Explicit Learning in ACT-R

Niels Taatgen
Cognitive Science and Engineering
University of Groningen, the Netherlands

Abstract

A popular distinction in the learning literature is the distinction between implicit and explicit learning. Although many studies elaborate on the nature of implicit learning, little attention is left for explicit learning. The unintentional aspect of implicit learning corresponds well to the mechanistic view of learning employed in architectures of cognition. But how to account for deliberate, intentional, explicit learning? This chapter argues that explicit learning can be explained by strategies that exploit implicit learning mechanisms. This idea is explored and modelled using the ACT-R theory (Anderson, 1993). An explicit strategy for learning facts in ACT-R's declarative memory is rehearsal, a strategy that uses ACT-R's activation learning mechanisms to gain deliberate control over what is learned. In the same sense, strategies for explicit procedural learning are proposed. Procedural learning in ACT-R involves generalisation of examples. Explicit learning rules can create and manipulate these examples. An example of these explicit rules will be discussed. These rules are general enough to be able to model the learning of three different tasks. Furthermore, the last of these models can explain the difference between adults and children in the discrimination-shift task.

Introduction

One of the basic assumptions all architectures of cognition share is that all learning can be described with a fixed set of mechanisms. The term 'mechanism' refers to the fact that learning is unintentional and is always at work. The term 'fixed' refers to the fact that learning never changes, and is the same for each person, regardless of age or intelligence. This view of intelligence seems to be at odds with the general view of learning in psychology. The hallmark of learning is adaptation, the capacity of the organism to change its behaviour to suit a particular environment. This does not necessarily imply that learning itself is susceptible to adaptation. But work from developmental psychology clearly suggests that learning changes with age. One classical experiment that shows the way children learn differs from adults is discrimination-shift learning (Kendler & Kendler, 1959). We will discuss this experiment in more detail later on. A second counter-intuitive aspect of learning mechanisms is the fact that learning is unintentional.

Although we have no complete control over learning, the idea that we have no control at all seems to be too strong.

The distinction between implicit and explicit learning is centred around this issue. Implicit learning is unconscious and unintentional, so is consistent with the mechanistic and fixed view of learning in architectures of cognition. In explicit learning, on the other hand, goals and intentions determine what is learned. Moreover, many studies suggest that explicit learning is much more powerful than implicit learning (see for an overview Shanks & John, 1994). Some things can't be learned by implicit learning, but can be learned by explicit learning. Finally, things that can be learned implicitly can often be learned faster explicitly.

Another aspect of learning that seems to be at odds with mechanistic learning is learning through insight, as discussed in section 4 of this book. Many learning mechanisms model gradual learning. Learning in the PDP neural network tradition is gradual (Rumelhart & McClelland, 1986), and chunking in Soar is inspired by, and can explain the power law of practice (Newell & Rosenbloom, 1981). A property of insight learning is a sudden qualitative shift. Take for example match stick algebra (MSA), as discussed in Knoblich and Ohlsson (1997). In MSA, the insights subjects gain concern a number of constraints that are part of normal algebra. The subjects have to discover that these constraints may be violated in MSA. Knoblich and Ohlsson conclude that once a subject has relaxed a certain constraint, it stays relaxed. In other words, the subject has learned something about a certain constraint in the context of MSA. This type of learning is not gradual, but step wise.

So the central question of this chapter is how learning mechanisms in architectures of cognition can be made consistent with an adaptive view of learning that allows for flashes of insight. Or, stated in another way, we need a theory of explicit learning. We will concentrate the discussion on the ACT-R architecture (Anderson, 1993), but some aspects may apply to other architectures as well.

Since mechanisms in architectures are fixed, we have to seek for other ways to explain adaptation in learning. The only thing that changes over time in an architecture is the knowledge in its memory. An explanation of changes in learning has to be an explanation in terms of changes in the content of memory. As a consequence, the learning capabilities of an individual can be divided into two classes, the implicit learning mechanisms of the architecture and explicit learning strategies in its memory. What is the nature of these explicit learning strategies? There are two possibilities. The first possibility is, that an explicit strategy can directly effect memory. In that case, we might have a rule in procedural memory that can directly change other rules by

adding conditions, changing weights, etc. This, however, doesn't seems to be a good option. One of the essential properties of procedural memory is, that it cannot access its contents directly. To be able to intentionally change a rule we need such a direct access. Moreover, it would violate one of the basic assumptions of an architecture of cognition, that the learning mechanisms in the architecture are sufficient. So we have to focus on a second, more likely possibility, that explicit learning is built on top of implicit learning in the sense that it learns by using implicit learning mechanisms.

A relatively simple example might explain this point. It is a well-known fact that people aren't very good at remembering facts that cannot easily be related to other knowledge they have. The whole tradition of theories about short-term memory that started with Miller's magical number seven is based on this fact (Miller, 1956). Atkinson and Shiffrin (1968) introduced the mechanism of rehearsal, to account for the fact that some facts in short-term memory do get stored in long-term memory, and others do not. Closer scrutiny however shows, that rehearsal isn't really a mechanism in the sense discussed earlier. Rehearsal isn't always at work, most of the time it isn't. People only rehearse if they have consciously decided to do so. So rehearsal isn't mechanistic, it is tied to intentions. Neither is rehearsal fixed. It turns out small children do not use rehearsal at all, so it must either be a dormant strategy that surfaces at some point, or a strategy that children acquire at some point during development. So rehearsal is a typical example of learning that is not part of the architecture, but rather a strategy represented in memory. We will return to this issue after a brief introduction of learning in the ACT-R architecture.

Learning in ACT-R

The ACT-R architecture has two memory systems, a declarative memory and a procedural memory. Associated to each of these memory systems is a number of learning mechanisms that add and maintain the knowledge in them. ACT-R is based on the theory of rational analysis (Anderson, 1990), and the learning mechanisms of ACT-R are no exception to this. According to rational analysis, the cognitive system is optimised to its environment. So a careful examination of the environment can shed as much light on the cognitive system as studying the cognitive system itself.

Implicit learning in declarative memory

A first example of the role of rational analysis in ACT-R is declarative memory. Elements in declarative memory, called chunks, closely resemble nodes in a semantic network. Each

chunk has an activation value. This value represents the odds that the chunk will be needed in the current context. To be able to estimate this value, the learning mechanisms have to keep a close track of the environment. The activation value of a chunk has two parts, the base-level activation and activation through associations with other chunks. The latter is context-dependent: once a certain chunk is part of ACT-R's current context, all chunks with an association to this chunk gain activation temporarily, since the presence of an associated chunk increases their chance of being needed. The base-level activation of a chunk is based on its past use. Two factors play a role: how many times a chunk was needed in the past, and how long ago this was. The learning rule used in ACT-R is derived from Bayesian statistics. Anderson shows that this rule both reflects regularities in the environment and empirical data from memory experiments. Association strengths are learned using similar rules. ACT-R uses activation values of chunks to order the chunks in the matching process, because the activation value determines the time it takes to retrieve the chunk.

Implicit learning in procedural memory

As the name implies, knowledge in production memory is represented by production rules. Associated with each rule are a number of parameters. The strength parameter is used to reflect past use of a rule, and is governed by the same laws as the base-level activation of chunks. The a, b, q and r parameters of a rule reflects its past cost-benefit characteristics. The a and b parameter represent the current and future cost of a rule, and the q and r parameters the chance of succeeding and reaching the goal. Bayesian statistics are again used to estimate these parameters. The cost-benefit parameters are used in conflict resolution. For each rule that is allowed to match, an expected outcome is calculated using the equation:

expected outcome = PG-C

In this equation, P is the estimated chance of success, calculated from q and r, G the estimated value of the goal and C the estimated cost of reaching the goal, calculated from a and b.

New rules are learned by the analogy mechanism. This involves generalisation of examples in declarative memory whenever a goal turns up that resembles the example. The examples are stored in specialised chunks, dependency chunks, that contain all the information needed: an example goal, an example solution, chunks (called constraints) that must be retrieved from declarative memory in order to create a solution, and sometimes additional subgoals that must be satisfied before the main goal can be reached.

```
dependency2+3
      isa dependency
      goal example-goal1
      modified example-solution1
      constraints fact2+3
example-goal1
      isa addition-problem
      arg1 two
      arg2 three
      answer nil
example-solution1
      isa addition-problem
      arg1 two
      arg2 three
      answer five
fact2+3
      isa addition-fact
      addend1 two
      addend2 three
      sum five
```

```
(p addition-problem-production1
=example-goal-variable>
      isa addition-problem
      arg1 =two-variable
      arg2 =three-variable
      answer nil
=fact2+3-variable>
      isa addition-fact
      addend1 =two-variable
      addend2 =three-variable
      sum =five-variable
==>
=example-goal-variable>
      answer =five-variable)
```

Figure 1. Example of ACT-R's analogy process. The left column shows the contents of declarative memory, the right column shows the analogised production rule.

Figure 1 shows and example of deriving a rule for doing a simple addition. The chunk dependency2+3 points to example-goal1 in which the addition 2+3 must still be calculated. In example-solution1 the answer is supplied in the answer slot. The additional fact needed to derive the rule is the fact that 2+3 equals 5. Whenever a new addition problem turns up, the rule production-problem-production1 will be derived and added to production memory. Note that all identifiers starting with an =-sign are variables, and can be matched with arbitrary chunks in declarative memory.

How can ACT-R's learning mechanisms, which are clearly implicit in nature, give rise to explicit, intentional learning? The next section will explore this question.

Explicit learning in ACT-R

Explicit learning in declarative memory

The implicit mechanisms to calculate the activation of a chunk in declarative memory provide a good estimate of the chance the chunk will be needed. But sometimes it is not enough to rely on the environment to cue the right facts the right number of times. As already observed in the introduction, rehearsal can provide a strategy that can help us memorise facts. Since base-level activation is calculated from the number of times a certain fact is retrieved, rehearsal can emulate extra uses of a certain fact. In other words, rehearsal tricks the base-level-learning mechanism into increasing the activation of a certain chunk.

Craik and Lockhart (1972) have observed that the effectiveness of rehearsal depends on

the level of processing. Simple maintenance rehearsal turns out to be much less effective than elaborate rehearsal, in which subjects try to associate the new fact with existing facts. This distinction can be easily explained in ACT-R, since elaborate rehearsal not only increases the base-level activation, but also increases associations with other chunks.

In a previous study I have showed how rehearsal can be modelled in ACT-R (Taatgen, 1996). Rehearsal is implemented by a set of production rules that create subgoals to do the rehearsal. The basic rehearsal productions can be enhanced by task-specific rules that can do elaborate rehearsal. The model was able to reproduce the data from classical free-recall experiments, and provide explanations for the primacy and recency effect. Rehearsal is clearly intentional in this model: it only occurs if ACT-R decides to push a rehearsal goal.

Explicit learning in procedural memory

The analogy mechanism creates rules from dependency chunks in declarative memory. The architecture, however, does not specify how dependencies are created. This is not necessary, since dependencies are just chunks in declarative memory. Dependencies may be picked up from the environment, or may be supplied by a parent or teacher. On the other hand, chunks may be created and manipulated by production rules. So an explicit theory of learning procedural rules involves dependency-manipulating rules in production memory. These rules create dependencies that are compiled into new rules by the analogy mechanism (figure 2). Again, strategies in procedural memory manipulate the implicit mechanisms: in stead of slowly gathering and generalising regularities from the environment, intentionally created rules are learned.

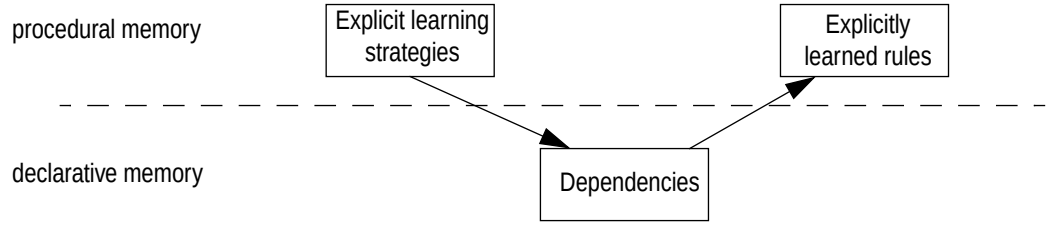


Figure 2. The process of explicit learning for procedural memory

The process described here is closely related to the concept of mental models (Johnson-Laird, 1983). A mental model is a mental construct that helps us predicting properties and events in the outside world. A dependency created by explicit learning strategies is a kind of mental model.

How to decide whether to do explicit learning?

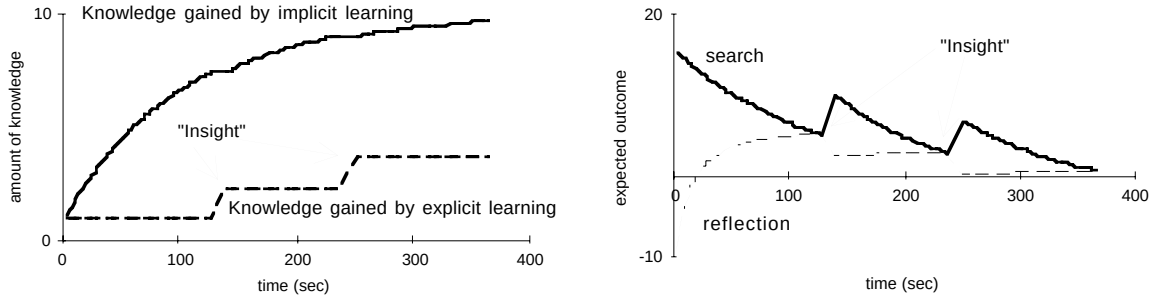
Since explicit learning is intentional, a decision has to be made at some point to start a learning attempt. We'll concentrate the discussion on explicit strategies for procedural knowledge, but similar points may be made on the decision to do rehearsal.

The easiest case that indicates an explicit strategy is needed, is when we get a certain outcome which turns out to be different from our expectations. An expectation-failure, in terms of Schank (1986). Often, however, indications aren't as clear. At some point we have to make a decision that our current approach isn't working, and something else is needed.

Decisions in ACT-R are based on a cost-benefit analysis of the available options. So suppose we present a new task to a subject. We have supplied some information to the subject about the task. The subject has several options. Should he just start an attempt in accomplishing the task using the information he has? Should he ask for advise, or go to the library to find more information? Or should he first reflect on the task, to come up with some new strategies? Suppose we restrict our analysis to two possible strategies: search and reflection. If the subject chooses search, he attempts to accomplish the task given his current knowledge about the task. If he chooses reflection on the other hand, he will try explicit learning strategies to come up with new rules.

In Taatgen (1997) I describe a model that calculates the expected outcome of either strategy over time. I will briefly summarise the results here. The expected outcome has three components, as mentioned before, P, the chance of reaching the goal using the strategy, G, the estimated value of the goal, and C, the estimated cost of a strategy. Since G is not related to a strategy, we will concentrate on P and C. For search, the cost is relatively constant, and typically low. The chance of success of search, however, decreases over time if the goal is not reached, reflecting the fact that repeated failure is a strong indication that the current knowledge is false or insufficient. An assumption of reflection is, that it needs existing knowledge to work with. You can't get something out of nothing. This knowledge can have several sources, as we will see later on. For now, we will assume that the only source of knowledge is implicit knowledge gained through search. So if we have little implicit knowledge to work with, the cost of reflection is high, since it is difficult to come up with something useful. As our explicit knowledge increases, the cost to come up with something new increases as well.

A final assumption of the model is, that search increases implicit knowledge, reflection increases explicit knowledge, and explicit knowledge is more powerful. This enables us to describe the increase of knowledge over time, related to the conflict resolution mechanism that



(a) (b)

Figure 3. Results of the model. (a) shows growth of knowledge over time, and (b) shows the conflict resolution process.

selects the strategy with the highest expected outcome. Figure 3 shows an example of the results of the model. Figure 3a shows the growth of knowledge about a specific problem, and figure 3b shows the expected outcome of search and reflection. The discontinuities in both graphs indicate a change of strategy.

A nice aspect of this model is, that it can give a rational account of the explore-impasse-insight-execute stages often identified in problem solving that requires insight. In the explore stage, the subject still thinks his existing knowledge is the best way to reach the goal, in the impasse stage he decides that the existing knowledge is insufficient, and in the insight stage reflection starts in an attempt to gain new explicit knowledge. Finally, in the execute stage the search process continues using the newly gained knowledge.

An ACT-R model of a simple explicit strategy

The model discussed above is just a mathematical model, and only represents the amounts of several types of knowledge with their cost-benefit characteristics. To see if the approach discussed here really works, we have to make a detailed ACT-R model in which actual learning occurs. The main parts of interest in such a model are the dependency-creating rules, since these rules form the explicit learning part of the model. These rules have to be quite general, since they must be applicable to a wide range of problems. So general rules are in principle context-independent. To be able to work though, they operate on context-dependent knowledge in declarative memory. Possible sources of knowledge are:

- Task instructions and examples
- Relevant facts and biases in declarative memory
- Feedback

- Old goals and old dependencies for the same problem

The beam task

The task we will start with is a beam task. The assumption is, that the model initially has no task-specific rules about beam-problems. So the only procedural knowledge the model has is a number of general rules. Later on, we will use the same general rules for other tasks.The problem is relatively easy: a beam is given, with weights on the left and the right arm. Attached to the arms of the beam are labels, each with a number on it. The task is to predict whether the beam will go left, right, or remains in balance. The number on the labels have no influence on the outcome. Figure 4 shows an example of a beam. Although the task is easy if we know something about weights and beams, it is much more difficult if we know nothing at all.

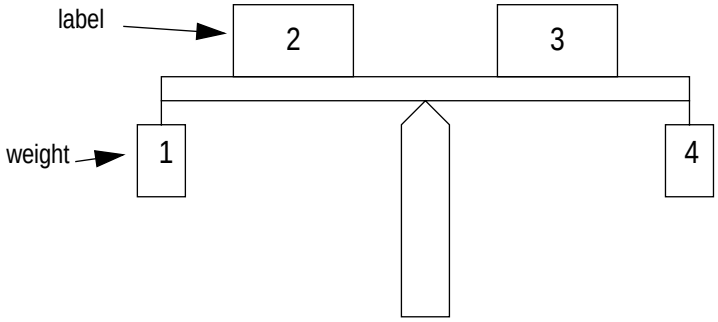


Figure 4. Example of the beam task

The general rules used to learn this task are the following:

Property-retrieval

If there is a task that has a number of objects, create a dependency that contains an example of retrieving a certain property of each of the objects. In the case of the beam task, the objects are the arms of the beam, and weight and label are possible properties. So this rule creates a rule that directs attention to a certain aspect of the task.

Find-fact-on-feedback

If feedback indicates that the answer is incorrect, and also contains the correct answer, set up a dependency that uses the goal and the answer as examples. Also, retrieve some fact that serves as a constraint in the dependency. To be able to generate correct rules for the beam task, we need to retrieve the fact that a certain number is greater than another number, in order to predict correctly whether the beam will go left or right.

Both general rules involve retrieving an arbitrary chunk from declarative memory, either a property or a fact. Normally the retrieval of arbitrary chunks will not produce the right rules.

The chunks retrieved are however not arbitrary, since ACT-R's activation mechanism ensures the chunk with the highest activation is retrieved. Since activation represents the odds that a chunks is needed, the chunk with the highest odds of being needed is retrieved. This activation can itself again be manipulated by explicit declarative memory strategies like rehearsal.

In the model this is reflected by the fact that both property-retrieval and find-fact-on-feedback can be influenced by prior knowledge. If there is an association strength between beam and weight, indicating knowledge that a beam has something to do with weight, property-retrieval will choose weight in favour of label. If there is an association strength between beam and greater-than, a greater-than fact will be retrieved by find-fact-on-feedback.

Since the general rules are just production rules, they can be in direct competition with the task-specific rules they generate. So if property-retrieval generates a rule X to retrieve the label, X will compete with property-retrieval. So if X is doing a bad job, which it will if it retrieves the label that has no relevance at all to solving the problem, its evaluation will decrease, and it will eventually lose the competition, in which case property-retrieval will create an example of retrieving weight. Although find-fact-on-feedback is only activated if feedback indicates an incorrect answer, so when an expectation-failure occurs, the rules it produces are in competition with each other. The rule with the highest success rate will eventually win.

Figure 5 summarises the property-retrieval rules, and figure 6 summarises the find-fact-on-feedback rules. Figure 6 shows the case in which a "Don't know" rule fires. If in stead an incorrect answer is predicted, a dependency is created in the same manner. Apart from the gen-

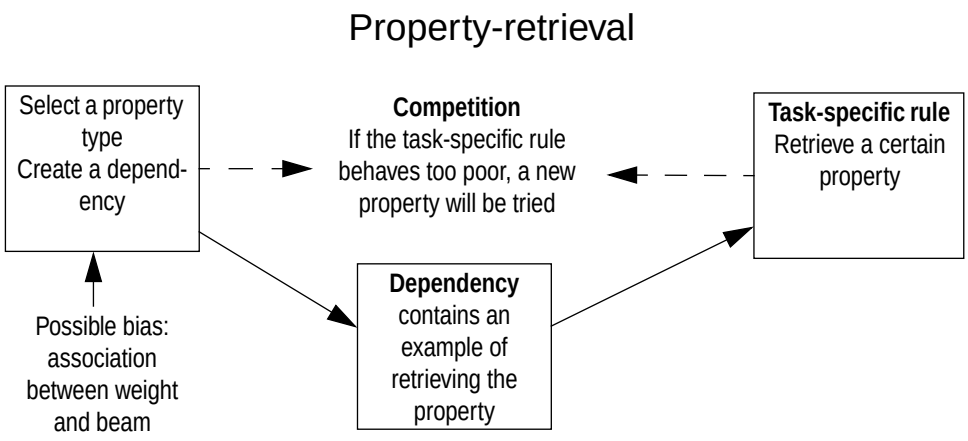


Figure 5. How property-retrieval works

eral rules, the model contains lisp functions to generate random beams, and production rules to give feedback. When the model produces an incorrect answer, it will try the same beam again

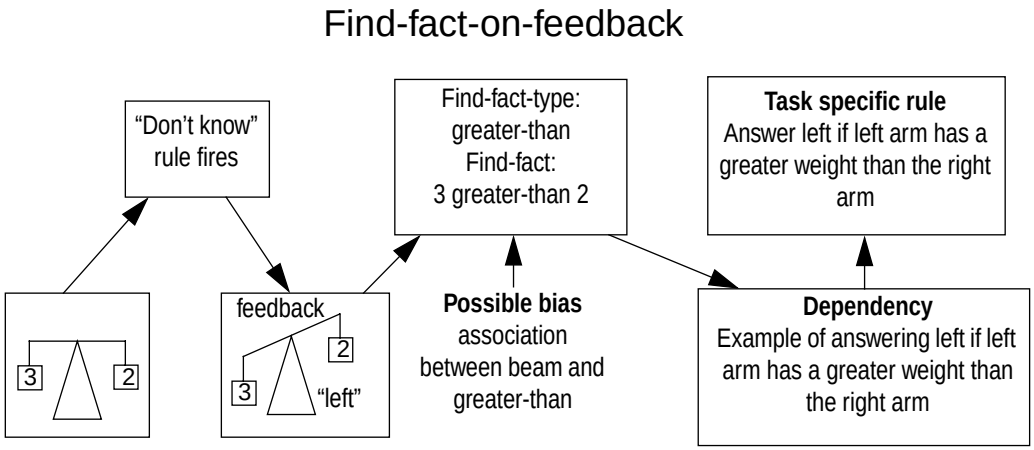


Figure 6. How find-fact-on-feedback works

until it can predict the right outcome.

Simulation results

The general rules turn out to be sufficient to learn the task. The following rules are examples of (correct) rules learned by the model. The rule generated by property-retrieval is a rule that retrieves the weight property for both arms of the beam, and stores them in the goal:

```
(P GEN-GOAL-PRODUCTION10
     =OLDGOAL10-VARIABLE>
        ISA GEN-GOAL
        TYPE SOLVE-BEAM
        OB1 =O6-VARIABLE
        OB2 =O7-VARIABLE
        PROP1 NONE
        PROP2 NONE
     =P7-VARIABLE>
        ISA PROPERTY
        OF =O6-VARIABLE
        TYPE WEIGHT
        VALUE =ONE-VARIABLE
     =P8-VARIABLE>
        ISA PROPERTY
        OF =O7-VARIABLE
        TYPE WEIGHT
        VALUE =SIX-VARIABLE
  ==>
     =OLDGOAL10-VARIABLE>
        PROP1 =ONE-VARIABLE
        PROP2 =SIX-VARIABLE
        PROPTYPE WEIGHT)
```

Again note that all identifiers starting with an =-sign are variables. One of the rules generated by find-fact-on-feedback is a rule that predicts when the right arm of the beam will go down.

```
(P GEN-GOAL-PRODUCTION12
    =OGOAL11-VARIABLE>
        ISA GEN-GOAL
        TYPE SOLVE-BEAM
        OB1 =O6-VARIABLE
        OB2 =O7-VARIABLE
```

```
      PROP1 =ONE-VARIABLE
      PROP2 =SIX-VARIABLE
      ANSWER NONE
      PROPTYPE WEIGHT
   =F61-VARIABLE>
      ISA GEN-FACT
      TYPE GT-FACT
      SLOT1 =SIX-VARIABLE
      SLOT2 =ONE-VARIABLE
==>
   =OGOAL11-VARIABLE>
      ANSWER RIGHT)
```

The model was tested in several conditions, differing in the bias given for the properties and the fact-type. The following table summarises the conditions:

P+ Association between beam and weight

P- Association between beam and label, so a bias for the wrong property

F+ Association between beam and greater-than

F- Association between both beam and greater-than, and beam and number, so two possible fact-types were favoured.

F-- No associations between beam and fact-types, four fact-types are possible.

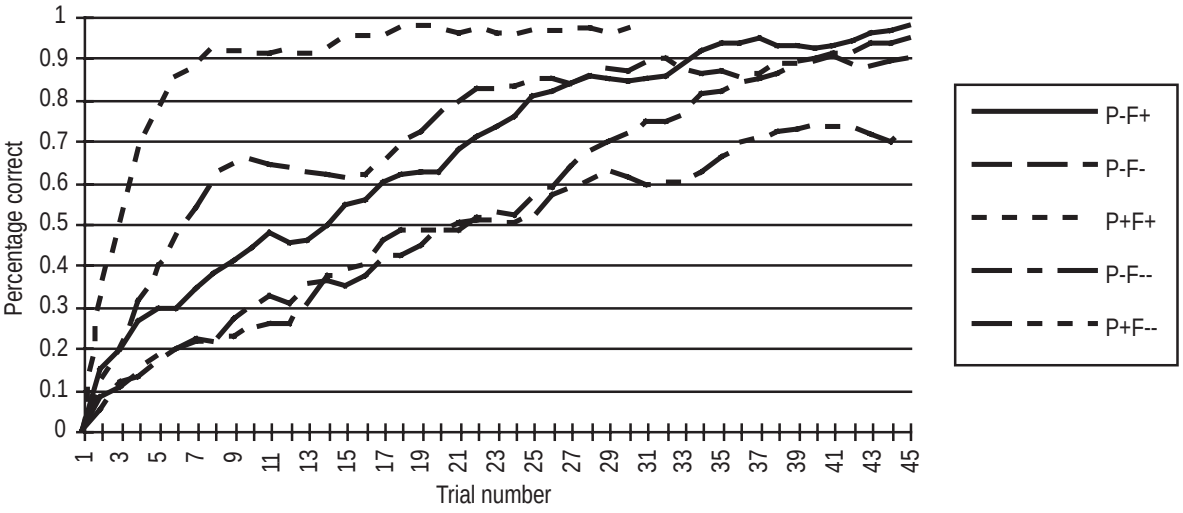


Figure 7. Results of the beam model

Each experiment has a P- and an F-condition. Each experiment has been run 30 times for 50 trials. Figure 7 shows the results. As can be seen in the graph, in the P+F+ condition ACT-R learns to solve the task quite rapidly, and the fact that the model doesn't reach a 100% score within a few trials is only due to the fact that beams are generated randomly, only occasionally producing a beam in which balance is the correct answer. Performance decreases if the model has less initial information. In the case of the P-F-- condition, the model often fails to find the correct rules for the task.

The results in the figure above suggest a gradual increase of performance. This is however not the case, but a result of averaging 30 runs. If individual runs are examined, each has a certain point where performance increases dramatically. To make this clear the following graph depicts the average number of incorrect tries for each trial in the P-F+ condition, averaged with respect to the point where the model switches from examining the label property to examining the weight property. So at x=0 the model creates a dependency that contains an example of retrieving weight.

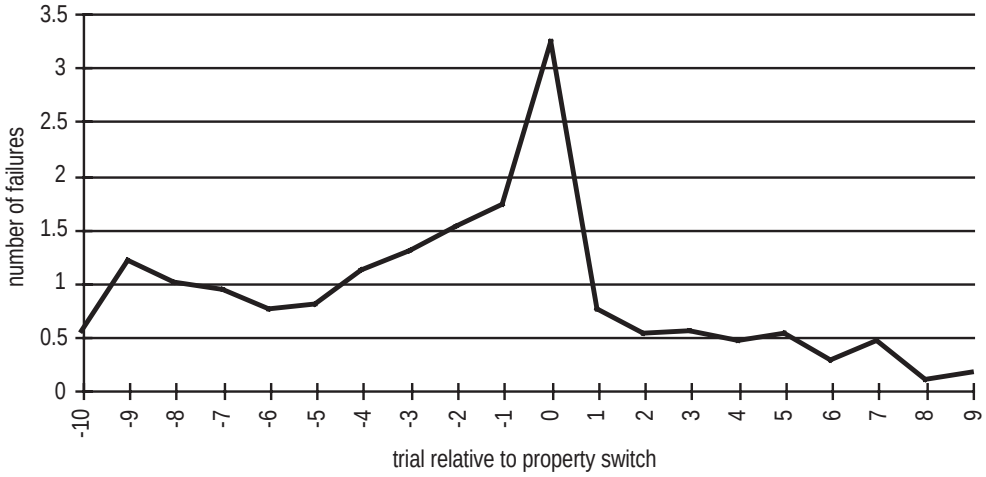


Figure 8. Average number of failures for trials relative to a property switch

The dependency is created at the moment that the model has failed several times to predict the right answer. As a result, the evaluation of the rule that retrieves the labels drops and the general rule can propose a new dependency. In a sense, this process resembles the impasse-insight state of insight problem solving.

The card task

General rules are of course only general if they can be used for several different tasks. So the same rules were used for a new task, a card-classification task. In this task, cards with pictures must be sorted into two categories. The pictures are either one or two squares or circles, which are either red or green and either large or small. The criterion to sort on is the colour (red=yes; green=no), which the subject has to discover. The same general rules can be used to learn this task. First, a property must be selected, so either colour, shape, size or number. After that, the relevant aspect is tied to the answer. The following rules are examples of rules learned by the model:

```
(P GEN-GOAL-PRODUCTION167
     =OLDGOAL1167-VARIABLE>
        ISA GEN-GOAL
```

```
       TYPE SOLVE-CAT
       OB1 =O164-VARIABLE
       PROP1 NONE
    =P165-VARIABLE>
       ISA PROPERTY
       OF =O164-VARIABLE
       TYPE COLOUR
       VALUE =GREEN-VARIABLE
 ==>
    =OLDGOAL1167-VARIABLE>
       PROP1 =GREEN-VARIABLE
       PROPTYPE COLOUR)

(P GEN-GOAL-PRODUCTION169
    =OGOAL167-VARIABLE>
       ISA GEN-GOAL
       TYPE SOLVE-CAT
       OB1 =O164-VARIABLE
       PROP1 =GREEN-VARIABLE
       ANSWER NONE
       PROPTYPE COLOUR
    =GREEN-VARIABLE>
       ISA GEN-FACT
       TYPE COLOUR
       SLOT1 GREEN
 ==>
    =OGOAL167-VARIABLE>
       ANSWER NO)
```

Discrimination-shift learning

On of the advantages of explicit learning strategies compared to implicit learning is, that they can handle change more easily. If something in the environment is no longer valid, an explicit strategy may react by proposing new knowledge to replace the old. An example of a task in which the rules change is discrimination-shift learning (Kendler & Kendler, 1959). Figure 9 shows an example of this task. Subjects have to learn to discriminate the four stimuli in two re-

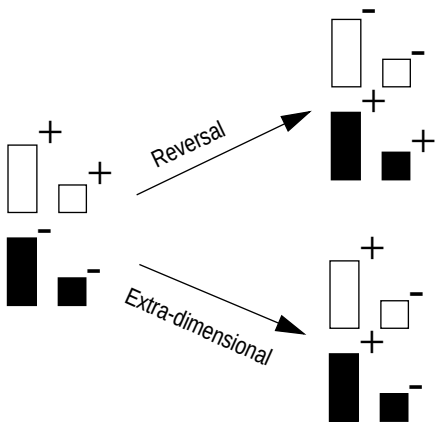


Figure 9. Example of a discrimination-shift task

inforcement categories, for example white is positive and black is negative. In this sense it closely resembles the card task discussed previously. After the subject has made 10 consecutive correct predictions, the reinforcement scheme is changed: either a reversal-shift, in which all

stimuli that received previous positive reinforcement get negative reinforcement and vice-versa, or an extra-dimensional shift, in which the dimension is changed on which the reinforcement is given, in the example from white to large. It turns out that adults and older children are faster at learning the reversal-shift condition, while young children and animals are faster at the extra-dimensional shift. Figure 10a shows the results of an experiment by Kendler and Kendler (1959). The ACT-R model of adult behaviour uses the same 8 production rules as used in the beam-task, implementing the property-retrieval and find-fact-on-feedback strategies. It learns rules that are quite similar to the rules for the card task. The small-child/animal model uses only 2 of the 8 production rules, implementing only limited find-fact-on-feedback strategy. So the latter model hardly uses any explicit reasoning at all, but rather stores regularities in the environment in dependency chunks. The results of these models are shown in figure 10b. Although the models do not mimic the empirical data precisely, the general effects are in the same direction.

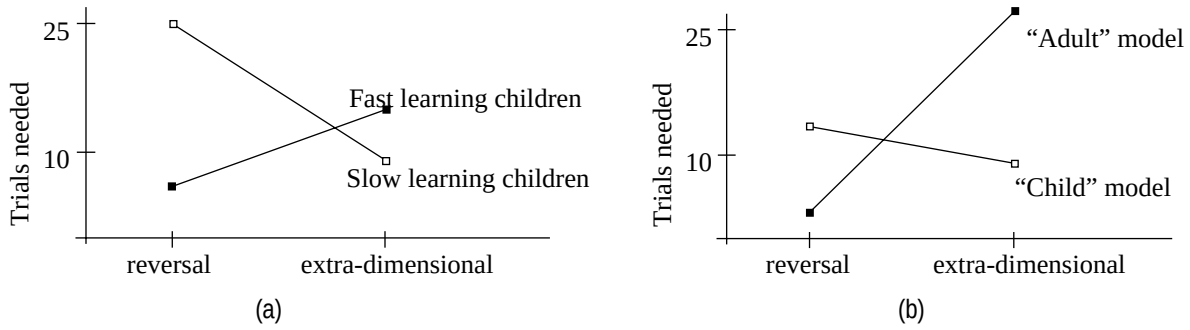


Figure 10. Trials needed to learn the discrimination-shift task, (a) from the Kendler & Kendler experiment, (b) by the ACT-R model

Despite the fact that the discrimination-shift task is generally not considered to be an insight problem, it nevertheless requires the subject to notice that something has changed, and to discover the new relations. So it can be seen, in a sense, as an elementary insight problem.

Conclusions

The goal of cognitive modelling is to create computer simulations of cognitive processes. A criterion for a good model is of course whether the results of the simulation match the empirical data. A second criterion, that becomes increasingly more important, is the question whether the model can learn the knowledge it needs. A model that uses a large set of specialised production rules is less convincing than a model that gathers its own knowledge. The learning mechanisms that are part of the architecture are often not capable of doing this job by themselves, so

they need augmentation. In this chapter, it is argued that the mechanisms of the architecture correspond to implicit learning. These mechanisms can be augmented by explicit learning, that is implemented by knowledge in memory that directs the implicit learning mechanisms. The following table summarises the memory systems and learning mechanisms and strategies in ACT-R.

	Implicit mechanisms	Explicit strategy
Declarative memory	Base-level learning	Maintenance rehearsal
	Association learning	Elaborate rehearsal
Procedural memory	Parameter learning	?
	Analogy	Dependency-manipulating rules

Although implicit mechanisms are fixed, explicit strategies have to be learned. So individuals probably differ in their explicit strategies, although they may well have lots of them in common. Rehearsal, for example, is a strategy used by almost all adults, though it is clearly not something we were born with. An interesting question is, whether the same property is also true for the dependency-manipulating rules. Is there a sequence of rules that unfolds during development? The model of discrimination-shift task at least hints in this direction. On the other hand we may well expect large individual differences. Experiments in which subjects have to solve difficult problems often show that every subject solves a problem in a different way.

An interesting question is how the issues discussed here can be related to other architectures. The emphasis on learning models is often attributed to the ascent of neural network models. A neural network model typically starts out with an untrained network, that gains knowledge by experience. Neural networks are powerful in the sense that a three-layer network can learn any function if properly configured. This power is also a weakness, especially is you take the time to learn something into account. Neural networks usually do not have any goal-structures, so lack the mechanisms to be able to focus learning. To summarise: neural networks do a very good job at implicit learning, but the step towards explicit learning is difficult to make because of the absence of any goals or intentional structures.

In the Soar architecture on the other hand, goals and deliberate reasoning are even more important than in ACT-R (Newell, 1990; see for an extensive comparison of ACT-R and Soar Johnson, 1997). The ACT-R models presented in this chapter only deliberate if existing simple rules prove to be insufficient and, more importantly, if there is any knowledge present on how

to deliberate. So if ACT-R has to choose between actions A and B, a cost benefit analysis between the rule “do A” and the rule “do B” will decide. Only if both rules prove to perform badly, explicit learning strategies will try to find a more sophisticated rule. A Soar model on the other hand will always try to make a deliberate and rational choice between A and B, a process that may require a lot of processing and specific task knowledge. A Soar model that has to choose between A and B, and has no particular additional knowledge, will get into a infinite sequence of impasses. Soar’s single learning mechanism is chunking, which summarises the processing done between an impasse and its resolution into a new production rule. Although chunking is a mechanism, it is only activated after an impasse has been resolved, so after a deliberate problem solving attempt. Since chunking is Soars only learning mechanism, this may cause trouble. For example, to learn simple facts, for which ACT-R has a separate declarative memory, Soar needs the elaborate scheme of data-chunking. Data-chunking eventually produces rules like “IF bird THEN note is has wings”. To be able to learn this, however, a lot of deliberating has to be done by production rules *that or not part of the architecture*. So in a sense Soar walks the reverse way: in stead of building explicit learning on top of implicit learning, it accomplishes typical implicit learning tasks by elaborate explicit schemes. The critical reader will be able to find more examples of Soars problems with simple satisficing behaviour in Johnson (1997).

Since many other architectures like EPIC (Kieras & Meyer, in press) currently support no learning at all, ACT-R presently seems to be the best platform to support explicit learning strategies on a basis of implicit learning. To be able to fully sustain explicit learning though, some technical issues in ACT-R must be resolved. Most notably a mechanism must be included to create new chunk-types. The models discussed in this chapter circumvented this problem by using a generic goal type (GEN-GOAL) for all goals.

This chapter may be a starting point for several strands of further research. A more thorough inventory of possible general rules has to be made. This leads to a further question: where do the general rules themselves originate? This question is best studied in a developmental setting. Is it possible to specify a sequence of general rules that are learned during development, that can account for the fact that older children can handle more abstract concepts? A good starting point would be to make a model of the complete balanced-beam task (Siegler, 1976) that can explain the stage-wise learning aspects of this task.

Norman (1993) distinguishes experiential and reflective cognition in a human-computer interaction setting. Experiential cognition corresponds to the search-process discussed in this chapter, and reflective cognition to reflection and explicit strategies. According to Norman, de-

signers have to ask themselves the question whether their design is supposed to support experiential or reflective processes, and create it accordingly. Current methods for task analysis used in human-computer interaction typically do not support any learning. In this sense our research apparatus for human-computer interaction can only model experiential cognition. To be able to model the user as an active learner, however, a modelling environment is needed that supports explicit leaning. The Soar architecture provides a deliberate learning environment, and is used as such already. The ACT-R architecture may well be a good alternative, since it provides a more flexible model of explicit learning that can account for individual differences, and can also model implicit learning phenomena.

References

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

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

Atkinson, R.C. & Shiffrin, R.M. (1968). Human memory: A proposed system and its control processes. In K.W. Spence & J.T. Spence (Eds.), *The psychology of learning and motivation* New York: Academic Press.

Craik, F.I.M. & Lockhart, R.S. (1972). Levels of processing: a framework for memory research. *Journal of verbal learning and verbal behavior, 11,* 671-684.

Knoblich, G. & Ohlsson, S. (1997). Can ACT-R have insights? Chapter 4.2 of this book.

Johnson, T. (1997). A comparison between ACT-R and Soar. Chapter 2.1 of this book.

Johnson-Laird, P. (1983). *Mental models*. Cambridge, MA: Harvard university press.

Kendler, T.S. & Kendler, H.H. (1959). Reversal and nonreversal shifts in kindergarten children. *Journal of experimental psychology, 58,* 56-60.

Kieras, D. & Meyer, D.E. (in press). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-computer interaction*.

Miller, G.A. (1956). The magical number seven plus or minus two: Some limits on our capacity for processing information. *Psychological review, 63,* 81-97.

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

Newell, A. & Rosenbloom, P.S. (1981). Mechanisms of skill acquisition and the law of practice. In J.R. Anderson (Eds.), *Cognitive skills and their acquisition* Hillsdale, NJ: Prentice-Hall.

Norman, D.A. (1993). *Things that make us smart*. Reading, MA: Addison-Wesley.

Rumelhart, D.E. & McClelland, J.L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition.* Cambridge, MA: MIT Press.

Schank, R.C. (1986). *Explanation patterns: understanding mechanically and creatively*. Hillsdale, NJ: Erlbaum.

Shanks, D.R. & John, M.F.S. (1994). Characteristics of dissociable human learning systems. *Behavioral and Brain Sciences, 17*(3), 367-447.

Siegler, R.S. (1976). Three aspects of cognitive development. *Cognitive Psychology, 8*, 481-520.

Taatgen, N.A. (1996). A model of free-recall using the ACT-R architecture and the phonological loop. In H.J.v.d. Herik & T. Weijters (Eds.), *Proceedings of Benelearn-96*, (pp. 169-178). Maastricht, the Netherlands: Universiteit Maastricht, MATRIKS.

Taatgen, N.A. (1997). A rational analysis of alternating search and reflection in problem solving. In *Proceedings of the 19th annual conference of the cognitive science society*. Hillsdale, NJ: Erlbaum.