



8: Concluding remarks

The goal of this thesis, as stated in chapter 1, is the development of a theory of problem solving that is psychologically plausible, and does justice to its complexity. The weak-method theory that stems from artificial intelligence acknowledges this complexity, but shows only very limited correspondence with human data. Theories from experimental psychology, on the other hand, neglect the complexity of problem solving, and theorize about how complex skills can be explained in terms of memory systems and basic information processing. In the course of this thesis, the subject of problem solving has been broadened to cognitive skills in general, since there is no principled distinction between the two. The approach I have chosen is to focus on learning cognitive skills. In chapters 5 to 7, a view of skill learning has taken shape, which I will try to explicate in this final chapter. This view is built upon the foundation the ACT-R architecture offers, ensuring a plausible system of learning and memory, and it exhibits the complexity of behavior that artificial intelligence is interested in. The theory of skill learning that emerges has a number of areas that need more investigation, and a number of possible applications. Both topics will be discussed in the final sections of this chapter.

Have I solved the problem of NP-completeness? The answer to this question is of course “no”. But I have tried to draw a picture of how humans can acquire the knowledge to at least partially solve hard problems. This is not a simple picture, and cannot be summarized in a clear-cut algorithm. When people start with a new task, they do not use a general machine learning algorithm to acquire knowledge about this task. Rather, they use a set of strategies, knowledge about other domains and tasks, instructions about the current task to start with, and keep adapting and refining their knowledge while they are working on the task. A better understanding of these processes is the key to understanding complex human problem solving.

8.1 *The skill of learning*

Skills are not separate entities, they almost always rely on other skills. In order to learn multiplication, one has to master addition first. In order to be able to add numbers, one has to be able to count, etc. The diagram I used in chapter 1 to outline the growth of knowledge for a certain problem (figure 1.4) is useful to intuitively sketch skill learning in general. Let's visualize the space of possible tasks in a two-dimensional diagram (figure 8.1). Again, the idea is that tasks that are similar are close to each other in the diagram. The vertical axis is used to indicate complexity. A task is higher in the diagram if the skill to perform this task relies on skills associated with a task lower in the diagram. The result is a partial ordering. Each task can be instantiated in numerous ways, so each of them is shown as a small set (a box). Now, if the set boundaries are dissolved, we're back at figure 1.4, except that the diagram now represents the set of instances of all possible tasks.

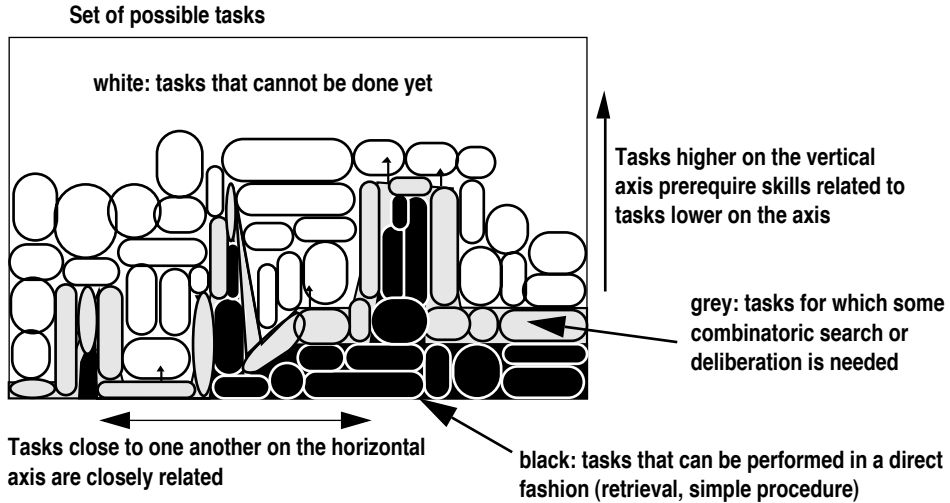


Figure 8.1. The set of possible tasks

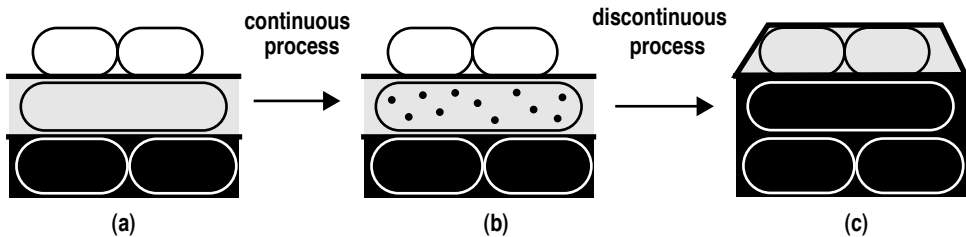


Figure 8.2. Summary of the skill acquisition process. The figure shows a detail of figure 8.1. (a) The rectangle in the middle represents the task to be learned. The two black rectangles at the bottom represent prerequisite skills, which already have been mastered. Experience with the task produces examples or instances (b). Once there are enough examples, generalization can be successful (rule learning), producing situation (c).

Figure 1.4 gave the impression of a gradual learning process, in which the boundaries between the black, grey and white areas slowly move upwards. On the basis of chapter 5 to 7, it is possible to revise this picture, and to show how discontinuous learning can occur. Suppose a new skill is being learned. To be able to start at all, the prerequisite skills have to be mastered first (i.e., the have to be in the black area). This means that the task itself has to be in the grey area (figure 8.2a). Experience with the task will produce examples of solutions. These examples can be retrieved later on, producing “specks of black” in the grey area that represents the new skill (figure 8.2b). This process is similar to instance learning, Piaget’s assimilation, implicit learning and the I-phase of the RR-theory. Although it seems that the black specks will eventually fill up the whole set, this is only true for tasks with a finite number of instantiations. Mastery of an infinite set of instantiations

8: Concluding remarks

would require an infinite number of examples. To master an infinite set as a whole, one or more generalization steps are necessary. Once a successful generalization is made (possibly after several unsuccessful ones), it is suddenly possible to efficiently solve the set as a whole (figure 8.2c). The process of generalization is similar to rule learning, Piaget's accommodation, explicit learning and the E1-phase of the RR-theory. A side effect of successful mastery of a skill is that it is now possible to learn skills that were previously unreachable. This means a sudden shift in the border between the grey and the white area (figure 8.2c).

An important aspect of this process is that the generalization step is a skill itself. As we have seen in chapter 5, adults use a more elaborate form of generalization than children do. It is this aspect of learning that makes human learning virtually limitless. Understanding this particular set of skills may well be the key to understanding many aspects of individual differences and cognitive development. In the next section, an ACT-R implementation of this idea will be outlined, based on the models from chapter 6 and 7. In section 8.3, I will elaborate on the issue of individual differences.

8.2 Processes involved in skill learning

In chapter 6, I proposed a first version of a general schema of skill learning (figure 6.1). Although this schema offered a good description for the two models discussed, the Sugar Factory and the Fincham task, it did have a problem: the vague notion of the initial method. In both the Sugar Factory model and the Fincham model these initial methods were hard-coded into the model. The origin of this knowledge was left unspecified. The scheduling model in chapter 7 showed how this problem can be solved. In this section, I will update figure 6.1 according to the modifications introduced in chapter 7. The final skill learning theory I propose encompasses two currently dominant theories, the theory of rule learning, rooted in the original ideas by Fitts (1964) and further extended by Anderson (1982), and the idea of instance learning, as posed by Logan (1988).

Now how can this theory produce the kind of learning discussed in the previous section? Suppose we have a task in the grey area (as in figure 8.2a). This means some way of doing this task is available, although it is inefficient. This initial method may involve a set of declarative rules that has been adapted through analogy to suit the current task, or is the result of interpreting instructions. Doing the task using these initial rules produces examples that are stored as instances. At some point, the explicit learning strategies will attempt some generalization (in terms of chapter 5: a reflection episode). As soon as the generalization is successful, and the new, efficient declarative rules are subsequently proceduralized, the skill is mastered.

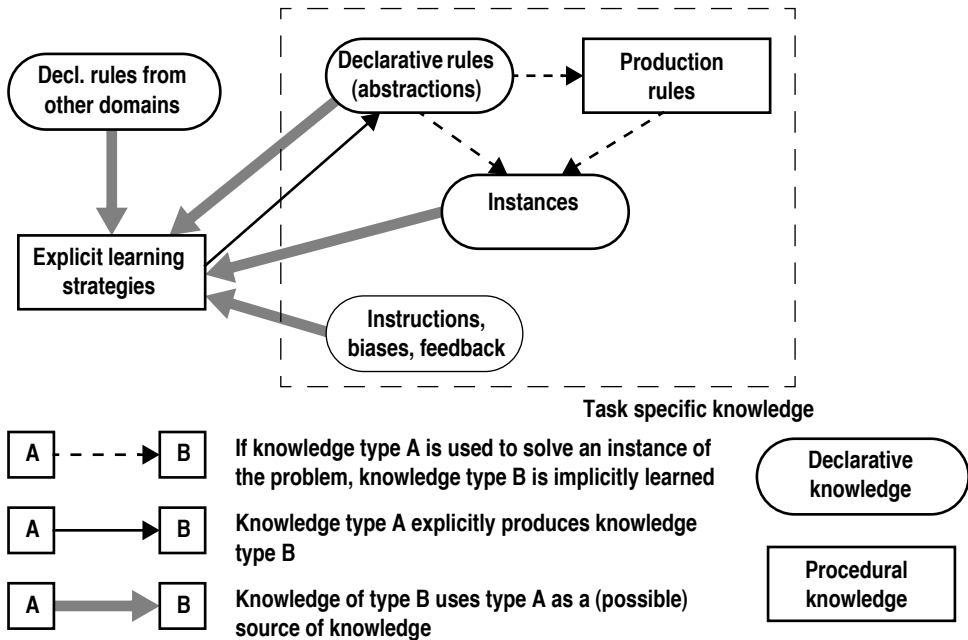


Figure 8.3. Overview of the proposed skill-learning theory

Figure 8.3 shows a revised overview of the theory, an expansion of figure 6.1. Each of the boxes in the diagram represents a type of knowledge. The dashed boundary indicates which of these knowledge types are task specific. Each rectangle represents a type of knowledge that is associated with a certain strategy. Each of these strategies needs knowledge associated with it to function properly. Within the task-specific knowledge, there are three possible strategies, each of which represents a possible way to perform the task: using a declarative rule, using a production rule, or retrieving an instance. According to ACT-R, the strategy with the highest expected outcome will win the competition. The instance strategy is generally the best strategy, followed by production rules, declarative rules and using an explicit learning strategy.

Using one type of knowledge to perform the task results in learning: not only is the knowledge itself reinforced, but it may also produce knowledge necessary for other strategies as a by-product. This implicit learning is represented by the dashed arrows. The strategy outside the task-specific knowledge represents explicit-learning strategies. The goal of this strategy is to produce task-specific knowledge in the form of declarative rules. In order to do this, it can use different knowledge sources, both within and without the task domain. The use of these sources is represented by the grey arrows.

8: Concluding remarks

Explicit learning strategies

There are many possible learning strategies, and they are an important source of individual differences. Explicit learning strategies themselves can also be considered skills, and can be learned in the same way as other skills. Eventually, some bootstrapping problem must be solved, so some initial strategy should probably be part of the architecture itself.

Some possible strategies are:

- Use analogy to apply declarative rules from other domains to the current domain
- Generalize instances into declarative rules
- Create a direct representation of the instructions
- Refine current declarative rules based on feedback

Both the Fincham model (chapter 6) and the scheduling model (chapter 7) use analogy initially, although the Fincham model does so in a task-specific fashion (i.e., the method of analogy was already part of the instructions). In some cases instructions provide for an initial method. In that case the main task is to translate the instructions into declarative rules that can be carried out. But in many cases instructions will also draw on knowledge people already have.

Declarative rules and Instances

Declarative rules have to be interpreted in order to be carried out. As such, they require attention and working-memory capacity. Since they are declarative, these rules can be subjected to deliberate reasoning, which may lead to new declarative rules. Whether or not learning of declarative rules will contribute to performance in the long run, not only depends on the available knowledge, but also on the task. The Sugar Factory experiment in chapter 6 illustrates this: the possibility of control is limited, the behavior of the system is non-linear, and there is random noise involved. Therefore, learning of declarative rules will fail for most participants, so performance can be explained by instance learning alone. In other cases learning of declarative rules may be successful, but will not prove to be the most cost-efficient strategy. This is the case when the set of instances is small, so that instance retrieval can dominate performance. In the Fincham task, the use of declarative rules was a very useful strategy, and in the scheduling task it was crucial, since the instructions alone are clearly insufficient to do the task and examples are not very useful.

Production rules

Production rules serve the same function as their declarative counterparts. But since they can be executed in one step instead of being interpreted, they are much faster. Another advantage is that they use less working memory capacity due to the fact that there is no longer a declarative rule that has to be kept active. Although

production rules provide no new knowledge when compared to their declarative cousins, they make it possible to solve problems that could not be solved before because of limitations in working memory capacity or available time. This may even open the way to learning more difficult skills.

In ACT-R, production rules are learned through a specific type of goals (dependencies). In that sense production-rule learning requires deliberate planning. The encapsulation explained in chapter 6 and 7 turns this goal-directed type of learning into an implicit learning mechanism. In the current models, each time a declarative rule is retrieved, there is a chance that a production rule will be learned. Although this method fits the data quite nicely, there are alternatives, such as a gradual transfer from the declarative to the procedural version of a rule. This raises the question whether procedural memory and declarative memory are really distinct in principle. Is proceduralization not a process of gradually speeding up the process of interpreting declarative rules while reducing interference?

Implicit and explicit learning

The distinction between implicit and explicit learning follows naturally from this theory. Implicit learning is a result of normal processing, producing new instances and proceduralization. Explicit learning is based on strategies. Using these strategies is also a form a normal processing, except with another goal. Instead of performing the task, the goal is to generalize new knowledge. Due to this fact, explicit learning is “conscious”, since the episode can be recalled later on by retrieving the learning goal, while implicit learning is not. Explicit learning is a skill that has to be learned. This fact can explain the large individual differences in explicit learning whether due to age, intelligence or brain damage.

8.3 Individual differences

In this thesis two sources of individual differences have been discussed in detail. In chapters 4 and 5, the idea has been put forward that individual differences stem from differences in explicit learning strategies. Experiments in discrimination-shift learning, for example, exhibit a qualitative difference in learning between young children and adults. The model in chapter 5 shows how this difference can be explained in terms of different strategies. The scheduling model in chapter 7 further suggests that these explicit learning strategies can be subdivided into a set of declarative rules that can be reused, and production rules that adapt these declarative rules to new situations.

Chapter 7 introduces another source of individual differences: working memory capacity, which corresponds to source activation in ACT-R. Correlational studies, for example by Kyllonen and Christal (1990), find correlations of between 0.80 and 0.90

8: Concluding remarks

between reasoning ability factors and working-memory capacity. All the tests in their experiment, however, were relatively short. The model in chapter 7 and the results in the previous section suggest that the advantage of working-memory capacity diminishes due to procedural learning. With respect to individual differences on a larger time scale, explicit learning strategies may become more important. Nevertheless, a high working-memory capacity may imply more flexibility, leading to a larger range of problems that can be solved in a declarative fashion. The evidence up to now is inconclusive with respect to the interaction between working memory capacity and reasoning ability.

Although the supporting evidence is still scarce, the modeling approach to individual differences has great advantages over the traditional approach in IQ-tests. Working-memory capacity and explicit learning strategies are information processing notions. The exact impact of these aspects on performance can be predicted by an ACT-R model. For example, Kyllonen and Christal classify a particular test as a working-memory test on the basis of intuition alone. In that case, working-memory capacity is just a factor that is defined as “what all the working-memory tests measure”. In the case of the digit-working memory task, on the other hand, working-memory capacity is a parameter in the model that can be separated out from, for example, the effect of learning. A further advantage of a parameter is that it leads to precise predictions about individual differences in other tasks. As a result, research in individual differences may move from a descriptive to a predictive stance.

Binet’s (1962; originally published in 1911) original IQ concept was based on ideas about development, in the sense that IQ represents the “mental age” divided by the real age of a child. This idea is, however, no longer valid in modern IQ tests. As was argued in chapter 5, individual differences due to development can best be explained by differences in available explicit strategies. I do not believe that the source activation parameter in ACT-R, the main determinant of individual differences in working-memory capacity, is susceptible to development. Anyone who has played the game of memory with children may attest this. Tests of working-memory span do not measure working-memory capacity only, but also memorization strategies like rehearsal.

The skill-learning theory presented in this thesis may also be useful in the study of cognitive development. Explicit learning strategies are themselves skills, and their development may resemble the development of skills, but on a larger time scale. The table in figure 8.4 shows how we may think about this issue in terms of a time scale of human learning. This table is analogous to the time scale of human action, as originally conceived by Alan Newell (1990). Individual skills are learned and proceduralized in terms of minutes and hours, while general intelligence develops in terms of weeks and years. This is not because a general learning strategy is fundamentally different from a skill. Skills are based on generalizing knowledge to

Time scale of human learning				
Scale (sec)	Time Units	Type of representation	Memory system used	
10^8	years	Learning strategies	Procedural	Develop- mental band
10^6	weeks	Generally useful declarative rules	Declarative	
10^4	hours	Task-specific production rules	Procedural	Skill band
10^2	minutes	Task-specific declarative rules	Declarative	
10^0	seconds	Task-specific facts	Declarative	Instance band

Figure 8.4. Time scale of human learning, analogous to Newell's (1990) time scale of human action

produce specific facts. To go from a fact to a skill takes two orders of magnitude on the time scale (from 10^0 to 10^2). It takes another two orders of magnitude to proceduralize this knowledge. An assumption behind these learning steps is that it includes all aspects that lead to an optimally useful representation, such as learning parameters, discrediting wrong generalizations, etc. To learn knowledge that is generally useful, knowledge related to specific skills needs to be generalized, which takes another two orders of magnitude.

8.4 Evaluation of ACT-R

This thesis could not have been written without the ACT-R cognitive architecture. It has provided both a theory and a modeling environment with just the right level of constraints. ACT-R's constraints actually help the modeler in designing a proper model for his data. Too few constraints leave too many choices, while too many constraints may force the modeler to spend too much time overcoming the limitations of the architecture. ACT-R offers a solid basic system of learning and memory, grounded in empirical data. An ideal architecture would be a system in which it is just sufficient to specify some necessary knowledge, after which the architecture exhibits psychologically plausible behavior. ACT-R comes close to this ideal.

In the course of the research described in this thesis, I often pushed the envelope of the capabilities of ACT-R. As a consequence, I have encountered some aspects of ACT-R that are still underspecified. I think this thesis can contribute to ACT-R by helping to point out and sometimes fill in some of the gaps.

Production compilation

One of the cornerstones of the ACT-R theory is the distinction between declarative and procedural memory. This distinction has proved to be very useful, not only in terms of the theory, but also in providing users of the theory with a relatively easy-to-learn modeling environment. Having two long-term memory stores, however, also produces additional complications. More specifically: ACT-R's current procedural learning is not yet completely adequate. Using dependencies in ACT-R is still too much like programming, and some of the productions that fill slots in a dependency goal lack any psychological plausibility. The method for creating dependencies introduced in chapter 6 and extended in chapter 7 abstracts away from this process by focusing on declarative rules. The actual production rules are learned more or less automatically if their declarative counterparts are used often enough. In order to use these declarative rules, and stay consistent with earlier ACT systems like ACT* (Anderson, 1983), a set of general interpretive productions is necessary, or at least a framework in which they can be defined. The scheduling model in chapter 7 offers some kind of solution, which may need some more elaboration to be useful in any setting.

An issue we should keep in mind, in the spirit of Newell, is not to be too dogmatic about the declarative/procedural distinction. From the fact that declarative knowledge apparently gradually changes into procedural knowledge we may deduct that at a deeper level declarative and procedural knowledge are similar after all. Finding this deeper equivalence not necessarily contradicts the ACT-R theory. Rather, it may strengthen the theory by pointing out how a conceptually useful distinction can be grounded in a more parsimonious, but probably less usable, system.

Chunk types

Another unresolved issue in ACT-R concerns chunk types. There is no mechanism that can learn new types, neither will it be easy to specify one. In most models, this problem does not surface, since most models are only concerned with a specific task. Production rules that implement explicit learning strategies need to be able to operate on several different types of chunks. The solution I have chosen, to use a generic goal type, is not entirely satisfactory. It makes chunks and production rules harder to read and understand. Because the generic chunk type must be multi-purpose, it contains too many slots. Apart from aesthetic reasons, large chunk-types have two serious disadvantages: spreading activation is diluted over more chunks, and collapsing two chunks with the same information is more likely to go wrong due to the fact that irrelevant bookkeeping slots have different contents.

Ideally, a model uses small production rules and chunks with only a few slots. It is almost impossible to satisfy both of these constraints at the same time in a model in which new production rules are learned. There are probably no easy solutions to this

problem, but it should be a consideration if one attempts to specify a new production compilation mechanism.

Base-level decay

The models in this thesis have used either 0.5 or 0.3 as the base-level decay parameter. A fast decay of 0.5, which is the official recommended value of the parameter, turns out to work best for decay within an experimental session. A slow decay of 0.3 is necessary for experiments in which an hour, a day or a week passes between experimental sessions, else ACT-R will forget all it has learned. This problem has also been noted by Anderson, Fincham and Douglass (submitted). A decay of 0.25 is even necessary to account for some of their data. Since base-level decay is a parameter that is supposed to remain constant, this poses a problem that has to be resolved. A possible solution, explored by Anderson et al., is to change the decay function, so that it decays fast at the start, but more slowly as time progresses. Another possible solution is to suppose that base-level decay is slow all the time, but that the apparently high decay during an experiment is due to interference. This interference, for which association strength learning in ACT-R can account in principle, may also be the key to resolving this issue.

Production-strength learning

The learning mechanism I haven't used in any of the models in this thesis is production-strength learning. Strength is a parameter maintained with each production rule, reflecting its past use in the same manner as base-level activation for chunks. Strength is a parameter in the equation that determines the time it takes to retrieve a chunk (equation 2.3). Since strength influences the retrieval time of chunks, production rules that do not retrieve chunks other than the goal are not affected at all by strength. A second problem is that the strength of the production and the activation of the chunk are added together in the retrieval time equation. As a consequence, if the strength of a production rule is high enough, it can retrieve almost any chunk in almost zero time.

Why would ACT-R need production-strength learning? Generally, strength learning is used in ACT-R models to account for the fact that there is a speed-up in performance on new tasks. These models often assume that participants have already learned the necessary task-specific production rules at the start of an experiment. Using these rules improves their speed. An alternative account would be along the lines of the scheduling model: at the start of an experiment, most task-specific knowledge is still declarative. This declarative knowledge is only gradually compiled into production rules, providing the speed-up normally explained by strength learning.

8: Concluding remarks

Assessing model fits

At several points in the thesis, I have criticized the R^2 measure as a method to assess the quality of fit between the data and the model. This measure is not sensitive to the spread of the data, and is not suitable if there are only a few data points to compare. Moreover, the number of free parameters in the model is not taken into account in the measure. So, if several models are compared with respect to the R^2 -measure, the model with the highest value is not necessarily the best model. It would be very desirable to have a method similar to the multi-level methods described in chapter 3, in which addition of a parameter to the model has to be defended by showing it provides for a significantly better fit. Of course, this problem is not specific to ACT-R, but applies to cognitive modeling in general.

A look back at Soar

One of the concerns in the research of cognitive architectures has always been: is it not possible to implement any model you want in any cognitive architecture? For example, would it not have been possible to model all the data discussed in this thesis by Soar models? This is tough to answer, since it is very hard to prove something cannot possibly exist. But let us take a very simple example, the Tulving model discussed in chapter 4. The important issue in that model is the notion of forgetting, and the fact that certain information is forgotten in a week and other information apparently not. The ACT-R model shows that this dissociation naturally follows from a rationally organized declarative memory, without the need to resort to an explanation that assumes separate memory systems for implicit and explicit memory. If one were to model this experiment in Soar, the process of forgetting would have to be part of the model. Although it is possible to come up with such a model eventually, the fact remains that the aspect of forgetting information is not part of the specific task, so should not be part of a model of that task.

Nevertheless, Soar has been a source of inspiration for many of the models in this thesis. The idea, introduced in chapter 5, of pushing a dependency as a subgoal in situations where no promising other rules apply corresponds closely to the Soar notion of pushing a subgoal in case of an impasse. The interpretation process of declarative rules, as discussed in chapter 7, also has close ties to the way Soar handles operators. There are many good ideas in the Soar architecture, and its failure to penetrate main-stream psychological research is probably due to the fact that the area in which it excels, complex problem solving, is a topic that is not as central in cognitive psychology as it should be.

8.5 *Practical implications*

In this thesis I have shown that problem solving cannot be studied properly without taking learning into account. Although this idea may not be too controversial in the domain of problem solving, many practical applications still assume that non-learning reasoning systems can be built that reason in a human-like fashion. The main applications of rule-like systems are expert systems, human-computer interaction and education.

Application in the domain of expert systems

The assumption of expert-system design is that it is possible to specify all the relevant task-specific knowledge for a certain task. This may be true in the case of simple tasks, but not of all tasks in general. It is impossible to make a non-learning expert system for scheduling. For tractable problems, one might also wonder whether the expert-knowledge approach is the best. Since it is estimated that the number of rules an expert has on a certain domain is around 50.000, it is highly impractical to try to specify all of them. Even if it is possible to specify all these rules, the subsymbolic knowledge associated with these rules also has to be defined. This subsymbolic information is crucial in finding the right information at the right moment. Expert problem-solving behavior is probably not the invocation of stored knowledge, but an active process of constructing new knowledge for the current purpose. Apart from this explicit learning aspect of expert behavior, implicit learning, by means of the ACT-R learning mechanisms, keeps organizing the subsymbolic aspects of the knowledge.

A promising alternative method for constructing expert systems is to use the skill-learning theory presented in this thesis. Knowledge the system has to use can be supplied in a declarative fashion, after which the system is submitted to a training program. As a result, the system will organize the knowledge in the most profitable way, either as production rules, or as examples, or it will forget knowledge that does not prove to be useful altogether. Some issues have to be resolved before this can be a viable method: the right set of learning strategies has to be found, and a set of generally useful declarative rules.

Application in the domain of cognitive ergonomics

The same point made about expert-system design can also be made with respect to task analysis for the purpose of interface design. For difficult tasks it will be impossible to make a complete task-analysis. Task analysis has another drawback: since it always investigates the knowledge of an expert user, it can only make approximate predictions about novice users. Instead of trying to identify all the knowledge an expert user has, a model can be supplied with the instructions a

8: Concluding remarks

novice user gets, and be submitted to the same training program novices have to go through. Again, the skill learning theory, when properly extended, can be very useful for this purpose.

The theory also allows for the study of the integration of task-specific knowledge with knowledge about other tasks (the assimilation paradox, see Mulder, Lamain & Passchier, 1992). A general guideline in interface design is that the interface should help the user to get an adequate mental model of the system (Norman, 1988). Although this guideline is considered very useful, the notion of a mental model is rather vague. Neither is it clear when a mental model is adequate enough, and to what kinds of mistakes a certain mental model may lead. These kinds of questions can all be studied in the proposed modeling framework of skill acquisition. The notion of explicit learning of declarative rules is closely related to the concept of a mental model. Both are consciously inspectable knowledge structures that can be used in an interpretive fashion to make decisions about what actions to take. When used within the skill-learning framework, it is possible to make predictions about how knowledge from a mental model is proceduralized.

Application in the domain of education

The idea that cognitive development involves explicit learning strategies can also have implications for education. The goal of education is not just to teach children specific skills, but also to teach children how to approach problems in general. This latter goal can only be achieved indirectly, since general strategies can only develop by learning specific skills. But if we know more about this process of strategy learning, we might be able to select a set of skills to teach that is optimal for general strategy development. This is not only applicable to children. One of the main goals of university education is to teach “an academic way of thinking”, although this is not taught in any particular individual course.

8.6 A Unified Theory of Learning?

In this final chapter I have outlined a theory of skill learning. This theory uses existing theories of learning, glued together by elements inspired on the principle of rational analysis. This theory is supported by several models discussed during the course of this thesis. These ingredients may eventually be parts of a unified theory of learning, which is itself a piece of the puzzle for a unified theory of cognition. In order to specify such a unified theory of learning, a simulation environment is needed that implements it. It might take the form of an extension to ACT-R, and be capable of learning its own task-specific knowledge from instructions. This implementation would strengthen the theory, and enable many new predictions and applications.