CHAPTER 5

Strategies of learning
5: Strategies of learning

5.1 Introduction

In the previous chapter we saw that learning is a concept with two layers. The bottom layer consists of the learning mechanisms of the architecture, while the upper layer is a set of learning strategies that manipulate the mechanisms of the bottom layer. We have already seen an example of a learning strategy in the form of rehearsal. In this chapter, the focus will be on learning strategies that try to infer new knowledge, a phenomenon that we have witnessed in the protocols of the participants in the scheduling problem. There are several questions to be answered with respect to learning strategies.

The first question is: when are learning strategies used. A learning strategy is tied to an explicit learning goal. This means that at some point during reasoning, a learning goal must be posed in favor of other processing. The protocols in chapter 3 demonstrate that several episodes can be distinguished in the problem solving process, some of which involve search, and some of which involve reflection. In the reflection episodes, participants discover new strategies, and the recurrence of these strategies in later episodes indicates that they have been learned during the first episode. But when, and for what reasons, does a participant decide to stop search and start reflection? This is a question of meta-cognition, often portrayed as a monitoring process that prevents unbounded search. An alternative, which I will pursue in sections 5.2 and 5.3, is to incorporate the function of meta-cognition without the need for a separate monitoring process. A separate process would require its own monitor, leading to endless regress.

A second question one might ask is how learning strategies themselves are learned, and what their nature is. Learning learning strategies is probably a long-term process, so it will be hard to investigate this process in a standard experimental setting. A better setting to investigate the nature of learning strategies is development. During development, a lot of learning strategies are acquired. Probably many differences between adults and children with respect to their reasoning capabilities can be explained in terms of what type of information they can represent, and what learning strategies they have available to learn this information. In section 5.4, three theories of development will be discussed, and what can be learned from them.

The third and final question is how to model strategy learning in ACT-R. New production rules have to be represented in memory. Some learning scheme has to be developed that is independent of the current task. In sections 5.5 and 5.6, I will propose some example learning strategies, and show how they can learn task-specific knowledge in two different domains. To emulate some of the developmental aspects of these strategies, I will do some “reverse development” by impoverishing the learning strategies. As we will see, this leads to behavior associated with an earlier stage of development.
5.2 Search vs. Insight

In chapter 1, I criticized the traditional approach of problem solving, in which solving a problem means no more and no less than finding an appropriate sequence of operators that transforms a certain initial state into a state that satisfies some goal criterion. The difficulty of problem solving is determined by factors as the length of the sequence needed, the number of possible operators, and the amount of knowledge available on how to choose the right operator.

The alternative insight theory stresses the moment at which the crucial step towards the solution is found. Insight can be viewed in two ways: as a special process, or as a result of ordinary perception, recognition and learning processes (Davidson, 1995). Despite the intuitive appeal of a special process, the latter view is more consistent with the modern information-processing paradigm of cognitive psychology, and is much more open to both empirical study and computational modeling. One way to look at insights from an information-processing viewpoint is that an insight involves the relaxation of constraints (see, for example, Knoblich & Ohlson, 1996). In the nine-dots problem mentioned in chapter 1, for example, the initial assumption that all lines should remain within the 3x3 square is a constraint that needs to be relaxed.

Another famous insight problem is the box-candle problem, in which a candle has to be affixed to a door, using a box of candles, a box of matches, and a box of tacks (see, for example, Mayer, 1983). The crucial constraint to be relaxed is the fact that the boxes are not just containers, but can also be used to support the candle. Knoblich & Ohlsson (1996) have shown in an experiment involving matchstick problems that once a constraint is relaxed, it stays relaxed.

Looking at insights as removing constraints is a rather negative approach: something that is there needs to be removed. A slightly different view on insight is to assume some new knowledge is gained at the moment of insight. This corresponds well with the idea that a constraint stays relaxed. Another advantage of this view on insight is that not all insights can be described as relaxing constraints. The fact that participants in the scheduling problem start using complex inferences during a reflection episode can of course be called “the relaxation of the constraint not to use complex inferences”, but this stretches the original idea so much it becomes almost meaningless: it is like defining the creation of a statue as removing marble.

Both the search and the insight theory select the problems to be studied in accordance with their own view. Typical “search”-problems involve finding long strings of clearly defined operators, as in the eight puzzle, the towers-of-hanoi task and other puzzles, often adapted from artificial intelligence toy domains. “Insight”-problems, on the other hand, can be solved in only a few steps, often only one. Possible operations are often defined unclearly, or misleadingly, or are not defined
at all, as the nine-dots and candle problems illustrate. Due to this choice of problems, both evidence from insight and search experiments tend to support their respective theories. Both theories ignore some aspects of problem solving. The search theory seems to assume that participants create clear-cut operators based on instructions alone, and fails to assign a significant role to reflection. Insight theory on the other hand offers no explanation of the role of processing that happens before the ‘‘insight’’ occurs. An obvious alternative is to think of both search and insight as aspects of problem solving, and to try to find a theory of problem solving that combines the two (Ohlsson, 1984).

One such view sees insight as representational change, which is a more general term that includes constraint relaxation and gaining new knowledge about the task. Search is needed to explore the current representation of the problem, and insight is needed if the current representation appears not to be sufficient to solve the problem. In this view, search and insight correspond to what Norman (1993) calls experiential and reflective cognition. If someone is in experiential mode, behavior is largely determined by the task at hand and the task-specific knowledge the person already has. In reflective mode on the other hand, comparisons between problems are made, possibly relevant knowledge is retrieved from memory, and new hypotheses are created. If reflection is successful, new task-specific knowledge is gained, which may be more general and on a higher level than the existing knowledge. All these theories, however, fail to specify at what time a certain mode of thinking will be used, and due to what influences the mode of thinking changes.

In the protocol analysis of the scheduling problem in chapter 3, we saw that all participants start with an experiential search strategy, and only later on switch to a reflective strategy. As we have observed, the process reflects the explore-impasse-insight-execute pattern described in the literature about insight (Ohlsson, 1984; Davidson, 1995). Some, but not all, of the participants show some sort of impasse, during which they stop searching, just stare at the screen for a minute, and then try a new approach. Furthermore, there is no difference between the explore and the execute stage: the participant just searches on, using the knowledge gained by reflection. Sometimes further reflection is needed to reach a solution.

5.3 A dynamic growth model

In this section a model is proposed that explores the distinction between search and reflection. The model is based on Anderson’s theory of rational analysis, the theoretical basis of ACT-R (Anderson, 1990). According to rational analysis, participants choose strategies based on a cost-benefit analysis: the strategy that has the lowest expected cost and the highest probability of success is selected in favor of others. The model is not an actual ACT-R model, but a dynamic growth model, in
which the trade-off between search and reflection is modeled in a coarse-grained way. Dynamic models are used in developmental psychology to describe developmental paths, for instance a model that describes stage-wise increases in knowledge (Van Geert, 1994; 1998). In section 5.6, the coarse-grained model will be applied in actual ACT-R models.

In order to give a rational account of insight learning, the first question is: why would participants initially prefer a search strategy in the scheduling problem? The reflective strategy seems to be much more powerful. There are several reasons for this. A first reason is that reflective reasoning has a high cost. To be successful, several aspects of the task must be combined and kept in memory. Additional knowledge must be retrieved from memory and it may be necessary to seek analogies with other problems. A second reason is that it is not immediately evident that search will be unsuccessful. In the nine-dots problem, but also in the scheduling problem, naive search alone does not work, but people generally do not know this when they start on these problems. Why not try the strategy which takes the least effort first? A third reason is that as a participant starts with a new type of problem, he has only read instructions and has seen an example problem. He first has to learn the basic rules and operators by experience, before he can attempt any higher level strategies.

Considerations like these are the basic ingredients for the model. In the model, search and reflection are two competing strategies, whose evaluations depend on expected gain. Estimates on these gains change in time, due to increasing knowledge and the successes and failures due to this knowledge.

The model
According to rational analysis (Anderson, 1990), strategies are chosen with respect to their expected outcome, according to the following equation:

$$\text{Expected outcome of strategy } s = P_s G - C_s \quad (5.1)$$

In this equation, $P_s$ is the estimated probability of reaching the goal using strategy $s$, $G$ is the expected value of the goal, and $C_s$ is the estimated cost of reaching the goal using strategy $s$.

The model will attempt to describe how search and reflection will alternate while solving a problem. The model is coarse-grained in the sense that the knowledge of the system with respect to a certain task is summarized in two variables $L_1$ and $L_2$. $L_1$ is a measure for the amount of basic task-knowledge, for example, in the case of the scheduling task, knowledge about adding a task to an existing plan and knowledge to judge whether a solution is correct. $L_2$ corresponds to the amount of higher-level knowledge in the system, for example the fact that it is a good idea to see how the tasks add up to the amount of time the workers have available. If a
participant starts with a new problem, we assume that both variables have a small value. Later on, they increase, since the participant builds up knowledge during problem solving. The assumption of the model will be that search will increase the amount of basic knowledge, represented by $L_1$, and reflection will increase the amount of higher-level knowledge, represented by $L_2$. The choice of two knowledge levels is somewhat arbitrary, as are some of the choices of parameters in the equations below. The reader should keep in mind that the goal is to produce a rational account of the alternation between search and reflection.

The following equations show how $L_1$ and $L_2$ grow in time, and are inspired by the growth equation used by Van Geert (1994):

If the strategy in step $i-1$ is search, then

$$L_1(i) = L_1(i-1) + R_1 L_1(i-1) \left( 1 - \frac{L_1(i-1)}{L_1^{\text{max}}} \right)$$ \hspace{1cm} (5.2)

else $L_1$ keeps its value, so $L_1(i) = L_1(i-1)$. $R_1$ is a constant that controls the rate of growth, and $L_1^{\text{max}}$ is the maximum possible value for $L_1$. The fraction at the end of the equation ensures that $L_1$ doesn’t exceed its maximum value. Assuming only search is used, the value of $L_1$ grows gradually and levels off once it approaches the maximum. Figure 5.1 shows an example of the growth of $L_1$ knowledge if only search is used, and $L_1^{\text{max}}$ equals 10.

The equation for $L_2$ is slightly more complicated, because the increase in value depends on the current value of $L_1$, reflecting the fact that we can only gain higher-level knowledge if we have enough basic knowledge.
A dynamic growth model

If the strategy at step \( i-1 \) is reflection, then

\[
L_2(i) = L_2(i-1) + S_{12} \cdot L_1(i-1) \left( 1 - \frac{L_2(i-1)}{L_{2\text{max}}} \right)
\] (5.3)

else \( L_2(i) = L_2(i-1) \). \( L_{2\text{max}} \) is the maximum possible value for \( L_2 \). The parameter \( S_{12} \) (support) controls the influence of basic knowledge on the increase of higher level knowledge.

Now that we have described how knowledge grows depending on the type of strategy, we have to describe the process by which a strategy is chosen. At this point, Anderson’s expected gain equations are introduced into the model. Whether the strategy at step \( i \) will be search or reflection is determined by their respective expected outcomes:

\[
\text{Expected outcome of search} = P_{\text{search}}(i) \cdot G - C_{\text{search}}
\] (5.4)

\[
\text{Expected outcome of reflection} = P_{\text{ref}} \cdot G - C_{\text{ref}}(i)
\] (5.5)

The strategy with the highest expected outcome will be chosen. In these equations \( G, C_{\text{search}}, \) and \( P_{\text{ref}} \) are fixed parameters. \( G \), the expected value of the goal, is assumed to be fixed as long as the goal is not reached. \( C_{\text{search}} \), the cost of search, may change in actual problem-solving situations, for example due to the fact that search becomes more complicated once more knowledge is involved. But since these fluctuations are task-dependent, the current model assumes that the costs of search remain constant. The influence of \( P_{\text{ref}} \), the chance of success of reflection, will be taken into account in the specification of the costs of reflection. \( P_{\text{search}}(i) \) and \( C_{\text{ref}}(i) \) are variable in time, and rise and fall due to the chosen strategy and the growth in knowledge.

The probability that search will reach the goal depends on the amount of knowledge and the current evaluation of this knowledge:

\[
P_{\text{search}}(i) = \frac{L_1(i)P_1(i) + wL_2(i)P_2(i)}{L_1(i) + wL_2(i)}
\] (5.6)

The constant \( w \) determines how much more useful higher-order knowledge is than basic knowledge. \( P_1(i) \) is the contribution to the probability of success of \( L_1 \) knowledge, and \( P_2(i) \) the contribution of \( L_2 \) knowledge. The probability of success increases as knowledge increases, but decreases over time if the goal is not reached. The decrease in knowledge is calculated by multiplying the probability of success by a decay parameter on each time-step search is used as strategy. New knowledge is given the benefit of the doubt, and is assigned an initial probability of success of 1. Both \( P_1(i) \) and \( P_2(i) \) can be calculated using:
5: Strategies of learning

\[ P_j(i) = \frac{p_{\text{decay}} P_j(i-1) \cdot L_j(i-1) + (L_j(i) - L_j(i-1))}{L_j(i)} (j = 1, 2) \]  

\( p_{\text{decay}} \) represents the decay in probability of success, and has typical values between 0.95 and 0.99 if the strategy in step \( i \) was search and the goal has not been reached. In the case of reflection in step \( i \), \( p_{\text{decay}} = 1 \). The \( p_{\text{decay}} P_j(i-1) \) part of the equation takes care of the decay of existing knowledge. However, new knowledge is added to the model as well, and this new knowledge starts out with the “optimistic” probability of success of 1. The \( (L_j(i) - L_j(i-1)) \) part of the equation takes care of that aspect. So on each search step, the probability of success decreases due to decay, and increases due to the addition of “fresh” knowledge.

The costs of reflection depend on two factors. The first is that the costs are higher if there is less basic knowledge, since higher level knowledge has to be based on more primitive knowledge. The second factor is that the costs are higher if there is already a lot of higher level knowledge. This reflects the idea that there is only a limited number of good ideas to come up with, and that it will be more difficult to discover a new idea if there is less to discover.

\[ C_{\text{ref}}(i) = C_{\text{base}} + \left( c_1 \frac{L_{1_{\text{max}}}}{L_1(i)} \right) + \left( c_2 \frac{L_2(i)}{L_{2_{\text{max}}}} \right) \]  

This equation assumes reflection has a certain base cost \( (C_{\text{base}}) \) that is increased by two factors: \( c_1 \frac{L_{1_{\text{max}}}}{L_1(i)} \) which decreases as level 1 knowledge increases, and \( c_2 \frac{L_2(i)}{L_{2_{\text{max}}}} \), which increases as level 2 knowledge increases.

Finally we have to say something about time, since we have talked about “steps” in the previous discussion. Each step takes an amount of time which can vary. So, following the ACT-R intuition that cost and time are related to each other, we take the estimated cost of the strategy at step \( i \) as the amount of time step \( i \) takes:

\[ T(i) = T(i-1) + C(i) \]  

where \( C(i) \) is either \( C_{\text{search}} \) or \( C_{\text{ref}}(i) \), depending on the strategy at step \( i \).

Results

If the appropriate constants and starting values are chosen for the variables described above, we can calculate the increase in knowledge over time. The model is simulated using a spreadsheet program, in this case Microsoft Excel. Note that the model assumes that the goal is never reached, so the results simulate a participant that never succeeds in reaching the goal. Figure 5.2 shows the value of...
A dynamic growth model

$L_1$ and $L_2$ with respect to $T$, and the corresponding evaluations for search and reflection. At the start of the task, search is superior to reflection, but as search fails to find the goal, and the basic (level 1) knowledge increases, reflection becomes more and more attractive up to the point (at $T=155$) where reflection wins from search. Since reflection leads to an increase of level 2 knowledge, search again becomes more attractive (using the newly gained knowledge), and since the cost of reflection increases with the amount of level 2 knowledge already present, reflection becomes less attractive. As a result search will again dominate for a while, up to $T=262$ where reflection wins again. We assume problem solving continues until both expected outcomes drop below zero, since then neither strategy has a positive expected outcome. In the example, this is the case at $T=533$.

As noted, $G$ is the value of the goal. Using a lower value for $G$ corresponds to the fact that a participant values the goal less, and is less motivated to reach it. If we calculate the model for $G=15$ instead of $G=20$, we get the results as depicted in figure 5.3. The

Figure 5.2. Value of level 1 and level 2 knowledge (top) and the expected gains for search and reflection (bottom) for $G=20$
result is that reflection occurs only once, and later (at T=239). Furthermore, at T=393 both evaluations drop below zero, so a less motivated individual gives up earlier. If G is further decreased to 12, no reflection at all takes place, and the give-up point is at T=277.

5.4 The nature of learning strategies

The dynamic growth model nicely describes the phenomena around insight in the literature and in the scheduling experiment. Furthermore, it explains why this behavior is rational. It also predicts changes in strategy due to motivational factors. It however poses new questions. What is the nature of the basic and higher-level knowledge? How will the model behave if the goal is reached at some point? What mechanism is responsible for gaining new knowledge, and how is it represented?
In the previous chapter, I proposed to define implicit learning in terms of learning by the mechanisms of the architecture, and to define explicit learning by activity of explicit learning strategies. In this sense, learning that occurs during search is implicit, since during search the goal is to solve the problem, not to learn something new. During reflection, on the other hand, the goal is to find a new way to approach the problem, so the goal is to discover something new. In this sense, reflection can be seen as explicit learning. As I have argued, there is no principal distinction between the knowledge learned by implicit learning and the knowledge learned by explicit learning, hence there is no real distinction between level 1 and level 2 knowledge in the dynamic growth model. It is just that level 2 knowledge might be more useful, because it has been constructed in a more clever way.

How to get more insight into learning strategies? As we have seen, they are a source of individual differences. On the other hand, there are explicit strategies that at least all adults share, as we have seen in the case of rehearsal. But even in the area of rehearsal, some people prefer to memorize items by verbal rehearsal, while others prefer memorizing information by visualizing it in some fashion. Since learning strategies that are unique for certain individuals are hard to investigate, I will focus on strategies that most adults share, and see how they develop in children.

**Piaget’s stage theory**

The first to acknowledge the fact that children reason in a different way than adults do was Jean Piaget (1952). Based on many experiments, among which the famous conservation experiments, Piaget concluded that children from different ages solve problems in different ways. He proposed a theory of stages, in which children in higher stages can reason more abstractly than children in lower stages. An example is the fact that very young children, who are in the first sensorimotor stage, only reason about objects that are in their field of perception. Once an object is hidden it is considered non-existent. In the second, pre-operational stage, children have mastered the concept of object permanence, and know an object is still there, although it cannot be seen at the moment. The transition between stages is a discontinuous jump: a child either has or hasn’t mastered the concept of object permanence. Piaget’s four stages are very strict: if a child moves to a new stage, they do so for all skills in all domains at once. It turned out that Piaget’s theory was too strong. Children can be taught skills that belong to a stage they have not reached yet, and children may be in different stages in different cognitive domains. Piaget was well aware of this problem, to which he referred to as “horizontal décolage”.

The mechanism that causes these discontinuous jumps is adaptation, which, according to Piaget, is a result of assimilation and accommodation. During assimilation elements from the external world are added to the knowledge of the child. Accommodation, on the other hand, is an internal process that modifies the assimilatory scheme on the basis of the assimilated experiences. So accommodation
can be seen as the process that produces “new” knowledge, and causes the sudden jumps. In order to do so, it needs the accumulated knowledge gained by the assimilation process.

**Fischer’s levels**

A modern version of Piaget’s theory by Kurt Fischer (1980) tries to remedy the flaws in the original theory. His theory has no less than thirteen stages or levels as he calls them, grouped into four tiers. He distinguishes between two levels of performance: the *functional level* and the *optimal level*. The functional level is the level a child performs at in a “normal” situation. There may be large variations in this level across domains. At the functional level, a child is no longer in a single stage, but has a different level of development for each cognitive domain. The optimal level, on the other hand, is the highest level that an individual can produce, and is attained
The nature of learning strategies

when given high levels of support and opportunities for practice. The fact that levels of development can differ across domains makes Fischer’s theory more realistic, but weaker than Piaget’s. A strong point of the theory is however that Fischer describes the kind of representations that are used at each level, and how they can be combined to reach a higher level. In that sense, the theory is much more precise than the original Piaget theory.

From the viewpoint of learning strategies, the optimal level can be associated with the learning strategies that are available to a child. A skill that is beyond the child’s optimal level is a skill for which it lacks the right learning skills. That does not imply that the child has already learned everything it could possibly learn given its current learning skills. For each domain, the child has acquired some of the domain-specific knowledge it can possibly gain given its current learning skills. This level can be associated with the functional level. To get from the current functional level for a skill to the optimal level, the child just has to learn additional domain-specific knowledge using its current learning skills. To go beyond the optimal level, new learning skills have to be acquired first.

Figure 5.4 is an illustration of some of the levels, in this case the third tier applied to the topic of what type of behavior agents can carry out. At the level of single representations, the top level in the table, children can represent that people or animate objects can carry out concrete actions and have concrete characteristics. They cannot yet combine these representations. At the next level, simple combinations of agent-behavior tuples can be made, for example: if you are mean, I will be mean. These combinations remain isolated, however, so there is no generalization of relationships between agent-behavior tuples. At the level of representational systems, combinations of representations are no longer isolated, but generalized. Instead of having a collection of combinations of representations, the actual mapping between representations is understood. At the final level of this example, the level of single abstractions, mappings between representations are combined, leading to concepts like intentions: the intention of a person influences the actual behavior they show while interacting. The complex pattern of interactions between mappings between representations are collapsed into new units: abstractions. In the next tier, abstractions are combined in the same manner as representations in this tier: first by simple combinations, later by systems, and finally by systems of systems.

An important property of Fischer’s theory is that the representations used at a certain level are combined in the next level, either by forming combinations, as in the shift from single units to mapping, or by generalization, by combining a set of mappings into a system. So, the end-products of a level are the building blocks for the next level. A simple experiment that shows that young children cannot combine representations in the same way older children can is the discrimination-shift task by Kendler & Kendler (1959). In this experiment, children are presented with blocks
that are either white or black, and either small or large. The children have to say either “yes” or “no” to each block. For example, they have to say “yes” when a white block is shown, or “no” when a black block is shown. The children do not know this, but have to discover this on the basis of feedback. After a child has made 10 consecutive correct predictions, the criterion is changed, unbeknownst to the child. Either a reversal shift is made, in which “yes” has to answered in response to black blocks, or an extra-dimensional shift is made, in which the dimension is changed, and the child has to answer “yes” when a large block is presented (figure 5.5). After the shift, the number of trials the child needs in order to be able to do ten consecutive correct trials again is counted. Figure 5.6 shows the results of a discrimination-shift experiment in which participants were children of 6-7 years old (Kendler & Kendler, 1959). Fast-learning children discover reversal shifts quickly, but need a lot more trials to discover an extra-dimensional shift. Slow-learning children show a pattern that is entirely opposite: they are faster at an extra-dimensional shift, while needing much more time for a reversal shift. Similar experiments have shown that adults are also faster at reversal shifts (for example, Harrow & Friedman, 1958), while small
The nature of learning strategies

children and animals (for example rats in Kelleher, 1956) are faster at extra-
dimensional shifts.

In terms of Fischer, the knowledge needed to successfully do this particular
discrimination-shift task can now be stated. The most compact representation is an
Rp3-system (the third row in figure 5.4), in which the color (or size) of the block has
to be mapped onto the response (yes or no). Before a shift takes place, the following
system has to be learned:

\[
\begin{array}{cccc}
\text{COLOR} & \text{WHITE} & \leftrightarrow & \text{RESPONSE} \\
\text{BLACK} & & & \text{YES} \\
& & & \text{NO}
\end{array}
\]  \\
(5.10)

A property of the block, its color, has to be used to select a response. If a child has
not mastered Rp3-representations yet, it has to use representations of one of the
lower stages of development, for instance the S2 stage of sensorimotor mappings.
This stage is not shown in figure 5.4, but is two levels below the S4/Rp1-level. At
this stage, it is not yet possible to reason about individual properties of an object,
but just about the object as a whole. The knowledge needed before the shift has to
be represented by a set of four sensorimotor mappings:

\[
\{ [\text{SMALL-WHITE-BLOCK}—\text{RESPONSE-YES}],
[\text{LARGE-WHITE-BLOCK}—\text{RESPONSE-YES}],
[\text{SMALL-BLACK-BLOCK}—\text{RESPONSE-NO}],
[\text{LARGE-BLACK-BLOCK}—\text{RESPONSE-NO}] \}
\]  \\
(5.11)

If we now look at the changes required in each of these representations to
accommodate the different types of shift, we can understand why reversal shifts are
easier if you use Rp3 representations, and extra-dimensional shifts are easier if you
use just S2 representations. In the Rp3 case (figure 5.7a), the reversal shift is easier,
because the system remains the same: only the mapping within the system changes.
In the S2 case (figure 5.7b), the extra-dimensional case is easier, since only two out
of four mappings change, while two mappings remain the same. In the reversal
shift all four mappings change.

In the introduction to this section I remarked that reflection corresponds to the use
of explicit learning strategies. Since learning strategies themselves have to be
acquired as well, it interesting to look at the development of reflection and the
relation with Fischer’s theory. Kitchener, Lynch, Fischer and Wood (1993) have done
a study in which they relate Fischer’s skill levels to reflective judgement. Each level
from Rp1 upwards can be related to an increased capacity of reflection. While
children at the Rp1 level can only reason about concrete propositions, like “I know
the cereal is in the box”, children at the Rp3 level can reason about the uncertainty of
knowledge. Kitchener et al. developed the Reflective Judgement Interview to assess
the level of reflection, and used participants who were between 14 and 28 years old.
The results show a steady increase in reflective capacity. Moreover, a specific version
of the test was used to assess the optimal level of performance by giving maximal
contextual support. In this version of the test the growth curve shows some evidence for growth spurts, as predicted by Fischer’s theory (figure 5.8).

In summary, Fischer’s theory is weaker than Piaget’s with respect to the predictions it makes. This is not a big problem, since Piaget’s original theory is not completely accurate. On the other hand, Fischer provides representations that can be used to analyze skills in different stages of development. These representations can also be

5: Strategies of learning

Figure 5.7. Changes in representation (indicated in an outlined font) due to reversal and extra-dimensional shifts using different types of representation. Abbreviated versions of (5.10) and (5.11) are used.

Figure 5.8. Increase in reflective judgement with age. From Kitchener et al. (1993)
The nature of learning strategies

used to describe developmental paths that lead from one stage to the next stage. In this sense Fischer’s theory is stronger than Piaget’s theory: it can specify how knowledge is represented, and how a higher-order representation can be built out of lower-order representations. It still lacks a real processing component, however, a specification of the processes that actually change the representations. Furthermore, Fischer’s representations in their current form are not precise enough to support a detailed processing theory. This is also the main criticism of stage theories of development, the fact that they put too much stress on the state of knowledge at a certain age, thereby neglecting the importance of what some researchers see as the main issue of interest in development: the process of change.

The dynamics of change in Fischer’s theory can be described by dynamic systems theory. Van Geert (1994) has developed models of the increase in knowledge on different levels, using growth equations similar to those presented in section 5.3. An interesting feature of van Geert’s model is that it can model the shape of the growth spurts, such as the slight regression in performance between age 17 and 18 in figure 5.8, followed by a fast increase between age 18 and 20. As the model is coarse-grained, it does not describe the changes in representations, nor can it explain by what changes a new level starts. Nevertheless, a dynamic growth model may be a good starting point for constructing a fine-grained model that does model knowledge representations.

Karmiloff-Smith’s representational redescription

A theory that puts more stress on the process of change than on levels of knowledge is Annette Karmiloff-Smith’s (1992) theory of representational redescription (RR). The RR theory is concerned with mastering skills in specific domains, so it has no global Piaget-like stages or Fischer-like optimal levels. An interesting feature of the theory is that it discriminates an implicit learning phase for a new skill, followed by several explicit learning phases. In each new phase, the representations of the previous phases are redescribed into a new representation. The phases are called I (implicit), E1 (explicit 1), E2 (explicit 2) and E3 (explicit 3). The last two phases are often collapsed into a single E2/3 phase. The difference between a phase and a stage is that phases are not related to age, and the cycle of four phases recurs for every domain that has to be mastered during development.

According to the RR theory, the I-phase in learning a new skill involves implicit, data driven processing. In this phase, the child creates “representation adjunctions” out of the external data, which are just stored in memory. No further processing is done on these representations, but they can contribute to successful performance. If the child has accumulated enough adjunctions, performance becomes consistently successful. The RR theory defines this as behavioral mastery. Although the child can perform the skill, it does not have conscious access to it, since the examples are not generalized into rules. Generalization takes place in the E1 phase, in which the focus is moved from external data to internal representations. Features from the
5: Strategies of learning

environment are disregarded in favor of the internal generalization process. This may lead to a decrease in performance, since generalizations may be wrong. In E2/3, the internal representations are made consistent with the external data, leading to a representation that supports successful performance, and offers the building blocks for new skills.

Each phase produces its own type of representations. The “representational adjunctions” are stored in procedural form. This procedural form is not the same as production rules in ACT-R, but shows a strong resemblance to popped goals that are stored in declarative memory. In the E1 phase, the representational adjunctions are redescribed into more compact abstractions that can be related to other domains. These abstractions are recoded in E2/3 into a representation that is available for conscious manipulation, and that can be verbalized. An important feature of these representations is that they all remain available, so even if a child has reached phase E2/3, the representational adjunctions are still available. In chapter 6 we will discuss some ACT-R models in which the ideas of representational redescription will be used and made precise in terms of ACT-R representations.

Siegler’s overlapping-waves theory

Siegler (1996) criticizes the stage, level and phase models by pointing out that the idea of a stage may well be an artifact of the way developmental psychologists collect their data. Typical experiments involve studying how two or more age groups of children perform a certain task, and contrasting their respective approaches. According to Siegler, however, it is a mistake to think about the way children think about a certain problem at a certain age. The result of these approaches are staircase models. For example, several strategies to do simple additions have been identified in children: small children tend to count both addends from 1, slightly older children start with the largest addend (the min strategy), and even older children retrieve the answer from memory (Ashcraft, 1987). A “staircase” interpretation of these differences is depicted in figure 5.9: first children use the sum strategy, then they switch to the min strategy, and finally to the retrieval strategy. Closer inspection of what strategies children use reveals that children do not use a single strategy to solve addition problems, but instead use several strategies. What changes with age is the frequency with which they use a certain strategy. The bottom graph of figure 5.9 illustrates this aspect using a study from Svenson and Sjoberg (1983). In this longitudinal study, the strategy use of 13 children was followed from first to third grade. As can be seen in the graph, at each point in time children use several strategies, and the frequencies of particular strategies fluctuate over time.

The main point Siegler makes is that children do not change strategies overnight. When a child discovers or learns a new strategy to do addition, it does not exclusively switch to this strategy but adds it to the set of existing strategies with
which it has to compete. If a strategy proves to be sound in the long run, and has an edge over other strategies, it will be used more often.

In chapter 3, we saw that some participants in the scheduling experiment sometimes use counting to do addition, which corresponds to the min strategy. This corresponds well with the overlapping waves model: even adults have all strategies available, but most adults just use retrieval as their sole strategy. Some individuals may however use other strategies occasionally. The fact that addition had to be performed in a situation where working memory load was already high may also have contributed to a shift in strategy. The matter of working memory load will return in chapter 7.

**Discussion**
The goal of this section was to get some idea of what learning strategies are by looking at development. Each of the four theories discussed offers some parts of the puzzle. Unfortunately, all four theories are mainly descriptive, and are not very specific about exact representations or processes acting on these representations.
An important topic in development is domain specificity. Although Piaget’s theory of pure global development has turned out to be too strong, the presence of some global factor is still under debate. Fischer and Karmiloff-Smith seem to contradict each other on this point. Fischer defines a global optimal level of performance at a certain age. When this level goes up, there is a global increase in development. This global increase is not witnessed in the way Piaget envisions it, because performance in specific domains may still be lagging behind. Karmiloff-Smith’s RR theory only describes development within a domain, without any need for global development.

One might ask whether it is at all possible to settle this debate on the basis of empirical evidence. In Fischer’s theory, it is always possible to define an optimal level: it is just the level of the domain that has progressed most. In order to assert an optimal level that is really meaningful, it has to offer some additional support to the learning process. Although it may be very hard to find empirical evidence, a modeling perspective may offer some sort of support.

One issue a model may resolve is whether it is at all possible to have knowledge that is useful for all domains. If such knowledge can be defined and represented, for example in ACT-R’s representations, the next step is to find a developmental path through this knowledge, and to specify how a more refined strategy can be learned from a more primitive one. If a system like this can be developed, and is capable of offering new explanations for old phenomena, it might offer a new type of evidence in the discussion. But in order to build such a system, the mechanisms of change have to be understood. The theories discussed here can offer some clues.

Karmiloff-Smith suggests the first (I) phase in learning a new skill is to store representational adjunctions. This phase only involves storing, retrieving and applying these adjunctions. Only when this set is sufficiently stable in the sense that behavioral mastery is reached, the explicit phases in which the information is integrated can be entered. This idea closely matches Piaget’s idea of assimilation and accommodation: during assimilation external experiences are stored, while during accommodation these experiences are integrated into a qualitively new behavior.

Siegler’s theory of overlapping waves shows that the discovery of a new strategy does not necessarily imply that this strategy will completely dominate behavior. A new strategy first has to prove it is better than the existing strategies. This illustrates the need for an evaluation mechanism: any new strategy has to be assessed with respect to the question whether it really is useful and better than the alternatives.

What have we learned with respect to learning strategies? Take Fischer’s theory as a starting point. Each new level in the theory involves a type of representation in which a single representation replaces a combination of representations from the previous level. Assuming these representations are mainly declarative, one needs accompanying procedural knowledge in order to handle these representations.
Modeling explicit learning strategies in ACT-R

Which of these comes first? In terms of ACT-R, the declarative representations have to be first, because a declarative example is needed to learn a new production rule. This also concurs with the RR model in which a set of representations is acquired and stored in the first phase. Only when a suitable set of knowledge is collected can generalization be attempted. Probably many generalizations are possible, so sorting them out may take some time, and may cause the rise and fall of certain strategies as Siegler has shown. Summarizing,

- learning strategies have to be general, so they can be used in several domains
- it has to be possible to find some developmental path through these learning strategies
- representing, storing and retrieving examples is an important first step in acquiring a new strategy
- since several generalizations are possible, an evaluation mechanism is needed to select the most useful strategies

In the remainder of this chapter, I will show a potential example of a general learning strategy, thus addressing the first point on the list. This strategy will be explored in models of two separate tasks. An interesting property of the strategy is that once it is impoverished by removing some of the production rules, it exhibits behavior consistent with a lower level of development. This property is important for the second point: the developmental path through strategies. The models in the remainder of this chapter will model the discovery of new rules, so accommodation in terms of Piaget, or the E1-phase of Karmiloff-Smith. The aspect of assimilation or I-phase, i.e. the use of examples, will be an important topic in the next chapter, as well as the evaluation mechanism.

5.5 Modeling explicit learning strategies in ACT-R

The goal of an explicit learning strategy is to learn new knowledge that is necessary for some new task or domain, or to improve the knowledge already available for an existing task or domain. In order to model this in terms of ACT-R, general learning goals have to be defined, and production rules that operate on these goals. The starting point for learning goals is the predefined dependency chunk-type (see figure 2.9 in chapter 2). Dependency chunks form the basis for new production rules: once a dependency is popped from the goal stack, it is compiled into a production rule. Intuitively, the best way to think of a dependency is to consider it as an example of how to do something. The goal of coming up with such an example can therefore be seen as an explicit learning goal. Eventually, this learning goal will produce a new production rule. In ACT-R, the dependency learning goal needs production rules that matches it. These rules are therefore also part of explicit
learning, and have to be domain independent. So at least the production rules that operate on dependencies are explicit learning strategies for learning new procedural knowledge.

When are explicit learning goals needed? As we have seen earlier in this chapter, we need them if the current approach to the task does not work well. But they are also needed, in the case of a psychological experiment, when participants have to do a task they have never done before, as is often the case. Participants in a psychological experiment need explicit learning strategies to set up initial knowledge structures to perform the task. These strategies need some domain-specific information to work with, for example the following types of information:

*Task instructions and examples.* In the case of an experiment or educational setting, a task or problem is explained by the experimenter or teacher, and sometimes a few examples are shown.

*Relevant facts and biases of other domains in declarative memory.* New tasks often build on existing knowledge. Knowledge from related domains can therefore be retrieved and adapted to the task at hand.

*Facts and biases in declarative memory from the current domain.* As someone gains experience in a new domain, popped goals are accumulated in declarative memory, while declarative learning maintains activation levels and associations with other chunks. This declarative knowledge, similar to the RR model’s implicit I-phase knowledge, may serve as a basis for new production rules.

*Feedback.* If a wrong answer is given based on the current knowledge, and feedback is provided on what the right answer is, this may also be used as a basis for new rules.

Figure 5.10 outlines how a learning strategy works: given initial information in declarative memory, a set of general production rules creates an example of how to do something, a dependency. This dependency is compiled into a new production rule, which has to compete with the rules that have created it. If the task-specific rule performs too poorly, the explicit learning strategies win the competition, and propose new rules, taking into account the feedback (if any) received on the faulty rule. The competition between the task-specific rules and the general learning strategies is the same competition as the competition between search and reflection modeled in the dynamic systems model earlier this chapter.
5.6 An ACT-R model of a simple explicit strategy

The beam task

The task we will start with is a beam task. It is a simplified version of the balanced-beam task, a task of used in developmental studies (Siegler, 1981). 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 remain in balance. The numbers on the labels have no influence on the outcome. Figure 5.11 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.

The assumption is that the model initially has no task-specific rules about beam-problems. The only procedural knowledge the model has is a set of general rules. Later on, we will use the same general rules for other tasks. The general rules used to learn this task are the following:

---

Figure 5.10. General schema of learning strategies in ACT-R

Figure 5.11. Example of the beam task
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. This rule creates a rule that directs attention to a certain aspect, attribute or dimension 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. The resulting rule will, given a goal, try to fill in the answer using some retrieved fact from declarative memory. 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 that the chunk with the highest activation is retrieved. Since activation represents the odds that a chunk 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 such as 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 favor 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. Although this is not part of the model presented here, a possible source of the relevant associations is an implicit learning phase in the sense of the RR theory as discussed in section 5.4.

Since the general rules are just production rules, they can be in direct competition with the task-specific rules they generate. If property-retrieval generates a rule X to retrieve the label, X will compete with property-retrieval. If X is not performing well, for example if it retrieves the irrelevant label, 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 (i.e., 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 5.12 summarizes the property-retrieval rules, and figure 5.13 summarizes the find-fact-on-feedback rules. Both are instantiations of figure 5.10. Figure 5.13 shows...
An ACT-R model of a simple explicit strategy

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:

Property-retrieval

![Diagram of property-retrieval process]

Find-fact-on-feedback

![Diagram of find-fact-on-feedback process]
IF the goal is of type SOLVE-BEAM and refers to two objects O1 and O2 of which no properties have been retrieved yet AND there is a property of O1 of type weight and value V1 AND there is a property of O2 of type weight and value V2 THEN add V1 and V2 as properties of type weight to the goal

One of the rules generated by find-fact-on-feedback is a rule that predicts when the left arm of the beam will go down.

IF the goal is of type SOLVE-BEAM and two properties V1 and V2 of type weight have been identified AND there is a fact of type greater-than that specifies V2 is greater than V1 THEN set the answer slot of the goal to LEFT

The model was tested in several conditions, differing in the bias given for the properties (P) and the fact-type (F). The following table summarizes the conditions:

<table>
<thead>
<tr>
<th>P</th>
<th>Association between beam and weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-</td>
<td>Association between beam and label, a bias for the wrong property</td>
</tr>
<tr>
<td>F+</td>
<td>Association between beam and greater-than</td>
</tr>
<tr>
<td>F-</td>
<td>Association between both beam and greater-than, and beam and number, so two possible fact-types were favored</td>
</tr>
<tr>
<td>F--</td>
<td>No associations between beam and fact-types, four fact-types are possible</td>
</tr>
</tbody>
</table>

Each experiment has both a P condition and an F condition. Each experiment was run 30 times for 45 trials. Figure 5.14 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 does not 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. Success depends heavily on the quality of the declarative information. This information does not have to be completely accurate, but some declarative stage before proceduralization is important for success.

The results in figure 5.14 suggest a gradual increase of performance. However, this impression is misleading, as it is caused by averaging 30 runs. If individual runs are examined, each has a certain point where performance increases dramatically. To get a better perspective on this increase, it is necessary to find the exact point at which the increase in performance starts. In one of the conditions, the P-F+ condition, this point is the most obvious: the moment the model switches from examining the label property to examining the weight property. Since this moment is easy to identify in an individual run of the model, it is possible to average results with respect to this point in time. An interesting aspect to average is the number of failed predictions the model makes before it makes the right predictions. Remember the model keeps trying to predict the right answer until it is successful. The result is shown in figure 5.15. It shows the average number of incorrect tries for each trial in the P-F+ condition. At x=0 the model creates a production rule that retrieves the weight properties. As is apparent from the graph, before ACT-R creates this rule, on average three failed predictions are made. Since this clearly establishes that the current task-specific rules are not correct, the general rules can take over and propose new task-specific rules. This process resembles the impasse-insight stages of insight problem solving, and is based on the same mechanisms of the dynamic growth model.
One of the advantages of explicit learning strategies compared to implicit learning is that they can handle change more easily. If something changes that has been stable for a while, 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, which I have explained in section 5.4. The ACT-R model of adult behavior uses the same 8 general production rules 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 beam task: a rule that focuses on one of the properties of the blocks, either the size or the color, and rules that map specific colors or sizes onto the answers yes and no. This knowledge is closely related to the Rp3-representation of Fischer’s theory (figure 5.7). The small-child/animal model uses only 2 of the 8 general production rules, implementing a limited find-fact-on-feedback strategy. The latter model hardly uses any explicit reasoning at all, but rather stores regularities in the environment in production rules. This representation closely resembles Fischer’s S2-representation. The results of both ACT-R models are shown in figure 5.16b, producing results quite similar to the Kendler & Kendler data in figure 5.16a.

Despite the fact that the discrimination-shift task is generally not considered to be an insight problem, it nevertheless requires the participant 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.
5.7 Discussion

The goal of cognitive modeling is to create computer simulations of cognitive processes. A criterion for a good model is 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 specialized production rules is less convincing than a model that gathers its own knowledge. The learning mechanisms which are part of the architecture, are often not capable of doing this job by themselves, so they need augmentation. In the previous chapter I have argued that these mechanisms correspond to implicit learning. The mechanisms can be augmented by explicit learning, that is, implemented by knowledge in memory that directs the implicit learning mechanisms.

Implicit mechanisms are fixed, but explicit strategies have to be acquired. Individuals probably differ in their explicit strategies, although they may well have many 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 other learning strategies. Is there a sequence of rules that unfolds during development? The model of the discrimination-shift task at least hints in this direction, as does Fischer’s theory. On the other hand we may well expect large individual differences. Experiments in which participants have to solve difficult problems often show that every participant 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, gaining 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 if the time taken to learn something is taken into account. Neural networks usually do not have any goal structures, so they lack the mechanisms that are able to focus learning. Karmiloff-Smith, for example, states that neural networks model implicit I-phase learning very well, but are not yet capable of modeling the more explicit phases of skill learning. Raijmakers, van Koten and Molenaar (1996) have shown that a standard feedforward neural network always behaves like a small child or animal in the discrimination-shift task, being faster at the extra-dimensional shift. To summarize: 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 goals and intentional structures.

In the Soar architecture (explained in chapter 2), 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 use deliberation when existing simple rules prove to be insufficient and, more importantly, if there is any knowledge present on how to deliberate. 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 an infinite sequence of impasses. Soar’s single learning mechanism is chunking, which summarizes 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 Soar’s only learning mechanism, this may cause trouble. For example, to learn simple facts, Soar needs the elaborate scheme of data-chunking. Data-chunking eventually produces rules like “IF bird THEN note it has wings”. To be able to learn this, however, a lot of deliberation has to be done by production rules that are not part of the architecture. In a sense, Soar walks the reverse way: instead 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 Soar’s problems with simple satisficing behavior in Johnson (1997).

Since many other architectures, like EPIC and 3CAPS, 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 circumvent this problem by using a generic goal type for all goals, but this is hardly a satisfactory solution in the long run.

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? Unfortunately, I will not answer this question in this thesis: in the next few chapters we will focus on adult problem-solving behavior only.