarative-rule goal has to compete with the rule that proposes to just pop the goal and be done. Since rules that propose additional processing imply additional costs, they will only occasionally win the competition. Building up declarative rules may therefore be a slow process, and may well be a source of individual differences.

Learning a new production rule presupposes a declarative example. The analogy strategy does not provide appropriate examples, because the process of retrieval and consequent analogy is too elaborate. The declarative rule, on the other hand, is a very good basis for proceduralization. Once a declarative rule is formed and applied on a regular basis, it will eventually be compiled into a production rule.

In the Anderson-Fincham task, learning of declarative rules, instance learning and production-rule learning are all viable strategies from the viewpoint of rational analysis. Declarative rule learning will lead to faster results but needs more initial effort, since rules are not learned automatically. Production rule learning will occur automatically, but presupposes declarative rules. Instance learning is eventually the fastest strategy, but requires extensive training to be fully effective.

Empirical evaluation of the model

In order to test the predictive power of the model, three experiments conducted by Anderson, Fincham and Douglass have been modeled. The first experiment was used to determine all the parameters, so the second and the third experiment can be considered as predictions based on the first. Each of the experiments tries to gain insights in the learning process by seeking evidence for the use of rules and the use of instances. The data discussed in the

experiments all come from Anderson, Fincham and Douglass (Anderson & Fincham, 1994; Anderson, Fincham & Douglass, 1997), the model outputs are produced by averaging 100 runs of the model.[3]

Experiment 1

In the first experiment (experiment 2 in Anderson & Fincham, 1994), participants had to learn eight sport-facts. In the first three days of the experiment, four of these sport-facts were tested in a single direction: two from left to right and two from right to left. On each day, 40 blocks of trials were presented. In each block, each of the four sport-facts was tested once. On the fourth day all eight sport-facts were tested in both directions. On this day 10 blocks of trials were presented, in which each of the eight sport-facts was tested twice, once for each direction. This means that on the fourth day there were three types of trials: practiced items tested in the practiced direction, practiced items tested in the reverse direction, and completely new items.

Table 2. ????????

	Data	Model
Same direction, practiced	8.9	8.4
Reverse direction, practiced	10.9	9.3
Not practiced	13	16

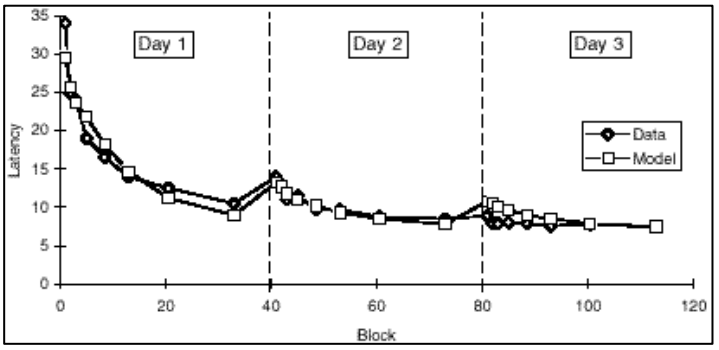


Figure 10. Latencies for days 1 to 3 in experiment 1, data and model.

[3] The model uses the following parameters: base-level decay (d) is set to 0.3, activation noise is set to 0.1, expected gain noise is set to 0.2, the retrieval threshold is set to 0.3 and the latency factor (F) is set to 1.0. Except for base-level decay, all these values are close to their recommended defaults.

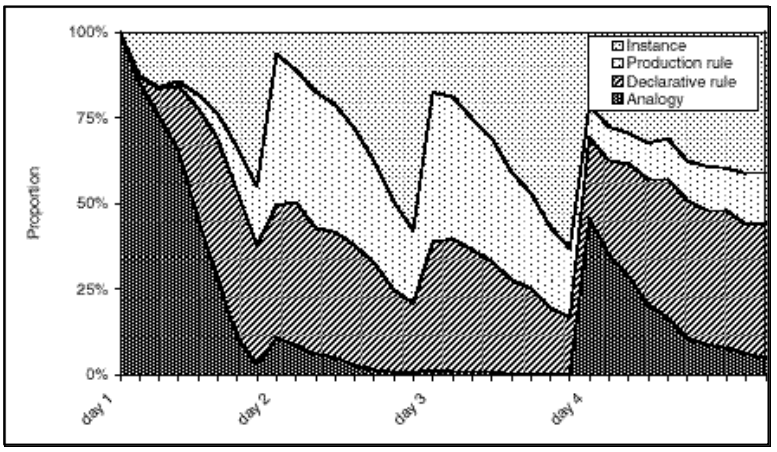


Figure 11. Proportion of the trials a certain strategy is used in experiment 1.

Figure 10 shows the latencies in the first three days of the experiment, both the data from the experiment and from the model. Although the results of the model are the product of four interacting strategies, this produces no discontinuities: the learning curve of the model resembles a power-function, except for a slight decrease in performance at the beginning of each new day. The fit between the model and data is quite good: R^2=0.94. Table 2 shows the results for day 4.

Both the empirically observed data and the model data show a clear directional asymmetry, since items in the practiced direction are solved faster than reversed items. As all the examples are trained in a single direction on days 1-3, one would expect that the rules are only learned in one direction. This becomes apparent on day 4, when rules are presented in both directions.

Figure 11 shows how the model uses the four strategies in the course of the experiment. At the start of the experiment, analogy is used most of the time, but both the declarative-rule and the instance strategies gain in importance after a few blocks of trials. The production-rule strategy appears later, and only plays a minor role during the first day. At the start of the second day, there is a large shift toward using rules at the expense of instances. This can be explained by the fact that the activation of a large portion of the instances has decayed between the two days, so that they cannot be retrieved anymore. Since only a few rules are needed for successful performance, they receive more training on average and are less susceptible to decay. This pattern is repeated at the start of the third day, although the instance strategy looses less ground due to more extended training of the examples. At the start of the fourth day, the frequency of use of the analogy strategy goes up again, since there are no production rules for the new four sport-facts. The

declarative-rule strategy can take care of the reversed items though, so in that case the expensive analogy strategy is not needed. This explains the fact that reversed items are still faster than completely new items.

Experiment 2

In experiment 2 (experiment 1 in Anderson, Fincham & Douglass, 1997) the directional asymmetry was further explored. Instead of having only a single transfer day, two rules were reversed on each day of the experiment. This requires quite a complicated experiment, since on each day a rule has to be presented in two directions which was already presented in one direction previously. So, on day 1 of the experiment, two out of eight rules were presented in two directions, while the remainder was only tested in one direction, on day 2 four out of eight rules, up to day 4 where all rules were presented in both directions. In all cases the rules that were trained in one direction were balanced with respect to direction: in half of the cases the second event had to be predicted from the first event, and in the other half of the cases the first event had to be predicted from the second event. On each day participants had to do sixteen blocks of ten to sixteen trials, ten trials on day 1, twelve trials on day 2, fourteen trials on day 3, and sixteen trials on day 4. To further investigate the difference between rule- and instance-based performance, participants were asked after each trial whether they solved it by using a rule or an example. Finally, on each day one of the sport-facts studied originally was offered as a trial somewhere between block 7 and 10. If performance on this original sport-fact is better than on other trials, this indicates the participant retrieves the answer instead of calculating it.

The model used for experiment 2 is identical to the model used for experiment 1, including all the parameter settings.

The latencies for day 1 to 4 are shown in Figure 12 for both the data and the model. Although the model is slightly slower than the participants, the learning curves are parallel. Directional asymmetries are calculated using the two rules that are presented in two directions for the first time that day. The solution time for the practiced direction is subtracted from the solution time for the reversed direction. The result is the extra time needed for the reversal and is shown in Figure 13. Both the data and the model show a gradual increase in asymmetry over the days, although asymmetry for the model is slightly larger than for the data. To be able to map the participants' reports of using either a rule or an example onto the model, we first have to decide when the model uses a rule or an example. The most logical choice is to assume that both the analogy and the instance strategy are strategies that use examples, and that the declarative and production-rule strategy are strategies that use rules. Figure 14 shows the results of both the model and the data on this aspect of the task. Since the "solve by example"-category includes both the slowest (analogy) and the fastest (instance) strategy, it

eventually becomes faster than the rule strategy, as analogy is not used anymore. Both the data and the model show this phenomenon.

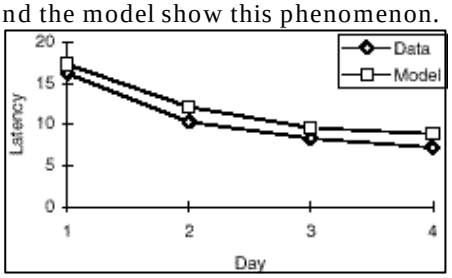
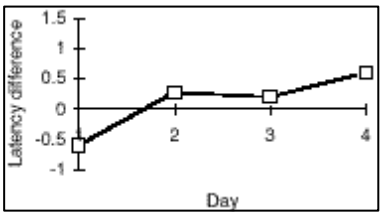


Figure 12. Latencies for experiment 2.

(a)



(b)

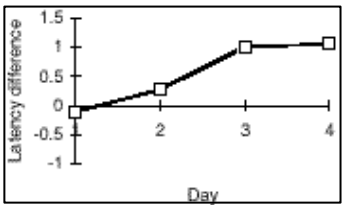
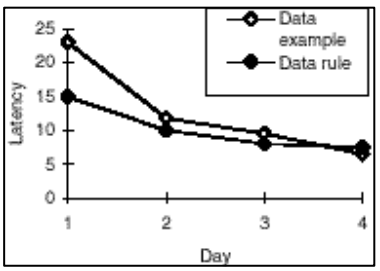


Figure 13. Directional asymmetry in experiment 2, (a) data (b) model.

(a)



(b)

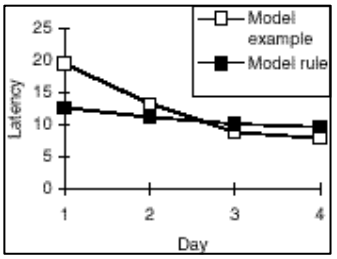
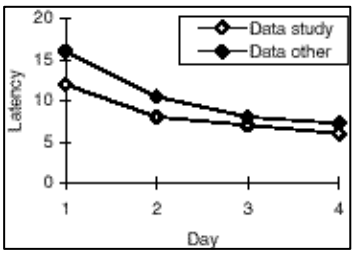


Figure 14. Time to respond as a function of whether a rule or example is reported, (a) data (b) model.

(a)



(b)

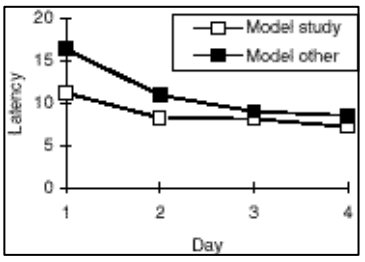


Figure 15. Time to respond for the studied example and other example, (a) data (b) model.

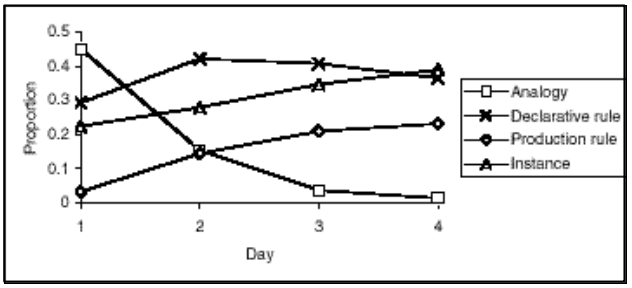


Figure 16. Proportion of the trials a certain strategy is used in experiment 2

The latencies for the original sport-fact that was presented between block 7 and 10 are shown in Figure 15, and are compared with the average latencies between blocks 7 to 10. Performance on original examples is clearly superior to other examples, indicating instance learning. Figure 16, finally, shows the strategies that were used by the model in the course of the experiment. It shows a pattern that is similar to the pattern in experiment 1.

Experiment 3

In experiment 3 (experiment 3 in Anderson, Fincham & Douglass, 1997), the effect of repeated examples is further explored. The same experimental setup as in experiment 2 was used, except that the experiment now took five days and each day consisted of 32 blocks of trials. On the first day eight sport-facts were tested in only one direction. On each subsequent day, a new pair of sport-facts was also tested in the reversed direction. So, on day 2 eight sport-facts were tested in the practiced direction, and two in the reverse direction. On day 3 eight sport-facts were tested in the practiced direction and four in the reverse direction, et cetera. To see if instances that are repeated more often than others are solved faster, half of the instances presented for a certain sport were identical, while the other half was generated in the usual way. Again, the model used for experiment 3 is identical to previous models, including settings for its parameters.

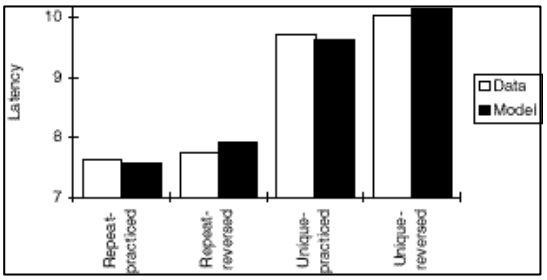


Figure 17. Latency (in seconds) in experiment 3 as function of condition. Only items that are reversed that day are used for these results.

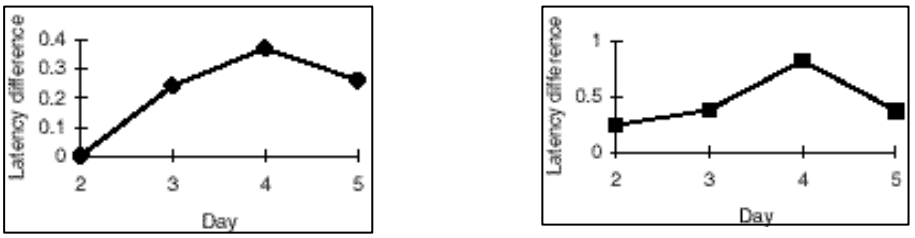


Figure 18. Directional asymmetry in experiment 3.

Figure 17 shows the results for both the data and the model. Repeated instances have a clear advantage over unique instances, which can be taken as further evidence for instance-based learning. Figure 18 shows the directional asymmetry results. After a steady increase between day 2 and 4, it decreases on day 5, both in the model and the data. On day 5, however, both the data and the model show a decline in asymmetry, indicating that instance-based reasoning, which has no asymmetry, takes over from rule-based reasoning.

Discussion

The power of instance learning

Logan (1988) has already shown that instance learning is a powerful method of learning. This was exemplified by the model of the Sugar Factory, where a model that just stores examples can reach the same level of performance as human participants. A question one might ask is whether there is more to skill acquisition than storing and retrieving examples. One would expect that some form of generalization is needed somewhere in the skill acquisition process. Logan (1988) also acknowledged that other forms of learning are possible:

> "I do not intend to argue that instance theory is more correct or more accurate than the other theories. Instead, I view the theories as ad-

> dressing different situations ... humans can learn in more than one way." (p. 148).

Although the Sugar Factory model just stores and retrieves examples, it nevertheless achieves some implicit generalization produced by the partial matching mechanism. By allowing retrieval of instances that are only similar to the current case instead of being equal, instances are generalized: it is no longer necessary to have an instance of every possible situation in order to be able to master the skill.

Explicit generalization

Is all generalization implicit? In the Sugar Factory experiment there is no evidence for explicit rule learning. This is, however, no surprise, as the experiment is carefully constructed to thwart rule learning by limiting control over the input parameters and adding a random component. Another issue in instance theory is the role of the initial method. The theory assumes such a method is available, but it remains unclear how that method is learned. Although in both experiments discussed here the initial methods are either very simple or almost completely specified in the instructions, they are not trivial. Although participants do not know how the Sugar Factory works, they do guess (correctly) that hiring more workers usually implies a higher production.

The Anderson-Fincham experiment explicitly invites participants to generalize, as the instructions spell out the fact that the learned sentences are rules. Also, implicit generalization is unproductive, as the elaborate analogy-strategy is needed to apply one example to another example. Participants therefore generalize by constructing new instances, declarative rules like "Hockey-Day-plus-2". Although a declarative rule is still an instance, it is a very suitable candidate for generalization. In order to move from a relatively unproductive instance-retrieval situation to a productive instance-retrieval situation, the original sentences have to be re-represented. In the present model, this is not an automatic learning process but rather a problem-solving strategy.

Despite the fact that the original examples learned in the Anderson-Fincham task cannot be generalized, retrieving them is still a viable strategy, especially if one example turns up more often in the experiment (experiment 3 is an example of this). A final strategy in the model is the use of actual production rules, compiled from the retrieval and application of declarative rules. In the model, this is the only way to produce directional asymmetry. One might ask in what sense this final step, from declarative to procedural rules, is what really happens in the cognitive system, or whether it is only a necessity in, or an artifact of, the ACT-R architecture. Is there no other model or slightly different architecture in which declarative rules do produce a

directional asymmetry? It is therefore useful to point out that in experiment 2 and 3 it takes a few days before there is any evidence of directional asymmetry. Also, the effect is very small. Participants are assumed to start using declarative rules right away in the experiment, so if these rules produced the effect, we would expect to see it on the first opportunity. Also, we would expect the effect to be larger, as the declarative rule strategy is so much more efficient than the analogy strategy. The difference in efficiency between the production rule and the declarative rule is only small, this is, however, consistent with the observed small experimental directional asymmetries.

The role of rules

The two models presented here show that the role of rules is not as central for cognition as is sometimes thought. In the Sugar Factory experiment rules do not play a role at all. Even in the Anderson-Fincham study, where the use of rules is very explicit in the experiment, instances play an important role. The instance strategy eventually becomes the dominant strategy, and rule generalization itself is achieved by memorizing redescribed instances.

If rules play such a minor role in skill acquisition, can't we somehow eliminate them? In ACT-R, production rules are a necessity, as declarative knowledge itself does not do anything. Also, procedural knowledge plays a major role in the Anderson-Fincham models (and many other models): it implements some general problem solving methods like analogy, and in this case also encodes some task-specific knowledge that is derived from the instructions.

An aspect of skills is that they depend on each other. Before we can learn algebra, we first have to master basic arithmetic. If skills were based on declarative rules alone, the cognitive system would need to juggle facts on different levels of abstraction at the same time. This is quite impossible in ACT-R, so mastery of the lower level skills is a prerequisite for learning higher-level skills. In the current models, mastery of a skill means all the relevant knowledge is proceduralized or all relevant examples are memorized (or a combination of the two).

The structure of the task

In the Sugar Factory experiment, it is very hard to find the rule that governs the behavior of the output. Examples about the behavior of Sugar Factory, on the other hand, prove to be very useful for controlling the system also in circumstances that differ from the exact context in which they were learned. We can see these aspects of the task are reflected in the behavior of the participants, as a model based on exploiting just these aspects simulates human behavior very well.

Although the instructions in the Anderson-Fincham model are very much aimed at rule learning, the structure of the task nevertheless also al-

lows for memorizing instances. As a consequence, participants exhibit both types of learning. If the task allows favoring one strategy, for example in experiment 3 where particular instances are repeated more often, the cognitive system exploits this fact and gears towards the most efficient strategy, in this case instance retrieval.

The general picture that emerges is that several learning strategies are in constant competition. The strategy that performs best will eventually dominate behavior, but this may take a while to settle as the performance profiles of the strategies change due to learning and the passage of time. A theory that only states that skill learning is a matter of both instance learning and rule learning is rather weak, and will certainly not end the debate. Our cognitive modeling approach goes beyond such a simple statement: the theory proposed in this paper is not just the conjunction of two existing theories in an integrative framework, but also adds the constraint that the structure of the task is a main determinant of which types of learning will have an impact on performance. The computational models presented in this paper demonstrate that the ACT-R architecture provides a powerful framework for the theoretical investigation of the interplay of different learning strategies.

The role of parameters

Cognitive models are often criticized by the fact that the outcome of the model is heavily influenced by adjustable parameters. The image that emerges is that of a modeler who can tweak the parameters of his model to produce any outcome he wishes. This is a serious objection to cognitive modeling, as the ACT-R architecture offers many degrees of freedom. In our models we have been very much aware of this problem, so we will discuss our considerations here.

In the Sugar Factory model, the main goal of the model is to show that a very limited model that only stores instances can perform as well as human participants. Although different parameter setting might decrease performance, this is not the point the model tries to make. We applied the model developed for the original Dienes and Fahey data set to a new experiment, without tweaking its subsymbolic parameters. As discussed in the Sugar Factory section, the model was individualized for each single participant with the sample of control episodes that the respective participant generated (or observed in the case of SO participants) before predicting the performance in block 2. While we have only reported aggregate data, we expect interesting results from applying this methodology to model individual learning trajectories.

In the Anderson-Fincham task, there are many adjustable parameters. However, we estimated all these parameters to fit the data of the first experiment, and used these parameters to fit the data in the second and third experiment. As a consequence, the predictions are sometimes slightly off: in

experiment 2 the latencies for the model are all too slow. This could easily have been mended by adjusting one of the parameters, but we elected to keep them fixed. Sometimes this leads to surprisingly good predictions, like the drop-off in directional asymmetry on day 5 in experiment 3.

In this paper we have presented a view on cognitive skill acquisition that takes the structure of a given task into account when theorizing about human learning. While the empirical data has allowed us to arrive at theoretical conclusions about the nature of skill acquisition, cognitive modeling in an integrative theory of cognition gave us the powerful framework necessary to analyze the complex interactions between the different processes and mechanisms involved in human learning.

Appendix: The ACT-R theory

The distinction between procedural and declarative memory is studied quite extensively from different perspectives in psychology and the neurosciences (Anderson, 1976; Squire, 1992). While ACT-R's declarative memory is conceptualized as a store for factual knowledge, elements of procedural memory encode production rules. The ACT-R architecture does not postulate a separate working memory but instead enhances declarative memory by an activation concept (see below) to control access to facts. Only declarative memory elements, so-called *chunks*, above a certain threshold can be retrieved and deployed in problem solving. To keep track of the current context, ACT-R uses a goal stack which organizes the system's intentions. The top element of the goal stack is called the focus of attention, a reference to an element in declarative memory that represents the current goal.

ACT-R's symbolic level

ACT-R comprises two levels of description: a symbolic and a subsymbolic level. On the symbolic level representations in memory are discrete items, and processing applies procedural items to declarative items in the recognize-act-cycle typical for production systems (Waterman & Hayes-Roth, 1976). Declarative memory uses chunks to represent knowledge. A chunk stores information in a propositional fashion and may contain a certain fact, the current or previous goals as well as perceptual information. An example of a goal chunk, in which two is added to six and the answer has not yet been found is:

```
GOAL23
    ISA ADDITION
    ADDEND1 SIX
    ADDEND2 TWO
    ANSWER NIL
```

In this example, ADDEND1, ADDEND2 and ANSWER are slots in chunk GOAL23, and SIX and TWO are fillers for these slots. SIX and TWO are references to other chunks in declarative memory, thus forming an interrelated structure of embedded chunks in declarative memory. The ANSWER slot has a value of NIL, meaning that the slot has no filler, i.e. the answer is not known yet. Let us assume that the chunk GOAL23 designates the current goal. If ACT-R manages to fill the ANSWER slot and focuses its attention on some other goal, GOAL23 will become part of declarative memory and takes the role of a fact representing that six plus two equals eight. GOAL23 can then be retrieved for later problem solving.

Procedural information is represented in production memory by production rules. A production rule has two main components: a condition-part and an action-part. The condition-part contains patterns that match the current goal and possibly other elements in declarative memory. The action-part can modify slot-values in the goal and may create sub-goals (or external actions, which will not be discussed here). In the recognize-act-cycle declarative elements are "matched" to the patterns in the condition-part of a production rule and applied in its action-part.

An example for a rule that tries to solve a subtraction problem by retrieving an addition chunk might look like:

IF the goal is to subtract *num2* from *num1* and there is no answer
AND there is an addition chunk that *num2* plus *num3* equals *num1*
THEN put *num3* in the answer-slot of the goal

This example shows an important aspect of production rules, namely variables. Variables allow for the applicability of a production in a class of situations and thus determine the abstract character of procedural knowledge. The symbols *num1*, *num2* and *num3* of the production shown above denote variables that can be instantiated by any value (with the restriction that same variables have to be bound to the same value). The above rule above can thus find the answer to any subtraction problem, given that the necessary addition chunk is available in declarative memory.

ACT-R's subsymbolic level

The symbolic level provides for the basic building blocks of ACT-R. Using only this level already allows for several interesting models for tasks in which a clearly defined set of rules can be applied — the "Tower of Hanoi" (Anzai & Simon, 1978) being a famous example of such a task (Anderson, Kushmerick & Lebiere, 1993). The symbolic level, however, leaves a number of details unspecified. The main topic that it delegates to the subsymbolic

level is choice. Choices must be made when there is more than one production rule that can match in a situation, or when there is more than one chunk that matches the condition pattern in a production rule. Other matters that are taken care of by the subsymbolic level are accounts for errors and forgetting, as well as the prediction of latencies.

At the subsymbolic level each rule or chunk is annotated with a number of numerical quantities. In the case of chunks, these parameters are used to calculate an estimate of the likelihood that the chunk is needed given the current context. This estimate, called the activation of a chunk, has two components: a base-level activation, which represents the relevance of the chunk by itself, and a context activation through association strengths with fillers of the current goal chunk.

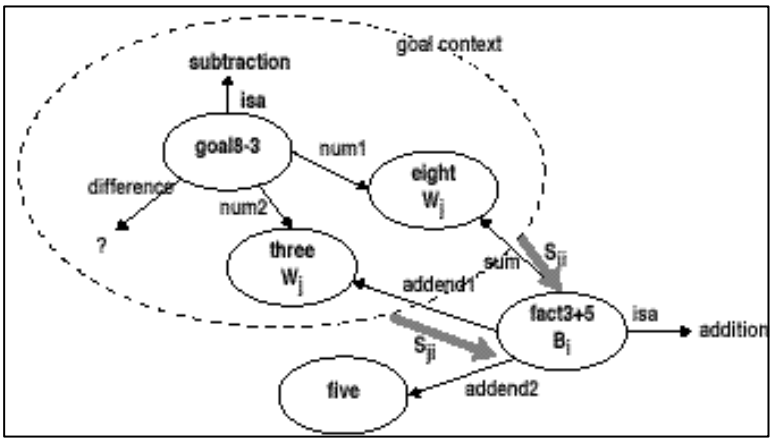


Figure 19. Example of subsymbolic activation in declarative memory. Each oval represents a chunk, arrows represent references between chunks, and the gray arrows represent spreading activation from the goal context.

Figure 19 shows an example in the case of the subtraction problem 8-3=?. The fact that eight and three are part of the context increases the probability that chunks associated with eight and three are needed. In this case 3+5=8 will get extra activation through both three and eight. The activation process is computed by the following equation:

$$A_i = B_i + \sum_j W_j S_{ji} + noise \qquad [A1]$$

In this equation, A_i is the total activation of chunk *i*. This total activation has three parts, *base-level activation* (B_i) and *context activation* ($\sum_j W_j S_{ji}$), and *noise*. As context activation does not play any important role in the models discussed here, we will leave out the details (see Anderson & Lebiere, 1998, p. 71).

The activation level of a chunk has a number of consequences for it's processing. If the activation is below a certain threshold, it cannot be retrieved by patterns in the condition part of a production. If there is more than one chunk that matches the pattern in a production rule, the chunk with the highest activation is chosen. As activation has a noise component, these processes are stochastic: there is no guarantee the chunk with the highest activation will be selected, but it has the highest probability of being retrieved.

Differences in activation levels can lead to mismatches, in which a chunk with a high activation that does not completely match the production rule is selected, while the activation of a completely matching chunk is too low to be retrieved. This process is called partial matching and was discussed in

more detail in the section describing the Sugar Factory model. Finally activation plays a role in determining the latency: the lower the activation of a chunk is, the longer it takes to retrieve it. This retrieval time is calculated using the following equation[4]:

$$Time_i = Fe^{-A_i} \qquad [A2]$$

Choices between production rules are determined by estimates of their expected gain. To calculate the expected gain of a certain rule, several parameters are used to make an estimate. The main equation that governs this estimate is:

$$\text{Expected gain for production } p = P_p G - C_p + noise \qquad [A3]$$

In this equation P_p stands for the estimated chance to reach the goal using production rule p, G is the value of the goal (i.e. the amount of time that the system is willing to spend for the achievement of the current goal), and C_p designates the estimated cost of reaching the goal using p. Again a noise parameter is introduced to allow for some stochasticity in the choice process. The unit of cost in ACT-R is time. Suppose for illustration that a participant is willing to spent 10 seconds on a certain goal (G=10), and let us assume there are two production rules *p1* and *p2* which both match the current goal chunk. Production *p1* reaches the goal in 60% of the cases (P_{p1}=0.6) in 2 seconds on average (C_{p1}=2). Similarly, we assume that P_{p2}=0.8 and C_{p2}=5. In that case the expected gain of *p1* is 4 (0.6·10-2), and the expected gain of *p2* is 3 (0.8·10-5). Given these values, *p1* is selected in favor of *p2* since its expected gain is higher.

In ACT-R representational objects (chunks, productions) are never removed from memory, although they may become virtually irretrievable resp. inapplicable because their activation or expected gain value become too low.

Learning in ACT-R

While ACT-R has two distinct memory systems with two levels of description, distinct learning mechanisms are proposed to account for the symbolic knowledge that is represented as well as for its subsymbolic parameters. At the symbolic level, learning mechanisms specify how new chunks and rules are added to declarative and procedural memory. At the subsymbolic level, learning mechanisms change the values of their numerical parameters.

A new chunk in declarative memory has two possible sources: it is either the encoding of a perceptual object, or a chunk created internally by the goal processing of ACT-R. ACT-R's internally created chunks are always previ-

[4] The symbol F is a scaling parameter that defaults to 1.

ously completed goals, as exemplified by the ADDITION-goal discussed in the earlier example. Consequently, any chunk in declarative memory that did not originate from perception has once been a current goal in ACT-R.

New production rules in ACT-R are learned by a process called *production compilation*. The basis for production compilation is an example chunk in declarative memory. This example chunk encodes how a certain goal is transformed into a new goal. Production compilation generalizes this example and produces a new rule. We will leave out the details of the process here, since they are rather technical and still under theoretical debate (Anderson & Lebiere, 1998, p. 460).

Parameters at the subsymbolic level estimate properties of certain knowledge elements, therefore, learning at this level is aimed at adjusting the estimates in the light of new experience. The general principle guiding these estimates is the well-known Bayes' Theorem (Berger, 1985). According to this principle, a new estimate for a parameter is based on its prior value and evidence from the current situation.

The base-level activation of a chunk estimates the likelihood that it is needed, regardless of the current context. If a chunk was retrieved a number of times in the immediate past, the probability that it will be needed again is relatively high. If a chunk has not been retrieved for a long time, the probability that it will be needed now is only small. Consequently, each time a chunk is retrieved, its base-level activation should go up, and each time it is not used, it should go down. This is exactly what the base-level learning mechanism does: it increases the base-level activation of a chunk each time it is retrieved, and causes it to decay over time. The following equation calculates the base-level activation of a chunk:

$$B_i(t) = \log \sum_{j=1}^{n} (t - t_j)^{-d} \qquad [A4]$$

In this formula, n is the number of times a chunk has been retrieved from memory in the past, and t_j's represent the times at which each of these retrievals took place[5]. The longer ago a retrieval has taken place, the less it contributes to the activation of a chunk.

The other parameters are estimated in a similar fashion, i.e. the probability of success of a production rules goes up each time a production rule leads successfully to a goal, and goes down each time the rule leads to failure.

References

Anderson, J. R. (1976). *Language, memory, and thought*. Hillsdale, NJ: Erlbaum.

[5] The symbol d in the equation is a decay parameter, which defaults to 0.5.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192-210.

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Erlbaum.

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

Anderson, J. R., & Fincham, J. (1994). Acquisition of procedural skills from examples. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 20(6)*, 1322-1340.

Anderson, J. R., Fincham, J. M., & Douglas, S. (1997). The role of examples and rules in the acquisition of a cognitive skill. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 2 (4)*, 932-945.

Anderson, J. R., Kushmerick, N., & Lebiere, C. (1993). The Tower of Hanoi and goal structures. In J. R. Anderson (Ed.), *Rules of the mind* (pp. 121-142). Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Erlbaum.

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review, 86*, 124-140.

Berger, J. O. (1985). *Statistical decision theory and Bayesian analysis*. New York: Springer.

Berry, D. C. (1991). The role of action in implicit learning. *Quarterly Journal of Experimental Psychology, 43*, 881-906.

Berry, D. C., & Broadbent, D.E. (1984). On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology, 36A*, 209-231.

Blessing, S., & Anderson, J. R. (1996). How people learn to skip steps. *Journal of Experimental Psychology: Learning, Memory and Cognition, 22*, 576-598.

Broadbent, D. E. (1989). Lasting representations and temporary processes. In H. L. Roediger, III & F. I. M. Craik (Eds.), *Varieties of memory and consciousness: Essays in honor of Endel Tulving* (pp. 211-228). Hillsdale, NJ: Erlbaum.

Broadbent, D. A., & Hayes (1988). Two modes of learning for interactive tasks. *Cognition, 28*, 249-276.

Buchner, A., Funke, J, & Berry, D. (1997). Negative correlations between control performance and verbalizable knowledge: Indicators for implicit learning in process control tasks? *The Quarterly Journal of Experimental Psychology, 48A*, 166-187.

Dienes, Z., & Fahey, R. (1995). Role of specific instances in controlling a dynamic system. *Journal of Experimental Psychology: Learning, Memory and Cognition, 21*, 4, 848-861.

Funke, J., & Müller, H. (1988). Eingreifen und Prognostizieren als Determinanten von Systemidentifikation und Systemsteuerung [Exploration and prediction as determinants of system identification and control performance]. *Sprache & Kognition, 7*, 176-186.

Hahn, U, & Chater, N. (1998). Similarity and rules: distinct? exhaustive? empirically distinguishable? In Steven A. Sloman & Lance E. Rips (Eds.), *Similarity and symbols in human thinking*. Cambridge, MA: MIT / Elsevier.

Haider, H. (1997). Regelgenerierung beim kognitiven Fertigkeitserwerb [Generation of rules in cognitive skill acquisition]. *Sprache und Kognition, 16*, 183-191.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P.R. (1986). *Induction: Processes of inference, learning and discovery*. Cambridge, MA: MIT Press.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review, 99*, 122-149.

Kessler, C. M. (1988). *Transfer of programming skills in novice LISP learners*. Unpublished doctoral dissertation. Carnegie Mellon University, Pittsburgh, PA.

Lebiere, C., Wallach, D., & Taatgen, N. A. (1998). Implicit and explicit learning in ACT-R. In F. Ritter & R. Young (Eds.), *Cognitive Modelling II* (pp. 183-193). Nottingham: Nottingham University Press.

Logan, G. D. (1988). Toward an instance theory of automatization. *Psychological Review, 95*, 492-527.

Logan, G. D. (1990). Repetitive priming and automaticity: Common underlying mechanisms? *Cognitive Psychology, 22*, 1-35.

Meyer, D. E., & Kieras, D. E. (1997). EPIC: A computational theory of executive cognitive processes and multiple-task performance: Part 1. *Psychological Review, 104*, 3-65.

Newell, A. (1973). You can't play twenty questions with nature and win. In W.C. Chase (Ed.), *Visual information processing*. New York: Academic.

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

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice- Hall.

Plunkett, K., & Marchman, M. (1991). U-shaped learning and frequency effects in a multi-layered perceptron: Implications for child language acquisition. *Cognition, 38*, 43-102.

Rabinowitz, M., & Goldberg, N. (1995). Evaluating the structure-process hypothesis. In F. Weinert and W. Schneider (Eds.), *Memory performance and competencies: Issues in the growth and development*. (pp. 225-242). Hillsdale, NJ: Erlbaum.

Reber, A. S. (1989). Implicit learning and tacit knowledge. *Journal of Experimental Psychology, 118*, 219-235.

Redington, M., & Chater, N. (1996). Transfer in artificial grammar learning: A reevaluation. *Journal of Experimental Psychology: General, 125*, 123-38.

Rips, L. J. (1994). *The Psychology of proof*. Cambridge, MA: MIT Press.

Squire, L. R. (1992). Memory and the hippocampus: A synthesis from findings with rats, monkeys, and humans. *Psychological Review, 99*, 195-231.

Taatgen, N. A. (1997). A rational analysis of alternating search and reflection in problem solving. *Proceedings of the 19th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Taatgen, N. A. (1999). *Learning without limits: from problem solving towards a unified theory of learning*. Doctoral dissertation, University of Groningen, Netherlands. (http://www.ub.rug.nl/eldoc/dis/ppsw/n.a.taatgen/)

Taatgen, N. A., & Anderson, J.R. (submitted). Why do children learn to say "Broke"? Manuscript submitted for publication.

Wallach, D. (1998). *Komplexe Regelungsprozesse. [Complex control processes]*. Wiesbaden: Duv.

Wallach, D. (submitted). Indirect priming in a word arithmetic task. Manuscript submitted for publication.

Waterman, D. A., & Hayes-Roth, F. (1976). *Pattern directed inference systems*. New York: Wiley.