2

The Expert Module

John R. Anderson
Carnegie-Mellon University

Intelligent tutoring systems, by their name, are supposed to bring intelligence in some way to the task of computer-based instruction. There are two key places for intelligence in an ITS. One is in the knowledge the system has of its subject domain. The second is in the principles by which it tutors and in the methods by which it applies these principles. Clearly, human tutors are effective only when they possess both kinds of intelligence; lack of either component leads to instructional ineffectiveness. Humans cannot tutor effectively in a domain in which they are not expert, and there are also inarticulate experts who make terrible instructors.

The focus of this chapter is the expert module of a tutor that provides the domain intelligence. In my view, this is the backbone of any ITS. A powerful instructional system cannot exist without a powerful body of domain knowledge. Frequently and perhaps typically, the expert modules in ITSs are incomplete, and as a consequence, they can provide only part of the instruction required in the domain. All existing ITSs need to be supplemented by human teachers. So, for instance, Steamer (Hollan, Hutchins, & Weitzman, 1984), which is used to train engineers about steam propulsion plants, knows a great deal about the mathematical properties of steam but rather little about how to operate a steam plant. As a consequence, Steamer provides only part of the instruction necessary to operate such plants. Nonetheless, it is judged to provide an important component of the instruction.

A powerful expert module must have an abundance of knowledge.

This is certainly the lesson from the expert systems work in artificial intelligence. It is also the lesson from the study of human expertise, where experts are invariably people with many years of experience. Hayes (1985) investigated what it takes to achieve levels of performance commonly ascribed to geniuses in areas ranging from mathematics to music. He determined that no genius produced a truly exceptional work without at least 10 years of experience. Presumably, these 10 years of experience were required for enough knowledge to accumulate to permit the exceptional performance.

It should be emphasized that a great deal of effort needs to be expended to discover and codify the domain knowledge. The sheer amount of knowledge required in most complex domains ensures that developing the expert module will always be labor-intensive. As techniques of intelligent tutoring evolve, authoring systems might be expected to assume much of the work involved in tutoring. However, authoring systems will never do the work of discovering and codifying the domain knowledge. Already, we estimate in our applications to programming and mathematics that over 50% of our effort goes into encoding the domain knowledge. This proportion will only increase as other components become more automated.

Having decided that we need to encode into the system a large body of knowledge, we must confront the problem of how to encode that knowledge. There are basically three options. The first is to try to find some means of computing that knowledge that does not require our actually codifying the knowledge that underlies human intelligence. For instance, a system can use mathematical equation-solving, which produces through numerical processes what humans achieve through symbolic processes. SOPHIE (Brown, Burton, & deKleer, 1982), for example, used the SPICE simulator for electronic circuits. This system performs its calculations by mathematical relaxation techniques. It does not have human knowledge of electronic currents, but it can still reason about them by simulating them with its mathematical model.

The second possibility is basically to go through the standard stages of developing an expert system.. This involves extracting knowledge from a human expert and devising a way of codifying and applying that knowledge. Although the knowledge comes from a human, the way it is applied does not have to correspond to the way the human expert applies it.

The third possibility is to go one step further and make the expert module a simulation, at some level of abstraction, of the way the human uses the knowledge. This is clearly the most demanding approach to developing an expert module, but I will argue that

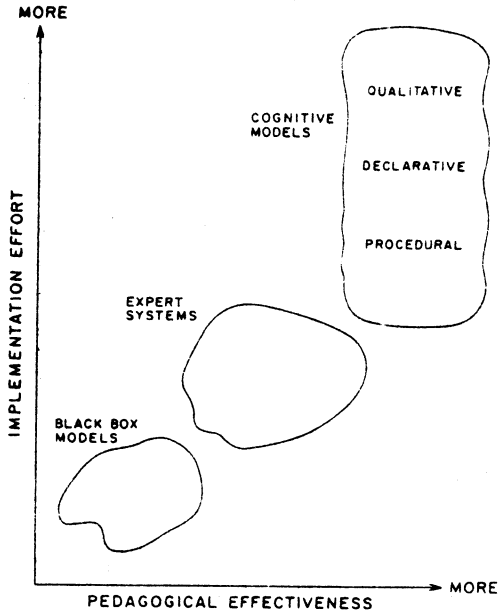


Figure 2.1 The tradeoff between the pedagogical effectiveness of an expert module and the effort of constructing it.

experience shows this approach to be essential to producing high-performance tutoring systems.

In general, expert modules cannot be reviewed in the abstract. It is necessary to understand how they will fit into an overall tutoring system. It certainly is the case that what is easy for the expert module considered in isolation is not easy for tutoring system in total. Thus using a set of mathematical equations, although expedient, would make it extremely difficult to generate articulate instruction. Figure 2.1 illustrates the relationship I perceive between ease of development and pedagogical effectiveness.

In what follows, the three approaches to the expert module will be reviewed, giving the greatest emphasis to the cognitive modeling approach, which lends itself most easily to powerful tutoring methods.

RELATIONSHIP OF EXPERT MODULES TO EXPERT SYSTEMS

Before analyzing the different types of expert modules, it is worthwhile to consider their relationship to the expert systems of artificial intelligence (see Hayes-Roth, Waterman, & Lenat, 1983). The first issue

is to define an expert system. There are two notions of expert systems, one that is tied to a certain methodology and a second that is criterion-based. A "knowledge-engineering" methodology has arisen for developing expert systems, and it involves deploying humanlike knowledge in nonhuman ways. When I refer to expert systems, I refer to products of this methodology. These are sometimes referred to as first-generation expert systems because they tend to be narrow and brittle. Another definition would be criterion-based: Any system that achieves high-quality performance could be classified as an expert system. Thus, because any kind of expert module in an ITS must be capable of doing a complex task proficiently, it would be considered an expert system by this criterion-based definition. In particular, cognitive models would be expert systems if they model complex, demanding problem solving. The reason I am not using the criterion-based definition is that it does not enable me to distinguish between the expert module and expert systems.

It is particularly important here to consider what have been referred to as second-generation expert systems. These systems have a more fundamental understanding of the domain and are not so narrow or brittle. One does not yet get the same practical performance from these systems, but they are often viewed as the hope of the future. Systems of this kind are discussed under the category of *qualitative process* models, a special kind of cognitive model. Qualitative process models are concerned with reasoning about the causal structure of the world. In actual fact, research in qualitative process models is only sometimes concerned with cognitive fidelity; however, the emphasis here is on research that does strive for cognitive fidelity.

Figure 2.2 illustrates some of the set relationships among the concepts we have defined so far: cognitive models, black box models, expert systems defined by methodology, expert systems defined by criterion, qualitative process models, and the expert module of an ITS. As can be seen, the criterion-based definition of an expert system is sufficiently encompassing to include everything except these cognitive models that concentrate on getting the details of some behavior correct. Black box models, methodologically defined expert systems, and cognitive models all intersect with the expert module of an ITS.

Work on expert modules could potentially increase the range of tasks that can be solved by computers. Given the criterion-based definition, fundamentally expanding the boundaries of what can be done by expert systems may be the long-range consequence of the cognitive modeling approach that I will be advocating. That is, it seems that a reasonable methodology for acquiring a working expert system is to make a running simulation of a human expert.

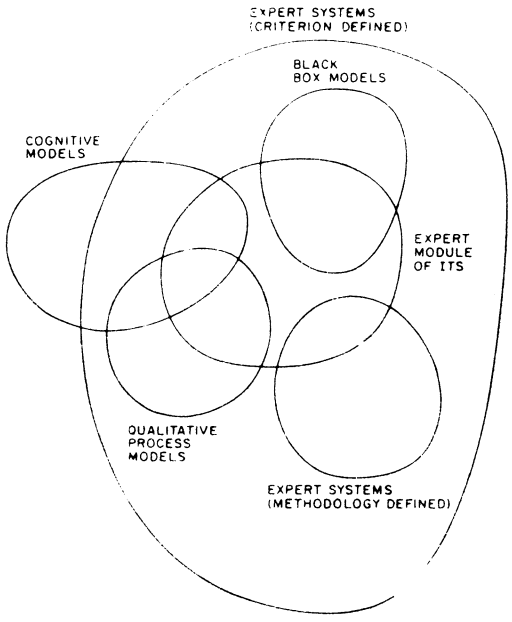


FIGURE 2.2 The set relationships among cognitive models, black box models, expert systems methodologically defined, expert systems criterion-defined, qualitative models, and the expert module of an ITS.

It must be pointed out, however, that no cognitive models to date have outperformed expert systems developed with the knowledge-engineering methodology. So far, the constraint of being true to human behavior has been more a burden than a stimulus.

By definition, intelligent tutoring systems can be built only for domains for which expert systems (criterion-defined) exist. This then poses an interesting question: Why build tutors to teach topics for which we already have expert systems to perform the task? Even if we are faced with a new domain requiring that a new expert module for our ITS be built, why not just quit at the expert module? There are three standard answers:

1. To satisfy the need for robustness. It is generally considered desirable that humans be able to perform functions that machines perform just in case these machines break down or are temporarily inaccessible. Thus, even though calculators are common, we teach our children basic arithmetic skills; and even though spelling correctors exist, it is considered valuable to know how to spell accurately. Presumably, this need for robustness is especially strong in military domains.

2. To establish prerequisite knowledge. Tutors can teach students knowledge that is often a prerequisite to learning skills that an expert system cannot acquire. Thus, we need to teach calculus problem-solving skills (Kimball, 1982) as a prerequisite to creating Ph.D. physicists. We need to teach basic LISP programming skills (Reiser, Anderson, & Farrell, 1985) as a prerequisite to using LISP for artificial intelligence programming. In effect, tutors can facilitate students' mastery of basic skills before they learn advanced skills. Because the tutor cannot teach the advanced skill, a premium is placed on having a tutor that smooths the transition from the tutoring environment to learning on one's own.

3. To teach part of a skill. A corollary of the previous answer is that tutors can sometimes teach part of a skill if not all of it. So, for instance, the Geometry Tutor (Anderson, Boyke, & Yost, 1985) can tutor the generation of proofs but only if the proofs do not require constructions, because generating proofs is much more tractable than creating arbitrary constructions. We can therefore provide tutors for part of a high school geometry course. A variation on this is that we can use the partial expertise of a system to provide partial feedback. So for proof systems that are too complicated to build into a viable expert system, we can still tell a student whether a step in a proof is logically correct although we cannot suggest the proof itself. Thus because we have the necessary expert module, logical validity can be tutored, but the same is not true for proof generation.

BLACK BOX MODELS

A black box expert is one that generates the correct input-output behavior over a range of tasks in the domain and so can be used as a judge of correctness. However, the internal computations by which it provides this behavior are either not available or are of no use in delivering instruction. The classical example of a black box model is the original work on SOPHIE (Brown & Burton, 1975). It used a general-purpose electronic simulator called SPICE II (Nagel & Pederson, 1973) and was intended to teach students how to troubleshoot faulty electronic circuits. The tutor used its simulator to determine the reasonableness of various measurements that the student would make in troubleshooting the circuit. Because the SPICE simulator worked by solving a set of equations rather than by humanlike, causal reasoning, it was not possible for SOPHIE to explain its decisions in detail. Later versions of SOPHIE (Brown, Burton, & deKleer, 1982) utilized a causal model of circuits to deal with this deficiency. I discuss this causal model under the category of qualitative process models.

One could imagine a black box expert for the game of chess that found good moves by searching over millions of sequences of chess

moves—something that human chess experts clearly do not do. Such a system could provide good advice about what move to make, but it could not explain why. A similar idea is used in the WEST program (Burton & Brown, 1982), in which a black box expert does an exhaustive search of the possible moves and determines the optimal move given a particular stragegy.

Clearly, such an expert can be used in a simple reactive tutor that tells students whether they are right or wrong and possibly what the right move would be. Quite possibly such a reactive tutor is more pedagogically effective than no tutor. The notion of a black box plus reactive tutor is interesting because it suggests a cheap way of converting off-the-shelf expert systems into tutors. Note that it is not limited to black box experts but could be used with any type of expert system (criterion definition).

However, the intelligent tutoring paradigm is based on the belief that what a tutor says is critical and that it is helpful to say more than just "right," "wrong," and "do this." The question is how to build a more articulate tutor around an expert system when knowledge of that system is not accessible. One way to build such a tutor is with a methodology dubbed *issue-based tutoring* by Burton and Brown (1982). The basic idea is to make patterns defined on the students' behavior and the experts' behavior and to attach instruction to those patterns. For instance, one issue recognizer in WEST is evoked when the expert chooses to bump and the student does not. It interrupts with an explanation of the usefulness of bumping. In WEST, the response of the issue-based recognizers not to single events but to patterns of events enables the system to respond in some fairly sophisticated ways.

Figure 2.4 illustrates the basic idea of issue-oriented tutoring based on observing the surface behavior of the expert and the student. Issue-oriented recignizers look for some configuration of the two surface behaviors that indicates that a tutorial issue is ripe for discussion. This idea of issue-based tutoring is very powerful and need not be restricted to black box modules. It is appropriate for other kinds of expert modules as well. So, for instance, in the Geometry Tutor (Anderson, Boyle, & Yost, 1985) an issue recognizer is invoked whenever the student uses an equality statement for a premise when a congruence statement is required. Attached to the issue recognizer is a dialogue that reinforces the difference between equality and congruence. Although this tutorial intervention could have been attached to the internal structure of our expert module (and some of the publications are written as if they were), it proves to be more economical and efficient to code this intervention as an issue recognizer defined on surface behavior.

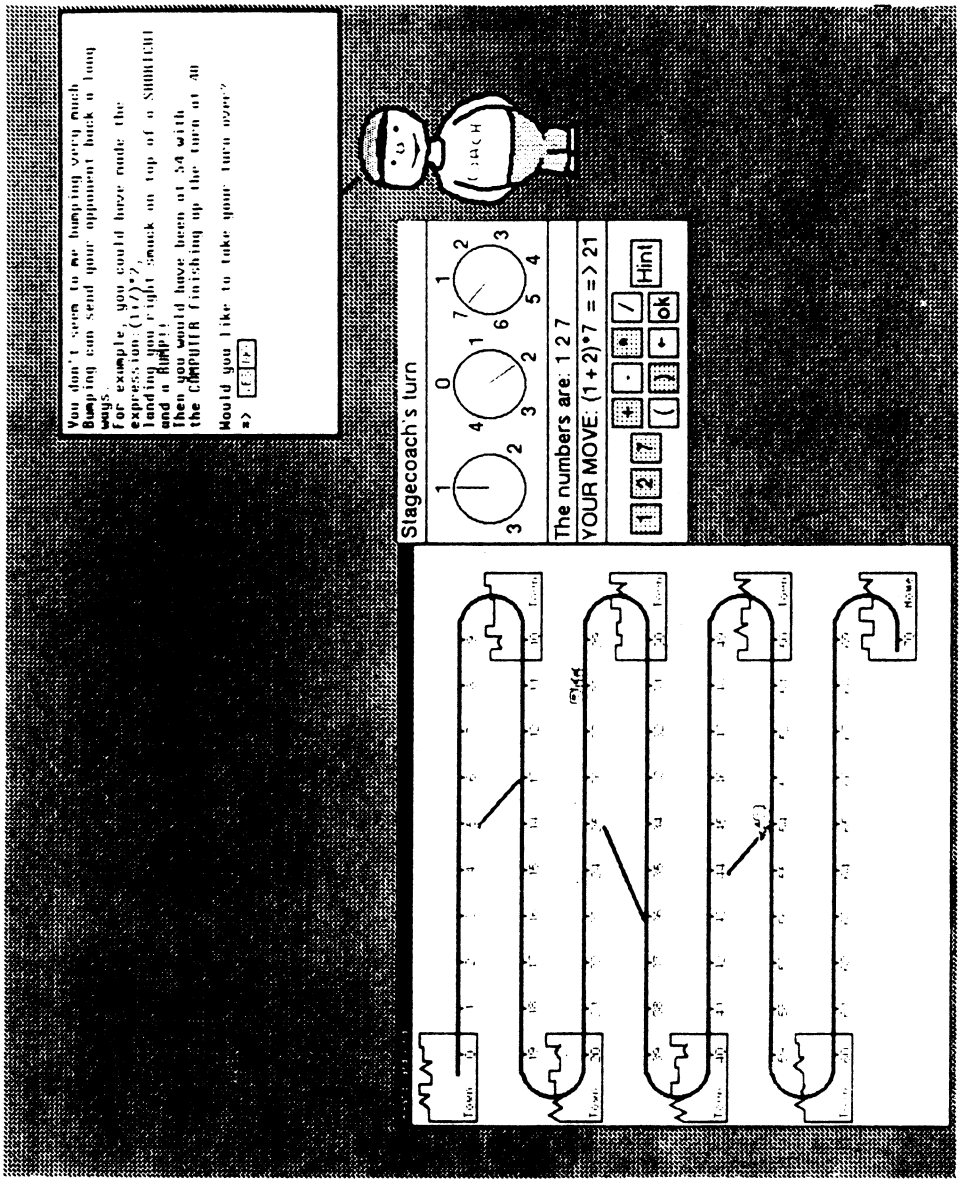


FIGURE 2.3 Tutoring of bumping in WEST.

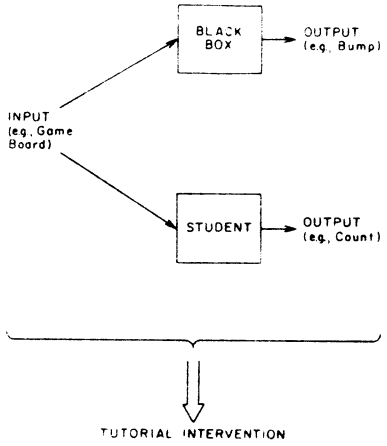


FIGURE 2.4 The pattern recognition that underlies issue-oriented tutoring.

However, as Brown and Burton recognized in their later versions of SOPHIE, there are things that cannot be tutored by such surface-level issue recognizers. Access to the internal structure of the expert is necessary for creating appropriate explanations. For instance, a standard mistake in geometry is to fail to use the reflexive rule of congruence when appropriate. (Because the reflexive rule can apply to every object in the diagram, there is a great potential for overusing it, and students appear to guard against overuse by never using the rule at all.) A tutoring system cannot explain to the student why the rule is appropriate in a particular context without access to the chain of reasoning that led the expert to conclude that the rule was appropriate.

Figure 2.5 illustrates the contrast between surface-level tutoring, which can be implemented with issue-oriented recognizers, and the kind of deep-level tutoring that can be implemented if there is access to the internal reasoning of the expert module. At the surface level we can note the legal problems with the student's response (a) and point to the correct behavior (b). However, if we model the student's error, we can explain the misconception to the student (c) and motivate the system's choice (d). Again, on the belief that explanation is helpful, deep-level tutoring should be more effective than surface-level tutoring.

GLASS BOX EXPERT SYSTEMS

A second category of expert modules is that of the expert systems that are prototypically generated in the knowledge-engineering

Surface Level versus
Deep Tutoring

Surface Level

(a) "The side-angle-side rule requires two congruent sides and a congruent angle; you have only given one congruent side and a congruent angle"

(b) "Try to prove $\overline{AB} \cong \overline{AB}$"

Deep Level

(c) "To apply the side-angle-side postulate you have to establish AB is congruent to itself. You cannot simply assume it"

(d) "Whenever you are trying to prove triangles congruent it is a good idea to prove that shared sides are congruent to themselves. This will give you a pair of corresponding parts"

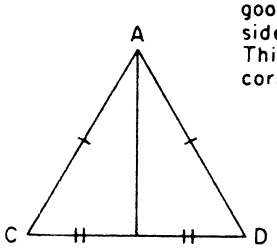


FIGURE 2.5 The contrast between surface-level tutoring and deep-level tutoring.

tradition. The basic methodology of building these expert systems involves a knowledge engineer and a domain expert who can identify a problem area and its scope, enumerate and formalize the key concepts in the domain, formulate a system to implement the knowledge, and then iteratively test and refine that system. These systems are characterized by the great quantity and humanlike nature of knowledge that is articulated. The knowledge acquisition process is recognized as the time-consuming component of building expert systems, and the one that great effort is being expended in an attempt to automate.

By the very nature of the enterprise, the expert system that emerges from this exercise is going to be more amenable to tutoring than a black box model because a major component of this expert system is an articulate, humanline representation of the knowledge underlying expertise in the comain. The expert system methodology in its variations has been very successfully used to tackle a wide range of intellectual behaviors. There are expert systems for interpretation,

```
IF
The infection which requires therapy is meningitis
Organisms were not seen in the stain of the culture
The type of infection is bacterial
The patient does not have a head injury defect
The age of the patient is between 15 and 55 years
THEN
The organisms that might have been causing the
infection are diplococus-pneumoniae(.75) and
neisseria-meningitidis(.74)
```

FIGURE 2.6 A typical MYCIN rule.

```
IF
The number of factors appearing in the domain
which need to be asked by the student is zero
The number of subgoals remaining to be determined
before the domain rule can be applied is equal to 1
THEN
Say: subgoal suggestion
Discuss the (sub)goal with the student in a
goal-directed mode
Wrap up the discussion of the domain being considered
```

FIGURE 2.7 An example of GUIDON'S TUTORIAL RULES. *Note.* From "Tutoring Rules for Guiding a Case Method Dialogue" by W. J. Clancy, 1982. *Intelligent Tutoring Systems* (p. 218). Copyright 1982 by Academic Press. Reprinted by permission.

prediction, diagnosis, design, planning, monitoring, debugging, repair, and control. Indeed, the expert system methodology is one way of incorporating tutoring expertise when the domain expert is also an expert teacher. This seems to be Stevens, Collins, and Goldin's (1982) approach, for instance, to the development of tutors.

Curiously, there have been relatively few examples of the classic expert systems' being used as the expert modules of tutors. One example might be the use of MACSYMA by Genesereth (1982), although it is questionable whether MACSYMA is really an expert system, methodologically defined. The classic and well-analyzed case is

GUIDON by Clancy (1972), which is based on MYCIN. MYCIN (Shortliffe, 1976), whose domain of expertise is the diagnosis of bacterial infections, is one of the best known of the expert systems. It consists of 450 if-then rules, such as the one in Fig. 2.6, which encode bits and pieces of the probabilistic reasoning that underlies medical diagnosis.

The basic instruction in GUIDON is driven by t-rules, which are an extension of Burton and Brown's issue-oriented recognizers. T-rules (like the issue-oriented recognizers) are defined on a differential between the expert's behavior and the student's behavior, but they are also defined on the expert's reasoning processes. An example of a t-rule is given in Fig. 2.7. Note that this rule refers to entities in the internal structure of the expert, such as rules and goals. The black box has been opened up.

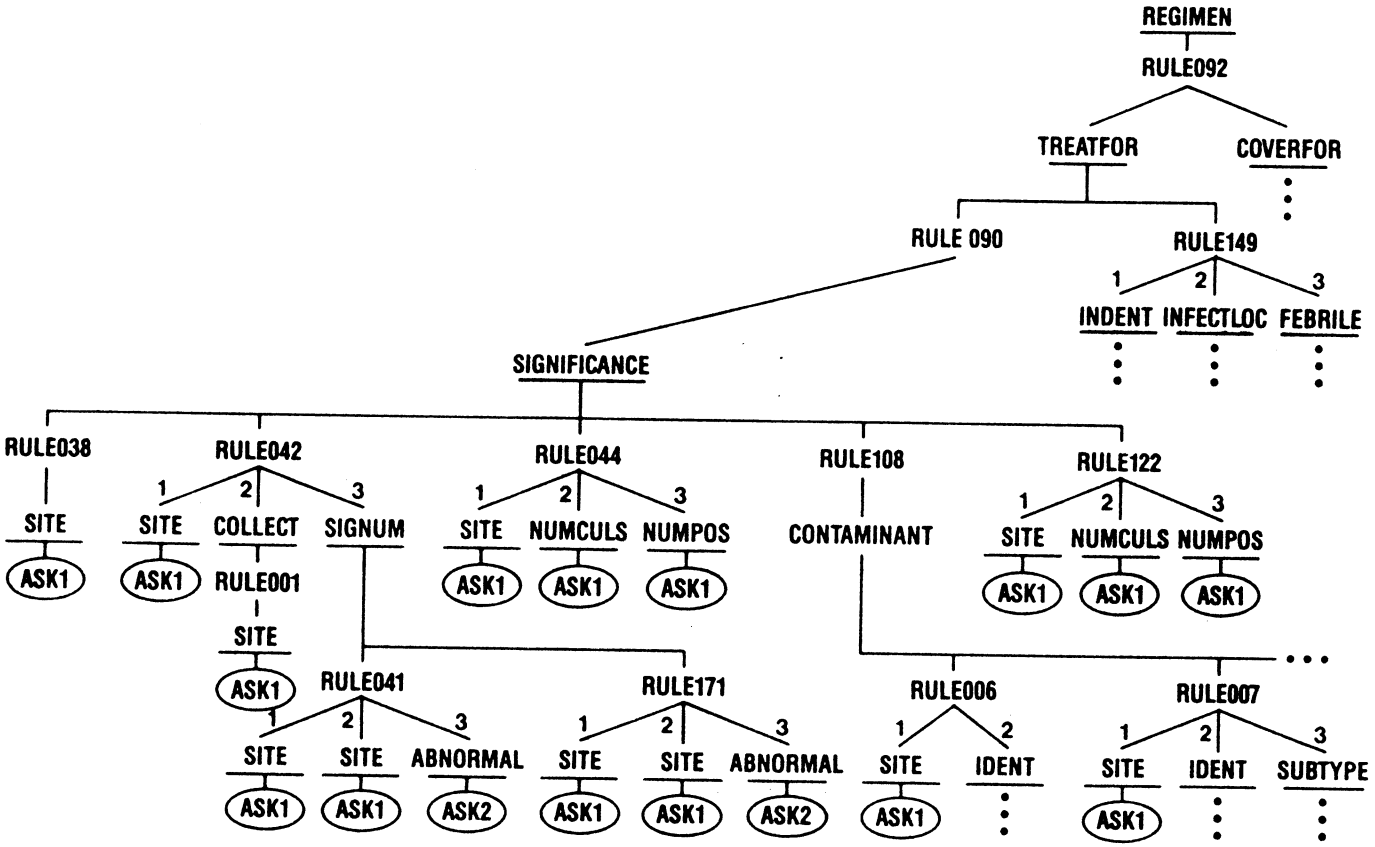
The basic instruction in GUIDON is driven by t-rules, which are an extension of Burton and Brown's issue-oriented recognizers. T-rules (like the issue-oriented recognizers) are defined on a differential between the expert's behavior and the student's behavior, but they are also defined on the expert's reasoning processes. An example of a t-rule is given in Fig. 2-7. Note that this rule refers to entities in the internal structure of the expert, such as rules and goals. The black box has been opened up.

Unfortunately, the actual reasoning process used by MYCIN to deploy its knowlege, an exhaustive backward search, is not the way the knowledge is deployed by humans. Figure 2.8 illustrates a fraction of that structure. This mismatch between the control structure of MYCIN and that of humans made an explanation of what to do next difficult. In addition, MYCIN's highly compiled rules of reasoning were difficult for GUIDON to justify. Also, many of the MYCIN rules, although appropriate for experts, were too complex to be directly taught to novices.

All of these difficulties led to the design of NEOMYCIN, in which an attempt was made to impose a different control structure on the domain knowledge. The control structure is now a domain-independent set of rules about how to use the domain rules. The currently active set of hypotheses is contained in a new data structure that is called a *differential* and that is designed to reflect some of the characteristics of human short-term memory. Also, a different data structure was used for the t-rules to facilitate explanation.

The fundamental lesson of GUIDON is that for tutoring systems to be truly effective, it is necessary to pay attention not only to the knowledge in the expert module but also to the way it is deployed. Many expert systems, although using humanlike knowledge, deploy that knowledge in the exhaustive manner so typical of computers.

FIGURE 2.8 An illustration of the backward search structure generated by MYCIN. *Note.* From B. Buchanan & E. Shoftliffe, *Rule-Based Expert Systems,* Copyright 1984, Addison-Wesley Publishing Company, Inc., Reading, MA. Figure 5-8. Reprinted with permission.



To be truly appropriate for tutoring, the expert module must deploy its knowledge according to the same restrictions as a human does. This practice leads us to the cognitive modeling approach.

Clancey's work was a watershed in the development of intelligent tutors because it illustrated that tutors were going to be seriously limited if they simply ported expert systems from artificial intelligence. Consequently, subsequent research has focused on the use of cognitive models. In many ways this research decision was a good one, but it has led to a neglect of practical issues, such as how off-the-shelf expert systems might be used. It would be comforting if there had been other projects besides Clancey's that explored extensively the use of expert systems for tutoring.

COGNITIVE MODELS

The goal of the cognitive modeling approach is to effectively develop a simulation of human problem solving in a domain in which the knowledge is decomposed into meaningful, humanlike components and deployed in a humanlike manner. The merit of this approach is that it gives us the expert module in the form that can be most easily and deeply communicated to the student. However, there are real costs in this approach. First of all, developing cognitive models is a more constrained and time-consuming task than simply developing expert systems. Fortunately, there have been dramatic improvements over the past 10 years in the ability of cognitive science to develop such models. These improvements have resulted at least in part from borrowing concepts from the expert systems work. A second difficulty is that running the computations of cognitive models can be quite computationally expensive. Fortunately, increasing computational power is diminishing this concern. Additional techniques for dealing with computational costs are addressed later in the chapter.

Another complexity is the issue of the amount of detail to be incorporated into a cognitive model. Many of the factors that are incorporated into some psychological simulations, such as the exact mechanisms of short-term memory search, seem irrelevant for tutoring. Faithfully modeling phenomena in an expert module adds an unnecessary computational burden. The question arises of which psychological components are essential for purposes of tutoring and which are not. I have argued (Anderson, in press) that tutoring systems depend on cognitive assumptions at the algorithm level and not at the implementation level. The algorithm level refers to high-level specification of mental computation that ignores issues of neural

implementation. The obvious analogy is to a program specified in a high-level programming language that does not address issues of machine implementation. The best exemplars of algorithm-level systems are the problem-solving models (e.g., Newell & Simon, 1972).

In discussing cognitive systems it is useful to distinguish between three types of knowledge that need to be tutored. There are domains like calculus problem solving where the main knowledge to be communicated is procedural, that is, knowledge about how to perform a task. There are domains like geography where the tutorial goal is to convey declarative knowledge in the form of a set of facts appropriately organized so that one can reason with them. Declarative knowledge contrasts with procedural knowledge in that it is more general and not specialized for a particular use. Third, there is causal knowledge, in the form of qualitative models, about a device that allows one to reason (in a task like troubleshooting) about the behavior of that device. I have listed these types of knowledge in the order that they are discussed. Coincidentally, the current success of our cognitive theories in dealing with these types has followed the same order. These classifications also have implications for the types of curriculum and instruction used to impart them, which is discussed by Halff (Chapter 4).

PROCEDURAL KNOWLEDGE

Our relatively advanced ability to model the procedural knowledge underlying human problem solving probably owes a lot to the importation of ideas from expert systems. Almost uniformly, the standard representational formalism has been some kind of rule-based system just as in expert systems. This rule-based approach is taken in the LISP Tutor (Reiser, Anderson, & Farrell, 1985), the Geometry Tutor (Anderson, Boyle, & Yost, 1985). Algebra (Brown, 1983), BUGGY (Brown & VanLehn, 1980; Burton, 1982), and the LEEDS modeling system (Sleeman, 1982) among others. The cominant type of rule-based system takes the form of production systems, which arguably provide good models of human problem solving (Anderson, 1983; Newell & Simon, 1972). Although there are many variations on production system models, they all involve a set of if-then rules matched to a working memory of facts. The working memory embodies some of the basic short-term memory limitations of the human. The production rules with their recogniz-act cycle capture the basic data-driven character of human cognition. One of the recent advances in production system models has been a set of ideas for modeling human

```
Constrained by both cognitive task analysis and
general theory

Sub () SatisfactionCondition: TRUE
   L1:   { } --- >                 (ColSequence RightmostTOPCell
                                     RightmostBottomCell RightmostAnswerCell)
COLSEQUENCE (TC BC AC) Satisfaction Condition: (Blank? (Next TC))
   L2:   { } --- >                 (SubCol TC BC AC)
   L3:   { } --- >                 (ColSequence (Next TC) (Next BC) Next AC))
SubCol (TC BC AC) Satisfaction Condition: (NOT (Blank? AC))
   L4:   { (Blank? BC)} --- >      (WriteAns TC AC)
   L5:   { (Less? TC BC)} --- >    (Borrow TC)
   L6:   { } --- >                 (Diff TC BC AC)
Borrow (TC) Satisfaction Condition: FALSE
   L7:   { } --- >                 (BorrowFrom (Next TC))
   L8:   { } --- >                 (Add10 TC)
BorrowFrom (TC) Satisfaction Condition: TRUE
   L9:   { (Zero? TC)} --- >       (BorrowFromZero TC)
   L10:  { } --- >                 (Decr TC)
BorrowFromZero (TC) Satisfaction Condition: FALSE
   L11:  { } --- >                 (Write9 TC)
   L12:  { } --- >                 (BorrowFrom (Next TC))
```

FIGURE 2.9 Production rule representation of the subtraction skill (Brown & VanLehn, 1980). *Note.* From "Repair Theory: A Generative Theory of Bugs in Procedural Skills," by J. S. Brown and K. VanLehn. *Cognitive Science, 4,* pp. 379-426. Reprinted with permission of Ablex Publishing Corporation.

learning within these models (Anderson, 1983; Holland, Holyoak, Nisbett, & Thagard, 1986; Laird, Rosenbloom, & Newell, 1986; Langley, 1985; VanLehn, 1983). This is an exciting potential for intelligent tutoring systems because of the prospect that the tutoring component can make its decisions by reference to the simulation of the student learning. Although this is an exciting possibility, no current tutoring systems actually use a learning simulation in this way. This is largely because these learning components are recent and tend to be very expensive computationally.

An exemplary set of procedural rules, shown in Fig. 2.9, represents the skill underlying multiple-column subtraction in the Brown and VanLehn model of subtraction skills. They make the point that the underlying knowledge is very use-specific. Although this knowledge is derived from the basic properties of addition, the actual rules are quite specific to subtraction and would not generalize to addition. Thus, for instance, we have rules about borrowing rather than rules about carrying, even though borrowing and carrying are based on the same abstract rules of arithmetic. The choice of using a procedural knowledge representation involves deciding whether such a use-specific representation of the knowledge is appropriate. It certainly is the appropriate model in the case of human subtraction skills because they have very little to do with addition skills.

The Brown and VanLehn work illustrates one use to which we

can put procedural representations. Brown and VanLehn propose that students make errors when they try to repair their procedures at the impasses created by missing production rules. By assumming that specific instances of these rules are missing, we can predict such students' errors. Extending a rule-based model to predicting errors puts an additional demand on its psychological reality. The rules in such a system now must capture the units of human knowledge because loss of the rules must correspond to human errors. If the rules were not the units of knowledge, then their loss would produce errors that are not seen in human behavior.

Their modularity is one of the major advantages of production rules for purposes of instruction: Each production rule is an independent piece of knowledge. This means that a rule can be communicated to the student independently of communicating the total problem structure in which it appears. This is not to say that production rules are context-free. Rather, they specify explicitly that part of the context that is relevant. So, for instance, if a production rule for using vertical angles in geometry makes reference to a goal of proving angles congruent, reference can be made to that feature of the problem and only that feature in explaining the rule:

> When you are trying to prove triangles congruent and they form vertical angles at one of their vertices, it is a good idea to prove these angles congruent by vertical angles. This will yield a pair of congruent corresponding angles which will help you prove the triangles congruent.

A frequent problem with earlier production rule models (Anderson, 1976; Newell, 1973) was that contextual constraints on the rules were not transparent. Rules had special tests built into their left-hand sides that constrained when they would apply; but it was difficult in looking at such rules to imagine when those tests would be satisfied. The current generation of goal-factored production systems (Anderson, 1983; Laird, Rosenbloom, & Newell, 1986) offer a substantial solution to this problem by making explicit reference in their conditions to goals that the production rules are relevant to. These goals, being structures with a well-defined semantics, facilitate the process of communicating to the student the relevant information about contextual constraints.

Another advantage of the modularity of production rules is that we can use the rules to represent the student's knowledge state. That is, the student's knowledge state can be diagnosed as a set of production rules. We can then use curriculum selection techniques, such as were pioneered with BIP (Barr, Beard, & Atkinson, 1975; Westcourt, Beard, & Gould, 1977) in which problems are selected to exercise instructional

units that the student has not mastered. In contrast to BIP, however, the problem selection can be defined in psychologically real units rather than by somewhat arbitrary topics. In recent work with the LISP Tutor (Anderson, in press), we have found that the underlying production rules seem to be learned systematically and independently of one another. Selecting problems to exercise those productions diagnosed to be weak leads to improved learning.

Model Tracing

One of the major advantages of the rule-based approach is that it makes possible the implementation of a tutoring methodology called *model tracing*. This is a technique used in WUSOR (Goldstein, 1982). In Kimball's integration tutor (Kimball, 1982), in Spade (Miller, 1982), as well as in our own Geometry and LISP Tutors (Anderson, Boyle, & Yost, 1985; Reiser, Anderson, & Farrell, 1985). In model tracing we try to place the student's surface behavior in solving a problem in correspondence with a sequence of productions that are firing in the internal student model. This correspondence then can be used to place an interpretation over the student's surface behavior. Clearly, the richness with which the student's behavior can be interpreted will map onto the richness of subsequent instruction. In our own research, which has a strong commitment to immediate feedback, the major function of such a model trace is to provide feedback on errors as close in time to the student's commission of these errors as possible. However, this is by no means the only function of model tracing, nor is it the only function for which model tracing has been used. Indeed, I would say our use of it for immediate feedback has been relatively unique.

Although it is nice to be able to interpret a student's thinking at every step through the problem solution, model tracing creates a number of demands that are quite stressful computationally. The major stress derives from the nondeterminism of the underlying student model. Typically, at each point there are a number of correct or incorrect productions that can fire. The combination of a few layers of production firings creates a space of thousands or millions of possible interpretations is naturally easier in the presence of a rich behavioral trace from a student. Ideally, if each production rule has an observable consequence, then the nondeterminism can be pared down at each cycle of the production system. Providing such a rich behavior trace creates an interesting demand on the interface design. Sometimes, however, efforts to obtain a rich behavior trace can lead to awkward

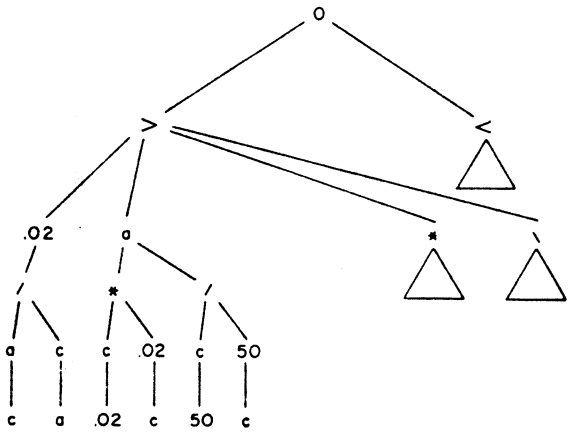


FIGURE 2.10 Some of the correct and BUGGY code sequences that a student might enter to determine whether a was less than 2% of c.

and artificial interactions. For instance, in some of our endeavours we have tried to create a trace by interrogating students about their intentions at points of ambiguity. Students report this to be an annoying and distracting feature of our tutors.

Even in the best of all possible worlds, where each production has a behavioral consequence, there are problems of ambiguity in which multiple sequences of production actions will generate the same observed sequence of student behaviors. This is a problem particularly when some of these interpretations are correct and some are in error. The tutor must either delay feedback until the ambiguity is resolved or interrupt with distracting questions. For instance, suppose we have a student who is trying to cope whether *a* is less than 2% of *c* and the student writes

$$(> (/\ a\ c)\ .02)$$

At what point can we tell the student that the choice ">" is inappropriate? Clearly, not when it is typed, because the student could be intending to reverse the arguments. As it turns out, the ambiguity is not resolved when the division sign is encountered either, because the student could have been intending

$$(> (/\ c\ 50.0)\ a).$$

The ambiguity is resolved only when the *a* is entered. Fig. 2.10 is an attempt to illustrate a small part of the problem space associated with this problem and the ambiguity in that problem space.

There are also serious problems with the efficiency of running

production systems. Despite the recent advances in the OPS family of production systems, they are still not the world's most efficient computational formalism (Forgy, 1982). A critical feature of any expert module is that it run sufficiently rapidly so that the student is not left waiting too long during its computations. One solution is to build more efficient domain-specific production systems. In our own work we have had to build such domain-specific production systems that were optimized to take advantage of special domain features.

Compiling the Expert Out

Many formalisms for expert modules, including production systems, can be very expensive in terms of time and space. This makes it difficult to deliver tutorial instruction on economically feasible machines. One way of dealing with this problem is to perform in advance all the possible computations of the expert for a particular problem and to store them in some efficiently indexed scheme on disk. This method, which we call "compiling the expert out," has been used with success in some of our applications. The cost is that they can tutor only a specific set of problems on which the expert has been run. The dynamic ability to tutor any problem the student might enter is lost. However, in some applications this trade-off may be well worthwhile.

DECLARATIVE KNOWLEDGE

Both the weaknesses and strengths of procedural knowledge representations are derived from the fact that they are use-specific. In some instances more generalized declarative knowledge may be desired. In many cases we want the students to understand the basic pprinciples and facts of a domain and how to reason with these generally, but are not concerned that the student become particularly facile at any one application of the knowledge. These are the situations that call for declarative knowledge representations.

It is not the case that the goals of procedural tutoring and declarative tutoring are mutually incompatible. We might well want a student to be both facile with the rules of a problem domain and articulate about the justifications for the rules. This seems to be the case in the domain of medical diagnosis, for instance (Clancey, 1982). Another need for declarative tutoring is illustrated in our LISP Tutor, for which we have created a special textbook (Anderson, Corbett, & Reiser, 1986) for teaching the declarative underpinnings of the

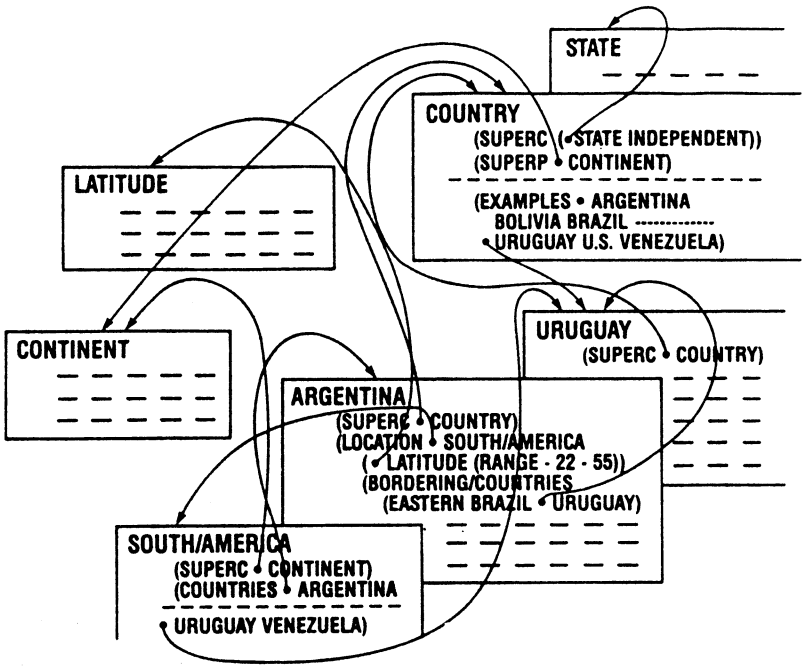


FIGURE 2.11 A portion of the semantic net in SCHOLAR. *Note.* From "AI in CAI: An AArtificial Intelligence Approach to Computer-Aided Instruction" by J. R. Carbonell, 1970, *IEEE Transactions on Man-Machine Systems, 11,* 190-202. Copyright 1970 IEEE. Reprinted by permission.

procedural knowledge the LISP Tutor teaches. It clearly would have been better to have extended the LISP Tutor to cover what is in the textbook. In fact, it is part of our general theory of knowledge acquisition (Anderson, 1983) that knowledge must start in a declarative form before becoming proceduralized.

The SCHOLAR project (Carbonell, 1970) was an early example of a project whose goal was to communicate information, in this case about South American geography. It was Carbonell's belief that the semantic net representation of the knowledge base used in this project was close to the internal knowledge structure of humans. This belief was reinforced by a fair amount of contemporary experimental work (e.g., Collins & Quillian, 1972). Figure 2.11 shows a fraction of the semantic network Carbonell was working with. It consists of nodes representing various concepts, such as countries and products, linked by various relationships, such as part-whole or generalization hierarchy. These links were used to define certain fundamental inference processes on the network. For instance, the system can

conclude that Santiago is in South America because Santiago is in Chile and Chile is in South America.

Subsequent to Carbonell's work, knowledge representations with semantic nets have become considerably more sophisticated and have evolved into frame and schema systems (Bobrow & Winograd, 1977; Brackman, 1978; Goldstein & Roberts, 1977; Minsky, 1975; Schank & Abelson, 1977; Stefik, 1980). However, the central idea has remained the same. We want hierarchical representations of knowledge structured such that flexible inference procedures on the knowledge base can be defined. Note that, in contrast to procedural representations, the knowledge base is separate from the inference procedures that are built on them. This clean distinction has been somewhat blurred by the use of "procedural attachments," in which various slots in the schema representations have procedures attached to them to define how they should be filled. But we still have a fundamental separation in a schema system between knowledge and control. This separation does not exist in procedural systems.

Carbonell's work has been continued by Collins (Collins, Warnock, & Passafulme, 1975; Stevens, Collins, & Goldin, 1982). Figure 2.12 illustrates one of the schema representations developed for evaporation, which is part of the knowledge base in the curriculum on rainfall. It is basically a schema representation consisting of various slots and fillers. In this case, there are slots for the actors in the evaporation schema, for the factors that influence the amount of evaporation, for the functional relationships among these factors, and for the result of evaporation. Bugs are created by various fallacious entries in these slots. So, for instance, many people believe that the sun is directly responsible for evaporation rather than that evaporation is a function of the temperature of the air mass and the water mass. This belief shows up as an erroneous filling in of the actor slot. Another bug involving the actor slot of this schema is what Collins calls the "small-moisture-source"—the idea that any body of water, including a small pond, is sufficient to produce rainfall.

The implicit presupposition in tutoring such knowledge bases is that the student already has the general inference procedures to be able to reason about the knowledge and that the real task is therefore to represent the knowledge in such a form that these inference procedures can be invoked. At some level this makes for a simple tutorial agenda, namely, to determine what a student has filled in at each slot and to fill in the missing information and debug the misconceptions. The major difficulty posed for tutoring systems is that declarative knowledge cannot be run the way procedural knowledge is, and so the criteria "if the student can use it he knows it" does not apply. For declarative knowledge tutors it is typical to

Evaporation

Actors
Source: Large-body-of-water
Destination: Air-mass

Factors
Temperature(Source)
Temperature(Destination)
Proximity(Source, Destination)

Functional-relationship
Positive(Temperature(Source))
Positive(Temperature(Destination))
Positive(Proximity(Source, Destination))

Result
Increase(Humidity(Destination))

FIGURE 2.12 A schema representation of some of the knowledge underlying our understanding of evaporation. *Note.* From "Misconceptions in Student Understanding" by A. Stevens, A. Collins, & S. E. Goldin, 1982, *Intelligent Tutoring Systems* (p. 16). Copyright 1982 by The Academic Press. Reprinted by permission.

fashion and interpret complex natural language dialogues. To understand these dialogues it is necessary to understand how students make inferences from their declarative data base, and a good deal of Collins's work has been devoted to just this (e.g., Collins, Warnock, Aiello, & Miller, 1975).

In contrast with tutoring procedural knowledge, tutoring declarative knowledge necessitates facing up to the full complexity of mixed initiative dialogues. The student says something that has to be clarified, which leads the tutor to ask a question, to which the student reacts with a question of his or her own, etc. One of the functions of the Socratic dialogues is to tame this process. Socratic dialogues really give the control to the tutor, who asks a series of questions to which the student is supposed to respond. In a true Socratic dialogue both instruction and knowledge assessment is supposed to be handled by asking questions. It might seem a bit unnatural to instruct by askins questions, but there are proven psychological benefits to knowledge that is generated rather than received (Bobrow & Bower, 1969). Moreover, asking questions enables the tutor to assess how well the instruction is being received. Figure 2.13 presents a piece of the Socratic dialogue with Collins's WHY system. Note that the discussion takes place with respect to specific cases. So, for instance, in Fig. 2.13 WHY does not directly ask the student whether there is a positive

WHY: Do you think the **AMAZON** jungle has heavy rainfall or light rainfall?
(Ask for a prediction about a particular case)
STUDENT: Heavy rainfall

WHY: Why does the Amazon have heavy rainfall?
(Ask for prior cause)
STUDENT: Because it's near the Atlantic

WHY: Yes, the Amazon jungle is near the equatorial Atlantic. How does that affect rainfall in the Amazon jungle?
(Ask for intermediate cause)
STUDENT: The water in the current is warm.

•
•
•
•
•

WHY: Do you think that any place with mountains has heavy rainfall?
(Form a general rule for insufficient cause)
STUDENT: Yes

WHY: Southern California has mountains. Why doesn't Southern California have heavy rainfall?
(Pick a counter-example for an insuffient cause)

FIGURE 2.13 An example of a Socratic dialogue. *Note.* From "The Goal Structure of a Socratic Tutor" by A. Stevens & A. Collins, 1977, *Proceedings of ACM 1977.* Copyright 1977 by the Association for Computing Machinery, Inc. Reprinted by permission.

IF
The student gives an explanation of one or more factors that are not sufficient
THEN
Formulate a general rule for asserting that the factors given are sufficient
Ask the student if the rule is true

FIGURE 2.14 An example of a rule for Socratic tutoring. *Note.* From "Processing in Acquiring Knowledge" by A. M. Collins, 1976, *Schooling and the Acquisition of Knowledge,* p. 343-344. Reprinted by permission.

functional relationship between the temperature of the source and evaporation. Instead, it probes the student's ability to apply this knowledge to the Amazon.

Collins formulated a set of tutoring rules for implementing the Socratic method. Figure 2.14 illustrates one that was involved in the question at the end of the sample dialogue. There are a couple of noteworthy features about such rules for Socratic tutoring. First, they have a family resemblance to the issue-based recognition rules we saw with the black box and expert models. Note, however, that the conditions of such rules refer to the underlying knowledge rather than to the surface behavior of the expert. Second, these rules involve a curious mix of knowledge assessment and instruction. The rule in Fig. 2.14 could be used to determine that the student is aware of all the factors underlying rainfall but has just not mentioned them, or it could be used to make the student aware of a new factor. Evoking this rule does not entail a commitment to the intended pedagogical outcome.

It should be clear that understanding natural language is the Achilles' heel of any effort to do such declarative tutoring. There have not been a great many of these tutors. Collins's and Carbonell's work is the only notable instance, and I think the difficulty of the natural language problem is the principle reason why. This area of intelligent tutoring is certainly waiting for fundamental progress in natural language processing.

QUALITATIVE PROCESS MODELS

A third category of expert module is concerned with the knowlege that underlies our ability to mentally simulate and reason about dynamic processes. As noted earlier, this is an important component of the ability to engage in troubleshooting behavior, which involves reasoning through the causal structure of a device to find potential trouble spots.

Models of qualitative reasoning are in a relatively immature state compared to the schema and rule-based formalisms of artificial intelligence. A number of notable research efforts are developing such models (deKleer & Brown, 1984; Forbus, 1984; Kuipers, 1984), but there is hardly an established methodology for using them. DeKleer's work on envisionment is an interesting case in point because it evolved within the context of the SOPHIE project and the need to communicate to students the causal structure of an electronic circuit.

DeKleer and Brown divide the process of envisionment into

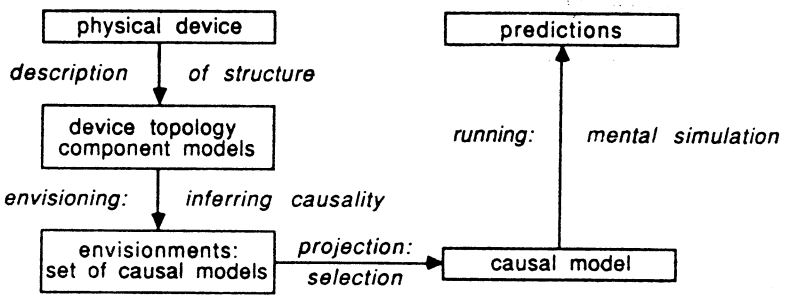


FIGURE 2.15 The development of a qualitative simulation according to deKleer & Brown. *Note.* Adapted from *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge* by Etienne Wenger, 1987, Los Alto, CA: Morgan Kaufman, Publishers, Inc. Adapted by permission.

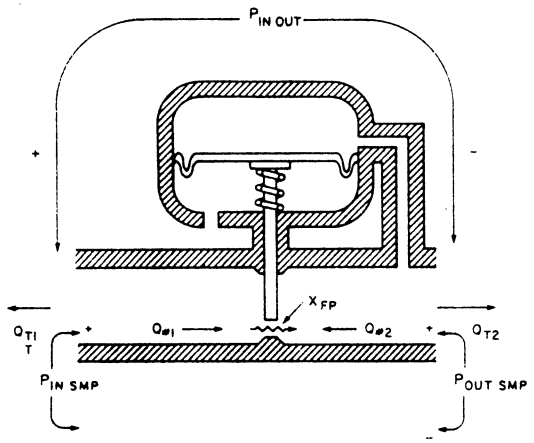


FIGURE 2.16 DeKleer & Brown's (1984) representation of a pressure regulator. *Note.* From "A Physics Based on Confluences" by J. deKleer & J. S. Brown, 1984, *AI Journal, 24*(1-3), pp. 7-83. Copyright 1984 by North Holland Publishing Co. Reprinted by permission.

constructing a causal model and then simulating the process in this causal model. Figure 2.15 illustrates their conception of this process. The causal structure of the device is inferred from its topology by examination of the local interactions among components. The assumption is that this causality can be understood locally, and it is called the "no function in structure" principle. When this principle is violated and description of a component makes reference to the functioning of the whole device, there is a danger that that component will assume the functioning of the device rather than explain it. Having

this causal model, deKleer and Brown then use a calculus to propagate the behavior of the device through these components. Much of the current work on qualitative models is concerned with various calculi for such propagations.

Figure 2.16 illustrates one of the devices, a pressure regulator, which has been a focus of deKleer and Brown's work. It consists of a set of components, such as a valve, which operate on certain local inputs. So, for instance, the valve operates so that the amount of water flowing through it varies with the pressure and the position of the valve control. This relationship is expressed by what deKleer and Brown call a confluence, which is a constraint among variables. The confluence for the valve is

$$\delta P_{in,\ out} + \delta Q_{\#1(vv)} + \delta X_{FP} = 0$$

where $\delta P_{in,\ out}$ is the change in pressure,

$\delta Q_{\#1(vv)}$ is the change in flow,

and δX_{FP} is the position of the valve control.

The entire devise is modeled by a set of such confluences. Reasoning about it involves tracing the constraints among the equations.

The psychological status of this work is quite ambiguous. As deKleer and Brown note, the no-function-in-structure principle is constantly violated in human reasoning. What they are trying to develop is more on the order of a prescriptive model of thinking. A constraint in this prescriptive model is apparently that it should be easy for humans to follow these prescriptions even if they normally do not. Such a prescriptive model is certainly appropriate as an expert module for an intelligent tutoring system.

It is not clear to me whether qualitative models really involve a category of knowledge fundamentally different from procedural and declarative knowledge. It might be argued that people have a set of declarative knowledge structures for representing the form and function of various devices and a set of procedures for reasoning about the causal interactions among these devices.

The real difference may not be in the knowledge type but in the indirectness of the knowledge so represented. The end goal in applications such as electronic troubleshooting is not to have the student correctly simulate the causal interactions in a circuit but to use that ability in service of the problem solving involved in troubleshooting. Thus, one of the issues that arises in a tutoring context is how to use the qualitative knowledge in a larger problem-solving

context. This issue has largely not been addressed in the work on qualitative reasoning.

As a consequence, how to include qualitative simulations in a tutoring paradigm has yet to be worked out. Qualitative simulations can obviously be used in all the ways a black box model like SPICE can, but this hardly justifies their development. There is the obvious potential for using them in explanations in which the tutor would tell the student how it reasoned to a particular conclusion about circuit behavior. White and Fredericksen (1986) use such models to actually define the curriculum sequence. There is also a need for more psychological study on how such process models are actually used in troubleshooting. Although I think it is clear that such models are used and that systems as deKleer and Brown's have at least a family resemblance to human qualitative reasoning, I think we know virtually nothing about how humans deploy these simulations to achieve their goals. Interestingly, there is a considerable body of negative results in getting students to bring mental models to tasks such as troubleshooting (Rouse & Morris, 1985).

The other possibility for qualitative models is to generate articulate simulations of a particular system, such as in the Steamer project or in SOPHIE. The simulation can illustrate the qualitative transformations assumed in the qualitative simulation. The assumption is that there is a pedagogical benefit to illustrating a process in the same terms as a student should use in reasoning about it.

BASIC RESEARCH ISSUES

Although there has certainly been dramatic progress in our understanding of how to build the expert module for a tutoring system, we need a great deal more basic research before construction of expert modules can progress as an engineering enterprise. As we saw in the work on expert systems, there are real limitations in using work from artificial intelligence, which has progressed without concern for cognitive fidelity. We still need to deepen our understanding of human cognitive processes and how they can be modeled. For instance, theories of learning, in contrast to theories of performance, have yet to be integrated into tutoring systems. The range of tasks for which accurate student models can be reasonably produced is relatively narrow and consists of tasks that are algorithmically tractable and that do not involve a great deal of general world knowledge. A prime example is calculus. To understand human expertise more generally will involve a great deal more empirical and simulation research.

Also, our understanding of the learning processes by which knowledge is acquired is still quite primitive. Evidence of this is the fact that no tutoring system actively uses a learning model in its computations. Any pedagogy needs to be rigorously founded in a theory of learning. Obviously, the cognitive science efforts in learning (Anderson, 1983; Holland, et al., 1986; Laird, Rosenbloom, & Newell, 1986; Langley, 1985; VanLehn, 1983) are prime candidates for support. Related to issues of learning are the issues of the origins of bugs. As is illustrated in the work on BUGGY, the representation of knowledge can be closely connected to possible bugs. Currently, most tutor builders have to invest large amounts of time building up bug catalogs. It would accelerate the development of tutors if we had a theory or theories of the origin of bugs.

There seems to be little point in supporting work in artificial intelligence, which is not cognitively motivated, if we want to further the goal of developing intelligent tutoring systems. There are two domains in artificial intelligence that are exceptions, however: qualitative process models and natural language processing for tutorial dialogues. Our need for mechanisms in these fields is so great that insisting on cognitive fidelity in the artificial intelligence system would be premature.

Development of the expert module is not independent of the rest of the tutoring system in which it resides. Much of my discussion of the expert module has been concerned with its implications for other components of a tutoring system. Although there is need for research on models of human expertise in the abstract, there is also need for research on how such modules will fit into an overall tutoring architecture. We have seen that various types of modules tend to be linked to various styles of tutoring—black box models with issue-based tutoring, cognitive rule systems with model tracing, and declarative systems with Socratic tutoring. There is room for expanding our catalog of architectures and their relationships to expert modules. We also need to explore how the design of an interface can change the nature of the expert module. To take a simple example, the advent of structure-based editors has eliminated the need for programming tutors to be concerned about teaching syntax.

Finally, we need a meta-theory of the expert formalisms we are using and of how they can be taught. Right now the development of expert modules is the domain of a few cognitive scientists even more select than the builders of expert systems. We need to develop methods for teaching the use of cognitive science formalisms to curriculum developers. Not only is this an important practical goal, but in pursuing it I think we will come to a deeper understanding of the nature of a cognitive theory.

I think we are in a position to develop an authoring environment around expert system formalisms such as production systems or schema systems. We could develop a set of tools and instructional materials that would make it esy for curriculum developers to use these systems. The first steps toward tutoring systems that teach students how to program with production systems already exist (Zhang, 1986). The facilities for actually delivering the tutoring could be made a prepackaged part of the authoring environment. All the curriculum designers would have to do is develop the expert module, which, of course, is currently half the job of developing an intelligent tutoring system. However, delivery of the tutoring could at least be automated, and the expertise for developing the expert module could be more widely distributed.

NEAR-TERM GOALS

Relatively little activity is currently occurring in intelligent tutoring that does not have the status of a basic research project whose goal is to get more basic knowledge rather than to actually build useful intelligent tutoring systems. However, the point has been reached where a few applications are feasible, and it might be worthwhile to pursue some of them both for the relatively immediate benefit and for some sense of how the engineering of these projects will progress.

In my view the one area in which we might develop reasonably good cognitive models that could be made part of intelligent tutors is that of rule-based systems for algorithmically tractable domains. These domains include mathematics at the high school or junior college level, basic sciences like physics, basic electricity and electronics, some engineering and statistics, introductory programming, and use of various packages like LOTUS 1-2-3. This is not to say that development in these domains will be cheap. It will probably take hundreds of hours just to analyze and codify the expert module for each hour of instruction, let alone build a full tutor. However, such time frames are at least within the same order of magnitude as those that go into building conventional educational software.

Another area that may yield some short-term payoff is use of off-the-shelf expert modules either developed as black boxes or developed out of the knowledge-engineeringg tradition of artificial intelligence. This tactic circumvents the hundreds of hours that go into building the expert module. As we have seen, issue-oriented methodology shows some potential for utilizing these tutors. Basic researchers have been somewhat reluctant to follow up these issue-oriented methods because

of their perceived limitations. Researchers have been moving to expert modules with greater cognitive fidelity, and even if they continue to use an issue-oriented methodology, they use a methodology appropriate for such modules. There may be a great practical payoff to seeing how to develop methods for use with the available expert systems. It would also be in the interest of the Air Force to identify and sponsor some project of particular interest to the military. Besides possibly delivering an actual system, this effort would uncover the issues specific to military applications. I can only guess where the needs of the military are, but I would think electronics and electricity instructors in service of maintenance would be a prime candidate. A fair amount of work has already been done in this field, although of a rather theoretical variety. It would be profitable to see what would happen if we made the practical compromises necessary to see an intelligent tutorial system in an actual classroom.

REFERENCES

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

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

Anderson, J. R. Analysis of student performance with the LISP tutor. In N. Fredericksen, R. Glaser, A. Lesgold, & M. Shafto (Eds.), *Diagnostic monitoring of skill and knowledge acquisition.* Hillsdale, NJ: Lawrence Erlbaum Associates, in press.

Anderson, J. R. (in press). Methodologies for studying human knowledge. *Behavioral and Brain Sciences.*

Anderson, J. R., Boyle, C. F., & Yost, G. (1985). The geometry tutor. In A. Joshif (Ed.), *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 1-7). Los Altos, CA: Morgan Kaufman.

Anderson, J. R., Corbett, A. T., & Reiser, B. J. (1986). *Essential LISP.* Reading, MA: Addison-Wesley.

Barr, A., Beard, M., & Atkinson, R. C. (1975). Information networks for CAI curriculum. In O. Lecareme & R. Lewis (Eds.), *Computers in education* (pp. 477-482). Amsterdam: North Holland.

Bobrow, D. G., & Winograd, T. (1977). An overview of KRL: A knowledge representation language. *Cognitive Science, 1,* 3-46.

Bobrow, S., & Bower, G. H. (1969). Comprehension and recall of sentences. *Journal of Experimental Psychology, 80,* 455-461.

Brackman, R. J. (1978). *A structural paradigm for representing knowledge* (Tech. Rep. 3605). Cambridge, MA: Bolt, Beranek, & Newman, Inc.

Brown, J. S. (1983). *Process versus product: A perspective on tools for communal and informal electronic learning* (Tech. Rep.), In *Report from the learning lab: Education in the electronic age.* Educational Broadcasting Corporation.

Brown, J. S., & Burton, R. R. (1975). Multiple representation of knowledge for tutorial reasoning. In D. Bobrow & A. Collins (Eds.), *Representation and understanding: Studies in cognitive science* (pp. 311-349). New York: Academic Press.

Brown, J. S., Burton, R. R., & deKleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 227-282). New York: Academic Press.

Brown, J. S., & VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science, 4,* 379-426.

Buchanan, B., & Shortliffe, E. (1984). *Rule-Based Expert Systems.* Reading, MA: Addison-Wesley.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 157-183). New York: Academic Press.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 79-98). New York: Academic Press.

Carbonell, J. R. (1970). AI in CAI: An artificial intelligence approach to computer-aided instruction. *IEEE Transactions on Man-Machine Systems, 11,* 190-202.

Clancey, W. J. (1982). Tutoring rules for guiding a case method dialogue. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 201-225). New York: Academic Press.

Collins, A. M. (1976). Processes in acquiring knowledge. In R. C. Anderson, R. Spiro, & W. E. Montague (Eds.), *Schooling and the acquisition of knowledge* (pp. 339-363). Hillsdale, NJ: Lawrence Erlbaum Associates.

Collins, A. M. & Quillian, M. R. (1972). Experiments on semantic memory and language comprehension. In L. Gregg (Ed.), *Cognition in learning and memory* (pp. 117-137). New York: Wiley.

Collins, A. M., Warnock, E. H., Aiello, N., & Miller, M. L. (1975). Reasoning from incomplete knowledge. In D. G. bobrow & A. M. Collins (Eds.), *Representation and understanding* (pp. 383-415). New York: Academic Press.

Collins, A., Warnock, E., & Passafulme, J. (1975). Analysis and synthesis of tutorial dialogues. In G. Bower (Ed.), *The psychology of learning and motivation* (Vol. 9, pp. 49-87). New York: Academic Press.

deKleer, J., & Brown, J. S. (1983). Assumptions and ambiguities in mechanistic mental models. In D. Gentner & A. Stevens (Eds.), *Mental models* (pp. 155-190). Hillsdale, NJ: Lawrence Erlbaum Associates.

deKleer, J., & Brown, J. S. (1984). A physics based on confluences. *AI Journal, 24,* 7-83.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence, 24,* 85-168.

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence, 19,* 17-37.

Genesereth, M. R. (1982). The role of plans in intelligent teaching systems. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 137-155). New York: Academic Press.

Goldstein, I. (1982). The genetic graph: A representation for the evolution of procedural knowledge. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 51-77). New York: Academic Press.

Goldstein, I. P., & Roberts, R. B. (1977). NUDGE, a knowledge-based scheduling program. *Proceedings of the Fifth International Joint Conference of Artificial Intelligence* (pp. 257-263).

Hayes, J. R. (1985). Three problems in teaching general skills. In S. Chipman, J. Segal, & R. Glaser (Eds.), *Thinking and learning skills* (pp. 391-406). Hillsdale, NJ: Lawrence Erlbaum Associates.

Hayes-Roth, F., Waterman, D. A., & Lenat, D. B. (1983). *Building expert systems.* Reading, MA: Addison-Wesley.

Hollan, J. D., Hutchins, E. L., & Weitzman, L. (1984). Steamer: An interactive inspectable simulation-based training system. *AI Magazine, 5*, 15-27.

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

Kimball, R. (1982). A self-improving tutor for symbolic integration. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 283-307). New York: Academic Press.

Kuipers, B. (1984). Commonsense reasoning about causality: Deriving behavior from structure.. *Artificial Intelligence, 24*, 169-203.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning 1*, 11-46.

Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science, 9*, 27-260.

Miller, M. L. (1982). A structured planning and debugging environment for elementary programming. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 119-135). New York: Academic Press.

Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision* (pp. 211-277). New York: McGraw-Hill.

Nagel, L. W., & Pederson, D. O. (1973). Simulation program with integrated circuit emphasis. *Proceedings of the Sixteenth Midwest Symposium on Circuit Theory* (vol , pp. -).

Newell, A. (1973). Production systems: Models of control structures. In W. G. Chase (Ed.), *Visual information processing* (pp. 463-526). New York: Academic Press.

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

Reiser, B. J., Anderson, J. R., & Farrell, R. B. (1985). Dynamic student modeling in an intelligent tutor for LISP programming. *Proceedings of the International Joint Conference on Artificial Intelligence-85* (Vol. 1, pp. 8-14). Los Altos, CA: Morgan-Kaufman.

Rouse, W. B., & Morris, N. M. (1985). *On looking into the black box: Prospects and limits in the search for mental models* (Tech. Rep. No. 85-2). Atlanta: Georgia Institute of Technology.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Shortliffe, E. H. (1976). *Computer-based medical consultations: MYCIN.* New York: American Elsevier.

Sleeman, D. (1982). Assessing aspects of competence in basic algebra. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 185-199). New York: Academic Press.

Stefik, M. (1980). *Planning with constraints* (Tech. Rep. No. 784). Palo Alto, CA: Stanford University.

Stevens, A., & Collins, A. M. (1977). The goal structure of a Socratic tutor. (Tech. Rep. No. 3518). Cambridge, MA: Bolt, Beranek, and Newman, Inc.

Stevens, A., Collins, A. M., & Goldin, S. E. (1982). Misconceptions in students' understanding. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 13-24). New York: Academic Press.

VanLehn, K. (1983). *Felicity conditions for human skill acquisition: Validating an AI-based theory* (Tech. Rep. CIS-21). Palo Alto, CA: Xerox Parc.

Wenger, E. (1987). Artificial intelligence and tutoring systems: Computational approaches to the communication of knowledge. Los Altos, CA: Morgan Kaufman.

Westcourt, K., Beard, J., & Gould, L. (1977). Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation. *Proceedings of the Association for Computing Machinery Annual Conference.* Association for Computing Machinery (ACM-777) (pp. 234-240).

White, B. Y., & Fredericksen, J. R. (1986). Progressions of qualitative models as foundations for intelligent learning environments (BBN Report 6277). Cambridge, MA: Bolt, Beranek, and Newman, Inc.

Zhang, G. (1986). *Learning to program in OPS5.* Unpublished doctoral dissertation. Pittsburgh, PA: Carnegie Mellon University.