ACT-R Symposium: implicit and explicit learning

Christian Lebiere
Department of Psychology
Carnegie Mellon University
Pittsburgh, USA
cl+@andrew.cmu.edu

Dieter Wallach
Department of Psychology
University of the Saarland
Saarbrücken, Germany
dieter@cops.uni-sb.de

Niels Taatgen
Department of cognitive science and engineering
University of Groningen
Groningen, Netherlands
n.a.taatgen@bcn.rug.nl

ABSTRACT
A useful way to explain the notions of implicit and explicit learning in ACT-R is to define implicit learning as learning by ACT-R's learning mechanisms, and explicit learning as the results of learning goals. This idea complies with the usual notion of implicit learning as unconscious and always active and explicit learning as intentional and conscious. Two models will be discussed to illustrate this point. First a model of a classical implicit memory task, the Sugar Factory scenario by Berry & Broadbent will be discussed, to show how ACT-R can model implicit learning. The second model is of the so-called Fincham task, and exhibits both implicit and explicit learning.

Keywords
ACT-R, implicit learning, explicit learning, skill acquisition, instance theory.

INTRODUCTION TO ACT-R

Knowledge Representation

ACT-R (Anderson, 1993; Anderson & Lebiere, in press) is a hybrid production system architecture for cognitive modeling. It is a hybrid architecture because it works at two interdependent levels: a symbolic level and a subsymbolic level. Each level is divided into a procedural and declarative component.

Symbolic Level

Declarative knowledge consists of chunks. Chunk structures are composed of a number of labeled slots, each of which can hold a value which can also be another chunk. Each chunk is an instance of a particular chunk type, which defines the name and number of slots. Procedural knowledge consists of productions. A production is a condition-action pair, which specifies the action to be taken if a particular condition is satisfied.

ACT-R is a goal-directed architecture. At any time, a goal is selected as the current focus of attention. Goals are organized on the goal stack, on which a goal can be stored (pushed) and later restored (popped). ACT-R operates in discrete cycles. At the start of each cycle, each production is matched against the state of the current goal. The productions that match enter the conflict set. A production is selected from the conflict set. The rest of the production condition can specify a number of chunk retrievals from declarative memory. If the retrievals are not successful, then the next production in the conflict set is selected. If the retrievals are successful, then the production action is executed. The action can modify the current goal, push it on the stack or pop it and restore a previous goal.

Subsymbolic Level

At the symbolic level, ACT-R operates in discrete, deterministic steps, but the subsymbolic level provides a measure of continuity and randomness. The previous section left two points unspecified: how are productions ordered in the conflict set, and if several chunks match a particular declarative retrieval, which is selected?

The productions are selected in order of decreasing expected utility. The current goal is assigned a value, or gain, equal to the worth of successfully achieving it. To each production is associated the probability and cost of achieving the goal to which it applies. The expected utility of a production applied to a goal is equal to the gain of the goal times the probability of success of the production, minus its cost. Noise is also added to the expected utility of a production, making production selection stochastic.

If several chunks satisfy a declarative retrieval, then the most active one is retrieved. The activation of a chunk is the sum of a base-level activation and an associative activation. The associative activation is spread from the sources of activation, which are the components of the current goal, to all related chunks in memory. Noise is added to each activation, making the retrieval of chunks stochastic. If no chunk activation reaches a retrieval threshold, then the retrieval fails. Furthermore, chunks which only partially match the retrieval pattern can also be retrieved, but their activation level will be penalized by an amount proportional to the degree of mismatch between the retrieval pattern and the actual chunk values.

Finally, the time to retrieve a chunk from memory is an exponentially decreasing function of its activation level. Therefore, although ACT-R operates in discrete cycles, the latency of each cycle, which is equal to the sum of the time to perform all the chunk retrievals plus the action time of the successful production, is a continuous quantity. Whereas the specification of an ACT-R model at the symbolic level has a precise, algorithmic quality, its operation at the subsymbolic level matches the stochasticity and continuity of human performance.

Learning

The previous section describes the performance of ACT-R assuming a certain state of knowledge. However, to provide an adequate model of human cognition, it is also necessary to specify how that knowledge was acquired.

In ACT-R, knowledge is learned to adapt the system to the structure of the environment (Anderson, 1990; Anderson & Schooler, 1991).

Symbolic Learning

When a goal is popped, it becomes a chunk in declarative memory. That (and the encoding of environmental stimuli) is the only source of declarative knowledge in ACT-R. The chunk resulting from a goal represents the statement of the task addressed by the goal and usually its solution. Therefore, the next time that task arises, its solution, depending upon the activation of the chunk, might be directly retrieved from declarative memory instead of being recomputed anew.

Productions are created from a special type of chunk called dependency. When a goal is solved through a complex process, a dependency goal can be created to understand how it was solved (e.g. which fact was retrieved or which subgoal was set). When that dependency goal is itself popped, a production is automatically compiled to embody the solution process. Thus the next time a similar goal arises, the production might be available to solve it in a single step instead of a complex process.

Symbolic knowledge is learned to represent in a single, discrete structure (chunk or production) the results of a complex process. Subsymbolic knowledge is adjusted according to Bayesian formulas to make more available those structures which prove most useful.

Subsymbolic Learning

When a production is used to solve a goal, its probability and cost parameters are updated to reflect that experience. If the goal was successfully solved, then the production probability is increased. Otherwise, it is decreased. Similarly, the production cost is updated to reflect the actual cost of solving that goal. Declarative parameters are adjusted in the same way. When a chunk is retrieved, its base-level activation is increased. The strength of association between the current sources and the chunk is also increased.

Subsymbolic knowledge does not result in new conscious knowledge, but instead makes the existing symbolic knowledge more available. Chunks which are often used become more active, and thus can be retrieved faster and more reliably. Productions which are more likely to lead to a solution and/or at a lower cost will have a higher expected utility, and thus are more likely to be selected during conflict resolution.

IMPLICIT LEARNING IN THE SUGAR FACTORY TASK

Introduction

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 behaviour 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; 1993), we present a simple ACT-R model that learns to operate a dynamic system based on the retrieval and deployment of specific instances which encode episodes experienced during system control. It is demonstrated that the ACT-R approach can explain available data as well as an alternative model that is shown to be based on critical assumptions.

The Task

Berry & Broadbent, (1984) used the computer-simulated scenario SUGARFACTORY to investigate how subjects learn to operate complex systems. SUGARFACTORY 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 ficticious factory. Unbeknown to the participants, the behavior of SUGARFACTORY is governed by the following equation:

$$sp_t = 2 * w_t - sp_{t-1}$$

The number entered for the workers *w* can be varied in 12 discrete steps $1 \le w \le 12$, while the sugar production changes discretely between $1 \le sp \le 12$. 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. If the result according to the equation is less than 1000, *sp* is simply set to 1000. Similarly, a result greater than 12000 leads to an output of 12000. Finally, a random component of ± 1000 is added in 2/3 of all trials to the result that follows from the equation stated above. Participants are given the goal to produce a target value of 9000 tons of sugar on each of a number of trials.

The models

Based on Logan's *instance theory* (1988; 1990) Dienes & Fahey (1995) developed a computational model to account for the data they gathered in an experiment using the SUGARFACTORY scenario. According to 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). In the following, we contrast the Dienes & Fahey (1995) model (D&S model) with an alternative instance-based ACT-R model and discuss their theoretical and empirical adequacy.

Algorithmic Processing

Both models assume some algorithmic knowledge prior to the availability of instances that could be retrieved to solve a problem. Dienes & Fahey (1995, p. 862) observed that 86% of the first ten input values that subjects enter into SUGARFACTORY can be explained by the following rules:

(1) If the sugar production is below (above) target, then enter a workforce that is different from the previous input by an amount of 0, +100, +200 (0, -100, -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.

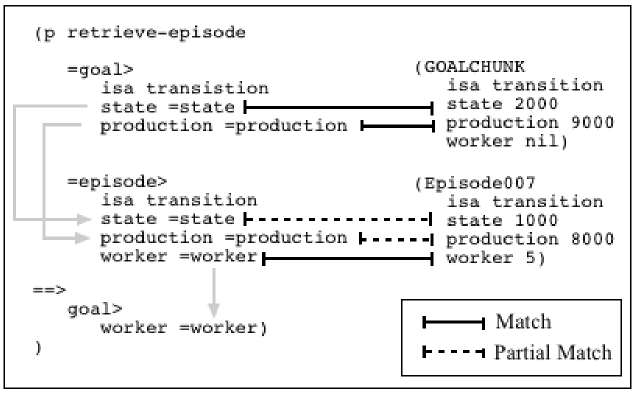
While this algorithmic knowledge is encoded in the D&F model by a constant number of prior instances that could be retrieved in any situation, ACT-R uses simple production rules to represent this rule-like knowledge. The number of prior instances encoded is a free parameter in the D&S model that was fixed to give a good fit to the data reported below. There is no equivalent parameter in the ACT-R model.

Storing Instances

Logan's instance theory predicts that every encounter of a stimulus is stored. The D&F model, however, does only store instances for those situations, in which an action successfully leads to the target, all other situations are postulated to be forgotten immediately. Moreover, the D&S model uses a „loose" definition of the target that was unavailable to subjects: While subjects were supposed to produce 9000 tons of sugar as *the* target state in the experiment, a loose scoring scheme was used to determine the performance of the subjects. Because of the random component involved in the SUGARFACTORY, a trial was counted as being on target if it resulted in a sugar production of 9000 tons with a tolerance of ±1000. The D&M model stores only instances that are successful in this loose sense and thus uses information about a range of target states that subjects were not aware of. ACT-R, on the other hand, encodes every situation, irrespective of its result The following chunk is an example for an instance acquired by the ACT-R model as a restored goal.

```
(transition1239
    ISA transition
    STATE 3000
    WORKER 8
    PRODUCTION 12000)
```

The chunk encodes a situation in which an input of 8 workers, given a current production of 3000 tons, led to subsequent sugar production of 12000 tons. While the model developed by Dienes & Fahey (1995) stores multiple copies of instances, ACT-R does not dublicate identical chunks.



```
(p retrieve-episode

   =goal>                            (GOALCHUNK
      isa transistion                   isa transition
      state =state                      state 2000
      production =production            production 9000
                                        worker nil)

   =episode>                         (Episode007
      isa transition                    isa transition
      state =state                      state 1000
      production =production            production 8000
      worker =worker                    worker 5)
==>
   goal>
      worker =worker)
)
```

Figure 1. Matching process in the Sugar Factory model

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 first instance after a finishing post determines the action of the model. An instance encoding a situation is regarded to be „relevant", if it either matches the current situation exactly, or if it is within the loose range discussed above. As with the storage of instances, memory retrieval in the D&F model is based on specific information not available to subjects. 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 (see Buchner, Funke & Berry, 1995 for a similar position in explaining the performance of subjects operating SUGARFACTORY). On each trial, a memory search is initiated based on the current situation and the target state '9000 tons' as cues in order to retrieve an appropriate intervention or an intervention that belongs to a similar situation. The production rule `retrieve-episode` (figure 1) is used to model the memory retrieval of instances based on their activation

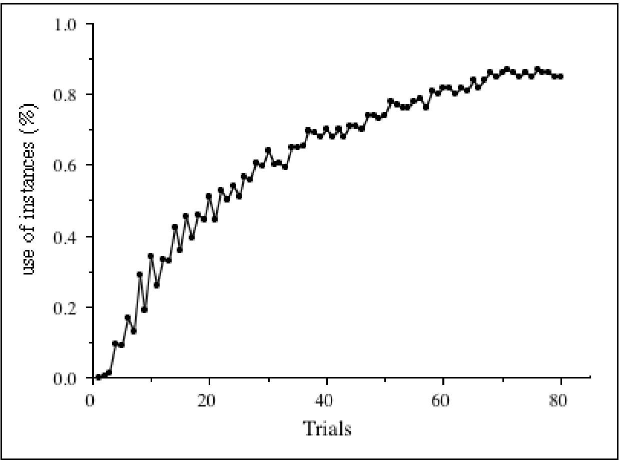


Figure 2. Relative use of instance retrieval per trial

level. Instances which only partially match the retrieval pattern, i.e. which do not correspond exactly to the present situation, will be penalized by lowering their activation proportional to the degree of mismatch. As a parameter of the ACT-R model, normally distributed activation noise is introduced to allow for some stochasticity in memory retrieval.

As figure 2 shows, the use of instances over 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).

Theoretical Evaluation

While both models of instance-based learning share some striking similarities, the theoretical comparison has shown that the D&F-model makes stronger assumptions with respect to the storage and the retrieval of instances that seem to be hard to justify. Dienes & Fahey (1995) found 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

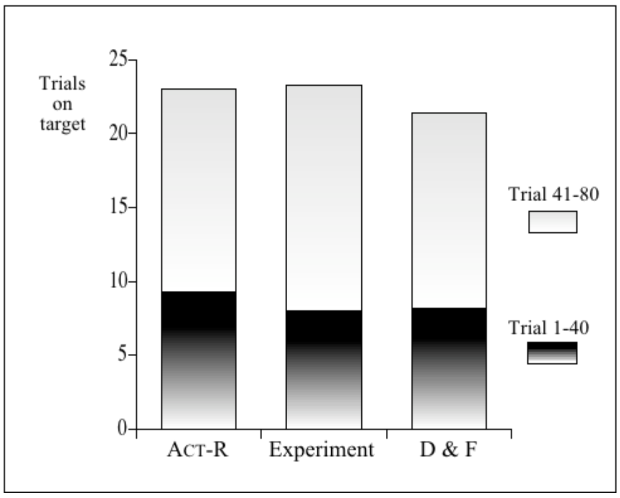


Figure 3. Results of the experiment, ACT-R model and D&F model

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 appropriate … 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. 856f).

Empirical Evaluation

While the theoretical analysis of the assumptions underlying the two models has favoured the ACT-R approach, we will briefly discuss the empirical success of the models with respect to empirical data as reported by Dienes & Fahey (1995). Figure 3 shows the trials on target when controlling SUGARFACTORY 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 subjects in the second phase. Since both models seem to explain the data equally well, we cannot favour one over the other.

Figure 4 shows the performance of the models in predicting the percentage of times („Concordance“) that the subjects gave the same (correct or wrong) response in a questionaire as they did when controlling the SUGARFACTORY. Again, both models seem to do a similar good job in explaining the data, with no model being clearly superior. Although space limitations do not allow for a detailed discussion, the picture illustrated by these two empirical comparisons remains the same after several additional model comparision tests.

Conclusion

We discussed and compared a simple ACT-R model to an approach based on Logan’s instance theory with respect to their ability to modeling the control of a dynamic system. While both models were similar in their empirical predictions, the ACT-R model was found to require fewer assumptions and is thus preferred over the model proposed by Dienes & Fahey (1995). Generally, ACT-R’s integration of an activation-based retrieval process with a partial matcher seems to be a very promising starting point for the development of an ACT-R theory of instance-based learning and problem solving.

IMPLICIT AND EXPLICIT LEARNING IN THE FINCHAM TASK

The learning mechanisms in ACT-R are all quite basic, and can be used in several different ways to achieve different results. The idea of a learning mechanism as an integral part of an architecture has properties in common with the psychological notion of implicit learning. Both types of learning are considered to be always at work and not susceptible to change due to development or great variation due to individual differences. One of the defining properties of implicit learning, the fact that it is not a conscious process, is harder to operationalize within the context of an architecture for cognition. The closest you can get in an architecture is the notion that implicit learning is not guided by learning intentions, but is rather a by-product of normal processing. The Sugar Factory model discussed in the previous section is an example of implicit learning, since ACT-R uses old goals that are stored unintentionally to improve its behavior.

Explicit learning, on the other hand, is tied to intentions, or goals in ACT-R terms. Since there are no learning mechanisms that operate on goals, explicit learning can best be explained by a set of learned learning strategies. An example of a learning strategy to improve memorization of facts is using rehearsal to improve base-level learning. Base-level learning increases the activation of a chunk each time it is retrieved. If this increase of activation through natural use is not enough for the current goals, rehearsal can be used to speed up the process. By repeating a fact a number of times, its base-level activation can be boosted intentionally.

In this section we will discuss a paradigm for skill learning that involves both an implicit and an explicit strategy. The implicit strategy corresponds to instance-based learning, and the explicit strategy to rule-learning.

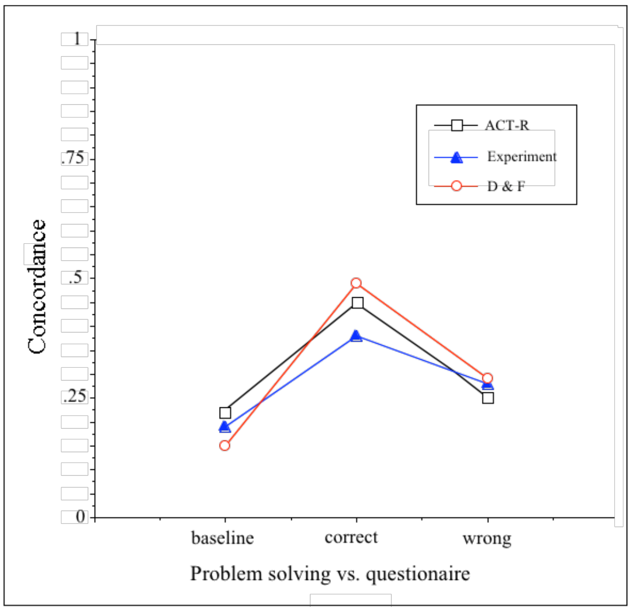


Figure 4. Concordances for the experiment en both models

Figure 5 shows an overview of this paradigm. First we assume that a participant has some initial method or algorithm to solve the problem. Generally this method will be time-consuming or inaccurate. Each time an example of the problem is solved by this method, an instance is learned. In ACT-R terms an instance is just a goal that is popped from the goal stack and is stored in declarative memory. Since this by-product of performance is unintentional, it can be considered as implicit learning.

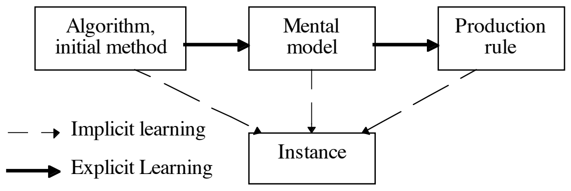


Figure 5. Diagram that illustrates the learning scheme used in the Fincham-task model

Other types of learning require a more active attitude from the participant. If the initial method is too time consuming, the participant may try to derive an re-representation of the information needed for the task to increase efficiency, which we will call, using Johnson-Laird's (1983) terminology, a mental model. If the initial method leads to a large number of errors, the participant may try to deduce or guess new relationships in the task in order to increase performance. The next step, from mental model to production rule, can only be made if the mental model is simple enough to convert to a production rule. Both the application of mental models and firing new production rules will create new instances. So regardless of what is going on due to explicit learning, implicit learning keeps accumulating knowledge.

So, if we have that many ways of learning, what type of learning will we witness in a particular experiment? To be able to answer this question we go back to the principle of rational analysis. According to this principle, we will principally witness that type of learning that will lead to the largest increase in performance. If we have task in which it is very hard to discover relationships or mental models, learning can probably be characterized primarily by implicit instance learning. Tasks in which there are too many instances too learn, but in which relationships are more obvious, will probably be better explainable by rule and abstraction learning. The Sugar Factory task is an example in which it is very hard to discover the rules the govern the system due to the random factor in the output and the fact that only one of the inputs that determine the output can be manipulated.

The 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 in the form of

"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 facts, participants were told the facts are really rules about the time relationships between the two events. So 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?" In 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 a sport-fact in one direction (by predicting the second event using the sport and the first event), and then reverse the direction (by predicting the first event using the sport and the second event) and look how performance in the reverse direction relates to performance on the trained direction. If the performance is worse in the reverse direction, this is evidence for the use of rules. 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. The basic findings we will focus on are as follows:

- In general, reactions times improve according to the power law of practice, starting at around 35 seconds for the first few trials and improving to around 7 seconds at the third session.
- There is evidence for rule learning as witnessed by directional asymmetry. However, the effect only starts at the third or fourth session, and is relatively small.
- There is evidence for instance learning, since problems that are repeated more often than others are solved faster.
- Although it can not be inferred directly from the data, participants report they use abstract versions of the rules, for example by memorizing "Hockey day +2" and "Hockey time -2".

On basis of this evidence, Anderson et al. conclude that participants use four strategies: analogy, abstraction, rule and instance. The interesting question is what learning processes play a role in changing strategies. Each of the four strategies can be related to one of the learning stages from figure 5.

The analogy strategy is the initial strategy: first the memorized example that has the same sport as the new trial is recalled, the relationship in this example is determined, and this relationship is mapped on the current trial. Analogy is not very efficient, since it consists of many steps.

The abstraction strategy assumes the participant has created and memorized a mental model of the sport that

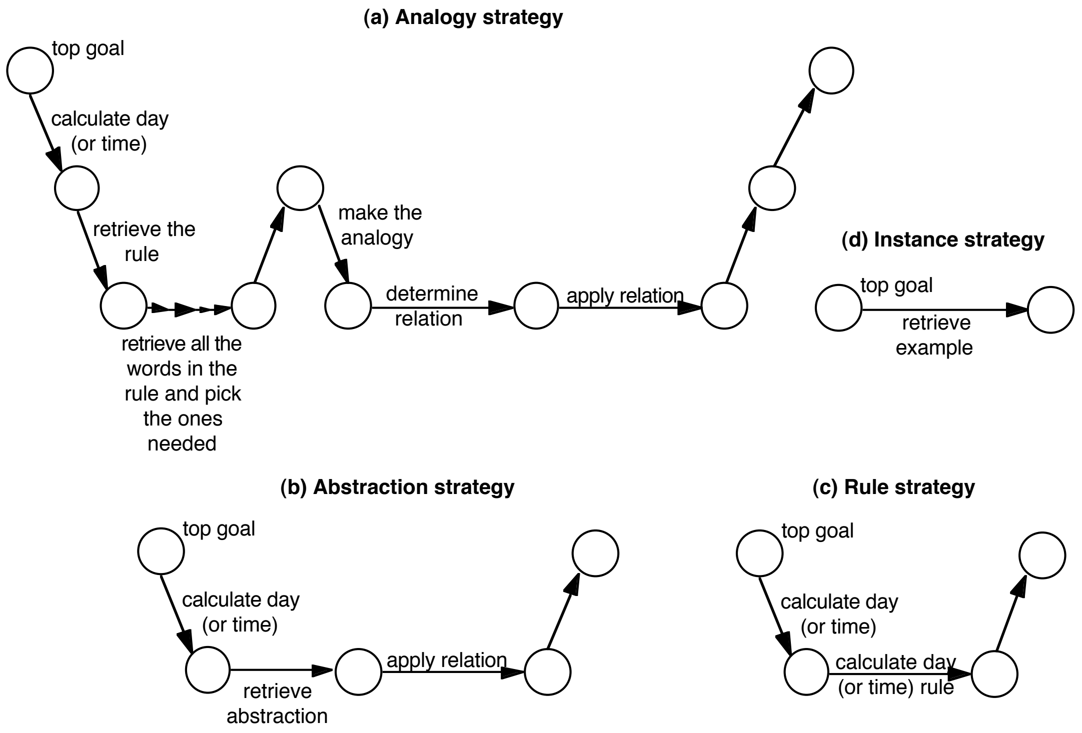


Figure 6. Overview of the four strategies in the Fincham task as modeled in ACT-R

corresponds to the current trial, like “Hockey day +2”. The strategy involves retrieving and applying the abstraction, which is easier and faster than the analogy strategy.

The rule strategy assumes a production rule has been learned that can fill in the answer directly. An example of this rule is (variables are indicated by italics):

```
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 rule strategy is more efficient than the abstraction strategy, since it requires only a single step in stead of two.

The instance strategy assumes the answer can be given using a previous example. This previous example must be the same as the current trial. So an instance may contain the following information:

```
item1434
    isa instance
    sport hockey
    type day
    left sunday
    right tuesday
```

To use the instance strategy, it is sufficient to retrieve the right instance. This will of course only succeed if this instance is present in memory and is retrievable.

An ACT-R Model

We will now briefly discuss the ACT-R model of the task and its results. A more extensive discussion can be found in Taatgen & Wallach (in preparation). Figure 6 shows a schematic diagram of the implementation of the four strategies.

The analogy, abstraction and rule strategies are performed in a subgoal, that focuses on calculating either the day or the time. The instance strategy attempts to retrieve one of these subgoals, and fill in the answer directly in the topgoal. So learning instances is an implicit process in ACT-R, since past goals are always stored in declarative memory, an reoccurrence of the same goal just increases the activation of that goal. Knowledge for the other two strategies has to be acquired in an explicit fashion. An abstract mental model of a sport is no automatic by-product of the analogy strategy, so an explicit decision must be made to memorize an abstraction. To learn a new production rule in ACT-R, a special dependency structure must be created in declarative memory, which is also an explicit decision. In the current model, learning a new production rule is only successful if there is already an abstraction present in declarative memory, else it is too difficult to collect the necessary information.

Results of the Model

In this paper we will only discuss results of the model on the second experiment of Anderson & Fincham (1994). In this experiment, 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 which 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. Figure 7 shows the latencies in the first three days of the experiment, both the data from the experiment and from the model. The fit between the model and the data is quite good (R^2=0.94).

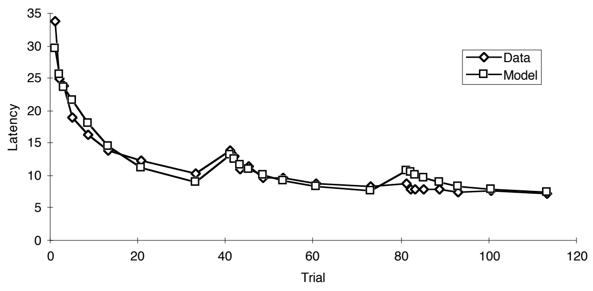


Figure 7. Latencies in experiment 1 for days 1-3

The results on day 4 can be summarized in the following table:

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

Both in the data and in the model there is a clear directional asymmetry, since items in the practiced direction are solved faster than reversed items. The fact that unpracticed items are slower than the reversed items indicates that rule learning can not be a sufficient explanation for all of the learning in the first three days of the experiment.

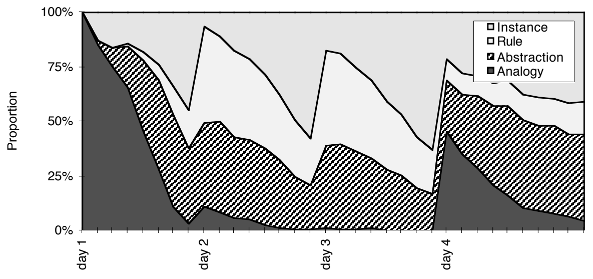


Figure 8. Strategy use in experiment 1 for days 1-4

Figure 8 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 abstraction and the instance strategy gain in importance after a few blocks of trials. The rule strategy only 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 can not be retrieved anymore. Since only few rules are needed for successful performance, they receive more training on average and are less susceptible to decay. Note that the abstraction strategy remains relatively stable between the days since it also less susceptible to decay than the instance strategy. 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 abstraction 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.

Except for a model of this experiment, the model has successfully modeled two other experiments as well, using the same parameters. The following additional phenomena could successfully be explained:

- The reaction time for examples that are repeated more often is shorten, since instance learning is more successful and the facts it represents have a higher activation.
- Directional asymmetry increases between day 2 to 4, but decreases again on day 5. The model can explain this by the fact that by day 5 the instance strategy starts dominating the rule strategy.
- The results of the model concur with participant's reports on whether they use a rule or an example to solve a particular trial.

Conclusions

The ACT-R architecture is an ideal platform to study implicit and explicit learning. It not only allows insights in both types of learning separately, but, more importantly, also in the interaction between them.

REFERENCES

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.M. (1994). Acquisition of Procedural Skills From Examples. *Journal of experimental psychology: Learning, Memory, and Cognition*, vol. 20, no. 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, vol. 23, no. 4, 932-945.

Anderson, J. R. & Lebiere, C.. (in press). *The atomic components of thought.* Mahwah, NJ: Erlbaum.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science, 2,* 396-408.

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

Buchner, A., Funke, J. & Berry, D. C. (1995). 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), 848-862.

Johnson-Laird, P.N. (1983). *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge, MA: Harvard University Press.

Logan, G.D. (1988). Toward an instance theory of automatization. *Psychological Review*, 95, 492-528.

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

Taatgen, N.A. & Wallach, D. (in preparation). Models of rule and instance-based skill learning.