Ann. Rev. Comput. Sci. 1986. 1:381-450

QUALITATIVE STUDENT MODELS

William J. Clancey

Department of Computer Science, Stanford University, Stanford, California 94304

1. ORIGINS AND GOALS

Instructional programs were among the earliest applications of computer programming. The original vision remains strong today: Instruction by computer offers the potential of better attention to individual student needs and interests than can be met in the typical classroom. Individualized instruction, modeled after the idea of a private tutor, allows a student to proceed at his own pace, to explore his interests, and to receive personal, detailed evaluation and direction (Crowder 1962, Suppes 1979). Realized as an interactive computer program, such instruction might be more effective, faster, and possibly less costly than traditional teaching. In addition, computer technology provides opportunities for new forms of instruction based on interactive graphics and programming itself, which foster intuition for abstract and creative thinking (Papert 1980, Brown 1983, diSessa 1984).

The goal of this review is to provide a comprehensive, but critical review of qualitative student models. A student model is the set of records in an instructional program that describe a student's knowledge about what is being taught and allow the program to adapt its presentations to his needs. A qualitative student model describes a student's knowledge structurally, in terms of relations among concepts and a problem solving procedure. I use the concept of a qualitative model as the focus of this review in order to compare alternative computational methods and to contrast domain requirements.

1.1. *Adaptive Instruction*

Individualized instruction has also been called adaptive instruction because it emphasizes selective presentation of subject material and exercises, based on an evaluation of the student's understanding. It is important to design pro-

8756-7016/86/1115-0381$02.00

grams that do this in a general way, rather than hand-crafting specific programs for each exercise or sequence of exercises. Generality makes it possible to reduce program preparation time; to make graphics and interactive methods accessible to teachers who do not program; to construct programs that respond to a variety of student behaviors without having to tediously anticipate every specific situation that will arise; and to collect, share, and improve the good lesson plans that teachers develop.

At first, the problem of adaptive instruction was conceived in terms of "selective generation of text." Simple instructional programs are little more than "programmed texts," printing prepared material and branching on the basis of student response (Atkinson & Wilson 1969). With the development of artificial intelligence (AI) programming methods, a new kind of instructional program became possible (Carbonell 1970). By representing knowledge in networks of concepts and relations, it is possible to generate text and exercises from a representation of subject material that is separate from the teaching logic. This affords generality (so the teaching program can be used for multiple problems and even multiple problem domains) and provides a new way to formalize and study instructional practice. Many researchers believe that the computational methods under development, and the subsequent accretion of subject matter and teaching expertise, will lead to better models of problem solving and hence a better formulation of what should be taught (Brown & Goldstein 1977).

Perhaps most important, combining a knowledge network with a procedure for solving problems enables the instructional program to solve the same problems it presents to a student. The reasoning of this program can then be used to evaluate student performance and provide assistance. How this is accomplished is perhaps the most innovative and exciting aspect of the research: On the basis of the interaction with the student, a second representation of knowledge and problem-solving procedure is constructed—that is, a student model (Self 1974). Unlike a simple numeric measure of achievement, a qualitative student model is a simulation model: It can be applied to specific problem situations to predict and explain student behavior. I use the term qualitative to draw a contrast with primarily quantitative stochastic or probabilistic transition student models (Atkinson 1972, Barr & Atkinson 1975, Fletcher 1975, Kimball 1973). A qualitative student model characterizes reasoning in terms of inferential steps (individual assertions about an evolving solution) and a control or inference procedure for focusing on different parts of the problem, gathering additional information, contrasting alternative solutions, and so on (Laubsch 1975).

1.2. *Qualitative Models of Processes*

An AI-based instructional program may represent three kinds of processes qualitatively: the reasoning process, as just described; processes in the real world, constituting the subject being taught; and communication processes by which the program interacts with the student—that is, teaching knowledge.

For formal domains, such as algebra or geometry theorem proving, instruction focuses on the reasoning process itself—how to simplify equations or how to construct proofs. For physical domains, such as electronics and medicine, instruction is concerned also with understanding processes in the real world, in addition to the reasoning process. AI-based instructional programs for physical domains typically incorporate, in the knowledge network, a qualitative model of physical processes, constituting the primary subject matter of the program. For domains such as electronics and medicine, the qualitative model describes observable behavior of some system (an electric circuit, a human body) in terms of causal relations among objects and processes. A computer program itself is described, in a way similar to that used for electronic circuits, as a network of input/output relations among data objects and processes. An instructional program typically uses its inference procedure to "read" the domain qualitative model to solve problems.

Finally, it is common for AI-based systems to play an active role in probing the student's understanding and assisting him. This model of teaching, or communication processes, constitutes a third qualitative model within an instructional program.

1.3. *Components of an AI-Based Instructional Program*

Figure 1 shows three kinds of models integrated into an idealized instructional program: (*a*) the model of problem solving to be taught (commonly called the subject material, now often called the target model, idealized model or expert model); (*b*) the constructed model of the student's problem solving (student model); and (*c*) the model of communication for interacting with a student to probe his understanding and to explain or teach the target model (teaching or tutoring procedure) (Laubsch 1975).

A representation of subject matter alone could be used for instruction. For example, a model of a complex system might be presented to a student for him to inspect and experiment with. However, research has gone beyond this to develop methods for constructing a model of the student's knowledge and reasoning process. This is the essential difference between computer-based instruction, in which the computer is a passive device for presenting information, and computer-assisted instruction, in which the computer communicates with the student. Put ideally, we assume that both the student and program have goals for exchanging information and learning from each other and that they will negotiate and satisfy these goals through some linguistic process (Brady & Berwick 1983).

A student model, particularly one that describes discrepant problem-solving procedures and misconceptions, provides a new basis for adaptive instruction. By modeling the communication process, researchers are attempt-

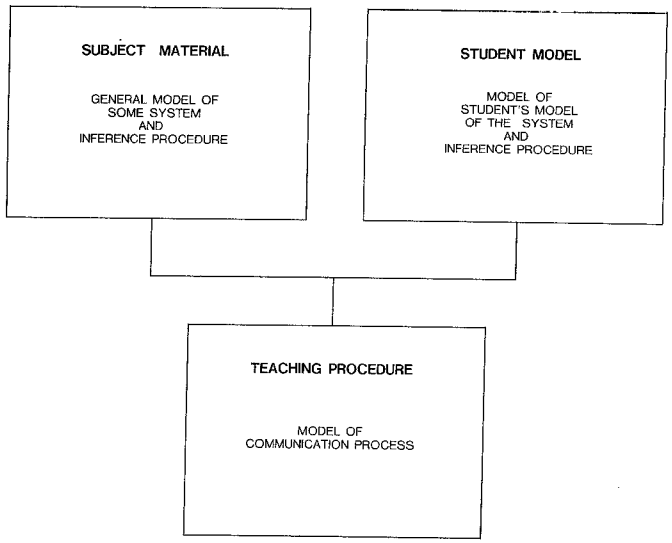

Figure 1 Three kinds of models in an instructional program.

ing to create computer programs that can share initiative with a student, probe his understanding, evaluate partial solutions, and tailor explanations to student needs. In this review, I focus on the different problems and methods for modeling a student's knowledge and reasoning process, emphasizing similarities and differences across problem domains.

Representing subject matter, the reasoning process, the student's knowledge of these, and the teaching procedure separately, in a general, reusable way, is a difficult research problem. Much progress has been made, but few programs have left the laboratory. By focusing on student modeling methods and articulating differences across domains, my intent here is to provide a framework by which researchers can relate their problems and methodologies, and practitioners can begin to apply these ideas in the schools.

1.4. *Contrast with Traditional CAI*

In the literature, AI-based instructional programs are referred to as "intelligent computer-assisted instruction" (ICAI) programs or "intelligent tutoring systems" (ITS). Unfortunately such names have a derogatory connotation, and they do not articulate what is new about the AI methodology. However, any attempt to define a clear boundary between traditional computer-assisted instruction programs and this new variety is fraught with difficulties. CAI programs have used simulation models of physical systems preciously (CERL 1977). They represent student knowledge to make predictions about behavior [state-transition models of student knowledge (Taber et al 1965)]. Some

provide a separate, primitive communication model that is reusable (Weber & Hagamen 1972). Furthermore, AI-based programs do not always have a model of student knowledge [e.g. Sophie-I (Brown et al 1976)], so we cannot even use that criterion in a restrictive definition. Instead, it appears that the most general, useful distinction is that AI-based instructional programs represent at least one component in the form of a qualitative model: some physical or computational process in the world, the reasoning process for solving problems, or the communication process for instruction. When I refer to instructional programs in this paper, I am referring to programs of this type.

1.5. *Historical Progression*

The earliest AI-based instructional programs evolved from natural language research on question-answering (Minsky 1982). At a basic level, the goals of the two areas of research are similar: to represent knowledge so that it can be selectively presented to a user, according to what he wants to know. The semantic network representation was developed for this purpose (Quillian 1968, Raphael 1982, Carbonell & Collins 1973). [Semantic networks encode factual statements about the world in terms of objects and properties, such as "Venezuela is a country" and "The language of Venezuela is Spanish." The name "semantic network" is a misnomer because, generally, what the terms in the network stand for, their meaning, is not represented (e.g. what is a language?).] In the work of Carbonell in the SCHOLAR program, the goal explicitly turns from information retrieval to instruction (Carbonell 1970). Later, especially in the work of Brown & Burton in the SOPHIE program, the interactive dialog turns from a simple question and answer format to presenting a specific problem for the student to solve. Natural language issues shift from simply interpreting student requests for information, viewed originally as a syntactic problem of parsing, to evaluating his partial problem solutions and modeling his understanding (Laubsch 1975, Burton 1976, Stevens & Collins 1977).

More recently, the widespread availability of personal workstations with bit-mapped displays has motivated researchers to reconsider the interactive format and shift the communication from words to pictures. What kinds of diagrams make it easier to understand the operation of complex machinery (Hollan et al 1984)? How can we reify the problem-solving process itself, to reveal its structure and regularities, facilitating comparison of alternative reasoning and planning procedures (Brown 1983, Richer & Clancey 1985)?

Meanwhile, the natural language discourse issues persist, in many ways paralleling the shift in that field from parsing input text to modeling the goals of the participants in some specific problem-solving situation (cf Grosz 1977). However, instructional research has tended to put the problem of modeling what the participants know and believe ahead of the problem of interpreting

and generating text. That is, what underlying knowledge representation is useful to understand what the student is doing, to represent and evaluate inadequate models, and to explain why a particular model is believable and worth using? Thus, instructional research is, at heart, research into the nature of problem solving. Instructional research has also tended to emphasize the problem of modeling a prolonged conversation, rather than single questions and statements (Clancey 1979a, Collins 1980, Woolf & McDonald 1984). Naturally, researchers have viewed communication as a process of teaching. They need to determine if the student can understand a teacher's statement. What is the student ready to learn next? Researchers have started with the premise that students already have a model of problem solving. They ask, How was the student's model formed? How do people form theories about the world? How do they relate new information to what they already know? Put in terms of natural language discourse, this research asks, What model of explanation and learning should order topic and problem selection?

In summary, the desire to "put something intelligent (and intelligible) on the screen" motivates very basic questions about how qualitative models might be represented, learned, used to solve practical problems, and communicated.

Figure 2 shows the three forces that have inspired this new research:

- **Communication media:** Computers have provided a basis for modeling processes computationally, as executable models that simulate physical and cognitive systems and that can communicate knowledge to people individually. Second-order effects include new forms of instruction and social changes in schools (diSessa 1984).
- **Modeling processes:** New computer languages provide a basis for describing knowledge and reasoning. Methods include the production-rule model (Newell & Simon 1972), causal-associational networks (Weiss et al 1978), procedural networks (Sacerdoti 1974), and structural device simulations (de Kleer 1979, Bobrow 1984). A second-order effect is the development of software tools for constructing qualitative models, often called knowledge-engineering tools (Clancey 1985a, Feigenbaum 1977).
- **Problem-solving models:** Specific problem-solving models have been developed using the computer and descriptive programming languages. The predominant theoretical base is as follows:
  - Knowledge is richly structured into patterns, called schemas, for relating situations in the world to problem-solving methods (Rumelhart & Norman 1983). This is exemplified by studies of differences between novices and experts (Chi et al 1981, Glaser 1985, Larkin et al 1980, Feltovich et al 1984, Johnson et al 1981).

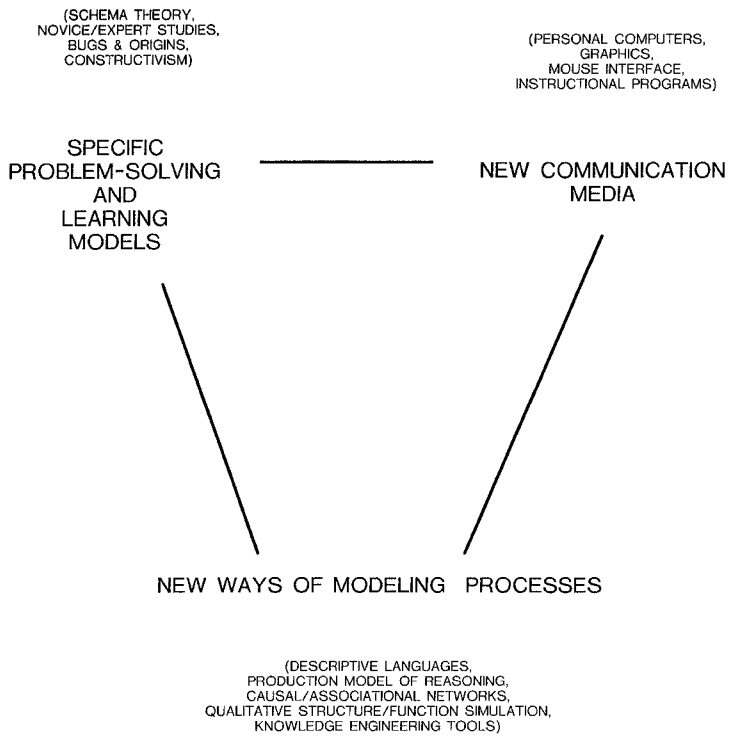

Figure 2 Three forces shaping development of instructional programs.

  - The mind actively constructs models of the world and methods for solving problems. It is not just an information repository. This is exemplified by the studies of problem-solving errors and their origins discussed in this review.

A triangle of interrelated forces shapes this research. For example, new graphics capabilities are currently leading to new ways of visualizing and formalizing reasoning processes. This graphic capability leads in turn to new models of problem solving that are incorporated in instructional programs (e.g. see Hollan et al 1984, Anderson et al 1985, Clancey 1986a).

1.6. *Range of Existing Programs*

To provide some orientation to the reader unfamiliar with the range of programs that have been developed, I outline the origin of some well-known systems. Most of the student-modeling programs cited below are summarized in Figure 6 for later reference.

First, program development is inherently empirical, driven by actual instructional needs.

- An existing environment motivates the use of an instructional agent (often called a coach) who watches a student in the normal course of some activity and offers suggestions by periodically interrupting or resolving impasses. Examples: WEST board-game coach (Burton & Brown 1979); MACSYMA ADVISOR computer program consultant (Genesereth 1982a); SOPHIE electronic troubleshooting coach (Brown et al 1974, Brown et al 1976); WUSOR Wumpus game coach (Goldstein 1977).
- An application is selected because of the suitability of qualitative methods for representing complex processes. Examples: METEOROLOGY question-answer program (Brown et al 1973); MENO-II computer programming (Soloway et al 1981); STEAMER propulsion steam plant operation (Hollan et al 1984).

Second, new efforts critically test and improve upon existing AI techniques.

- An existing computational model is adapted for use in an instructional program.
  - Knowledge representation: SCHOLAR semantic nets; GUIDON production rules (Clancey 1979b, Clancey 1982a).
  - Learning: ACT* production-rule learning model (Neves & Anderson 1981, Anderson et al 1985, Reiser et al 1985); ACM problem-space and discrimination learning model (Langley et al 1984).
  - Natural language: MENO-TUTOR discourse analysis (Woolf & McDonald 1984).
- An existing instructional program is improved (in explanation and modeling capability) by a second-generation knowledge representation (reconfiguring and extending a problem-solving model). Examples: SOPHIE-III (Brown et al 1982a); NEOMYCIN (Clancey & Letsinger 1984); WHY (Stevens & Collins 1977); PROUST (Johnson & Soloway 1984); REPAIR THEORY (VanLehn 1983a).

Other methodological characteristics of this field are discussed in Section 8.2.

2. SCOPE OF THE REVIEW

Our experience is now broad and varied. Knowledge representations in these programs include semantic, classification, and procedural networks (Charniak et al 1980). Subject areas vary from algebra to geography and from computer programming to medical diagnosis. Teaching focuses on facts, logical reasoning, understanding processes, and methods of diagnosis and design. Methods

of modeling student knowledge have proliferated, spawning a jargon that includes "overlay," "bug library," and "mal-rules."

There is much interest now in generalizing these programs. With an increasing number of psychologists and educators outside of computer science becoming involved, it is timely to step back and review what we have learned. Can we derive some methodological lessons from the variety of programs we have developed? Can we formalize these lessons in program "shells" that could be used by non-AI experts to construct their own instructional programs?

2.1. *Review Topics*

AI-based instructional programs are very complex, involving problem-solving, modeling, and discourse components; they are typically constructed by a team of researchers over many years. While this complexity makes a review like this important, it indirectly helps us because there are just a dozen or so programs to study and understand. As outlined above, these programs span a variety of problem domains and provide some basis for generalization. This review considers

- what aspects of student knowledge and reasoning, called assessments, are described by student models;
- the computational methods developed for reconstructing a student's knowledge from his problem solutions, including detection of gaps and misconceptions (commonly used to refer to both concepts and relations that a teacher believes to be incorrect);
- the different nature of student misconceptions in formal domains (mathematics, programming) and physical domains (physics, medicine), and the extent to which modeling methods for formal domains carry over to physical domains;
- the models of learning that have been developed for explaining the origin and development of misconceptions.

As indicated in the introduction, I use the idea of a qualitative model as the central concept for describing and comparing instructional programs. I proceed by describing the role such models play in instruction (Section 3); what student models, in particular, describe (Section 4); how domains differ in the nature of student errors and information available to the modeling program (Section 5); how student models are represented (Section 6); and how individual student models are constructed (Section 7).

The arguments are complex, because the model of the problem solving (cognitive) system is intricately related to the model of a physical system (such as an electronic device) or a formal system (such as a computer

program). There are important differences in problems of modeling physical and formal systems, and we must tease out these distinctions to usefully generalize the methods used by different programs.

2.2. *Important Distinctions*

Because of the disparity of domains and difficulty of describing only indirectly observable reasoning processes, the possiblity for confusion is great. In this review I draw the following distinctions:

- **Behavioral vs functional description**—I distinguish between student behavior (what the student is observed doing) and the inference procedure by which he is interpreting his observations and general model of the world. That is, I distinguish between a behavioral description of actions in the world and a functional description of beliefs and goals (what he knows and what he is trying to do). Behavioral models are domain-specific, failing to relate problem-solving behavior to general operations for focusing and achieving coherency and consistency in the constructed understanding.
- **General model vs problem-solving procedure**—I distinguish between errors of facts about the world and errors in problem-solving procedure for using these facts. In some modeling programs, the general model and problem-solving procedure are confounded, providing a questionable basis for remedial instruction. For example, descriptions of student plans do not always clearly distinguish between a planned process (e.g. a computer program) and the process by which it is planned (e.g. a programming method).
- **Formal vs physical domain**—I distinguish between the nature of facts in formal and physical domains. In particular, if facts are axioms, how does this simplify the modeling problem? What modeling methods will be inadequate for domains in which facts are beliefs based on more detailed models, as in physics? And what about the middle ground of designed artifacts, such as electronic circuits, with axiomatic functionality at the highest level but with nonformal bases in natural phenomena? Methods for modeling behavior in formal domains have not been adequately related to the requirements of physical domains.
- **Classification vs simulation model**—I distinguish types of models of processes. In general, for all the emphasis on models, highlighted by the very use of the term "student model," research often fails to emphasize explicitly that the subject matter of an instructional program constitutes a model. Specifically, in AI the term *knowledge base* has been used without explicit acknowledgment that knowledge bases contain qualitative models of processes (e.g. see Clancey 1984a). In adopting this new perspective, we shift our focus from superficial notational differences ("rules" vs "frames") and reclassify knowledge bases according to the different ways they describe processes. That is, we can begin to identify and study types of qualitative representations. Of particular interest, we can relate schema classification models (describing patterns of process causes and manifestations) to simulation models (replicating how the components of a system functionally interact). This description of knowledge bases, in terms of a qualitative modeling methodology, is especially valuable for communicating the nature of AI programming to scientists, engineers, and teachers who are familiar with numeric modeling techniques.

For each distinction summarized above, I place kinds of models on a spectrum and discuss how they are related.

2.3. *Relation to Cognitive Psychology*

Sharing the evolving view of educational psychologists—Dewey, Bruner, Piaget—the developers of today's AI-based instructional programs believe that a theory of teaching should rest on a model of the learner (Goldstein 1978, Norman 1979). Furthermore, they believe that the problem-solving models of AI may provide the basis for describing how learning takes place. Ohlsson (Ohlsson & Langley 1985) summarizes the shift in perspective: describing mental processes, rather than quantifying performance with respect to stimulus variables; describing individuals in detail, not just stating generalities; and giving psychological interpretation to qualitative data, rather than statistical treatment to numerical measurements.

An increasing number of cognitive psychologists have become involved in this research (Greeno 1980). Studying electronic troubleshooting (Kieras 1984), computer programming (Adelson 1984), physics (Chi et al 1981, Larkin et al 1980), medical diagnosis (Feltovich et al 1984, Lesgold 1983), and other areas, they are using AI concepts to describe memory organization and inference and how novices differ from experts (Glaser 1983, Chi et al 1986). However, my references to cognitive psychology in this review are limited and idiosyncratic. In particular, I do not provide a general discussion of the psychology of knowledge representation, now often called "mental models" (Gentner & Stevens 1983a, Rouse & Morris 1985). This is a review of student models in AI-based instructional programs—what they do and how they work.

2.4. *Relation to Other Areas of Artificial Intelligence*

Almost every "core" area of AI plays a part in instructional programs: knowledge representation, problem solving, natural language, and learning. The essential difference is that instructional programs must integrate these to

- understand what another problem solver is doing by watching him perform a task (or looking at the results of his reasoning);
- in particular, recognize or simulate human problem solving;
- actively and systematically articulate problem-solving methods, not just passively respond to requests for information;
- in particular, explain reasoning so that it is learned and mimicked, not just accepted as a justification;

Of these criteria, the problem of interpreting a trace of human reasoning is what most directly distinguishes this research from other areas of AI. Student-modeling programs must deal with noisy data, shifting focus of attention, and goals or reasoning strategies that may be foreign to an ideal model of the world. In this respect, instructional research is most similar to natural language research concerned with conversations about tasks (e.g. Grosz 1977, Perrault et al 1978, Appelt 1982). Such programs are often called "job aids" or "intelligent assistants." Contrasting the two areas, research on task-directed conversations has emphasized relating the surface elements of text to goals and beliefs, while instructional research has emphasized understanding errors in problem-solving knowledge. In all other respects, instructional research is generally indistinguishable from efforts to model human problem-solving, learning, and explanation of expert reasoning (e.g. Swartout 1981).

Meriting special note, recent AI studies of "models of knowledge and belief" (e.g. Konolige 1984, Moore 1982, Appelt 1982) directly address some of the issues that arise in student modeling. The difference is that instruction requires integrating problem solving, learning, and communication with belief modeling. In essence, recent research on models of belief provides a notational formalism in which the knowledge and inference models developed for instruction might be recast. The framework that I present here, unifying all work under the idea of a qualitative model, can be seen as the first step in this process.

2.5. *Special Emphases*

In finding dimensions for describing instructional programs, this review has a number of side purposes. First, I want to make clear that development of these programs is not an "application" area of computer science, a mere matter of putting well-known AI methods into practice. Quite the opposite is true. For example, student modeling research has from the onset produced new models of learning (Sleeman & Smith 1981, VanLehn 1983a) and new knowledge representations (Brown et al 1973, Brown 1977a, Brown et al 1982a, Clancey 1983a, Clancey 1983b). Explaining how a process occurs or why it makes sense, reconstructing another problem solver's solution, and accounting for

learning require knowledge representation methods that go well beyond the familiar "rules" and "schemas" of most AI programs. The problem of constructing instructional programs is broadening the meaning of AI research; it is not derivative (Stefik 1985). Nevertheless, recent AI texts do not reference research on instructional programs, even though they cite expert systems (e.g. see Charniak & McDermott 1985).

In the same vein, the theory of models and modeling emerging from this research has strong implications for how we think about "knowledge-based" programs and the nature of intelligence in general. Constructing a computer program that itself constructs a model of problem solving (the student model) requires understanding what models are and going far beyond what is required to produce an acceptable first-order problem-solving program. Indeed, the very idea that "expert systems" have anything to do with "models" is rarely mentioned in the literature (but see Soo et al 1985, Szolovits & Long 1982, Kahn et al 1985). This work has the potential of changing how AI researchers think about what they are doing.

Finally, I want to draw some sharp lines to make the methodology of our work clear, to allow new research recruits to realize that "intelligent tutoring" means more than using graphics, having a natural language front-end, or programming in "rules" or "objects."

Putting these three points together, my intent is to broaden our perspective on what computational techniques have been developed, and to derive measures of quality for evaluating new research.

3. THE ROLE OF QUALITATIVE MODELS IN INSTRUCTION

This section broadly describes the kinds of models in an instructional program and introduces the following concepts: general model, situation-specific model, inference procedure, diagnosis, and bug. This framework is then applied in the next section to classify student-modeling programs.

3.1. *What are Qualitative Models?*

It is easy to get bogged down in philosophical questions about the nature of models and theories. Terms like "explain" and "cause" have been the subject of much philosophical debate (e.g. Von Wright 1971, Achinstein 1983). In this review, I adopt some informal definitions and show how they are useful for describing what programs do.

A model is a representation, for some purpose, of some object or process (Webster 1983, Goldstein & Goldstein 1980). Scientists and engineers are familiar with the idea of numeric or quantitative models, such as Ohm's laws of electricity, economic models, mechanical engineering models of the stress

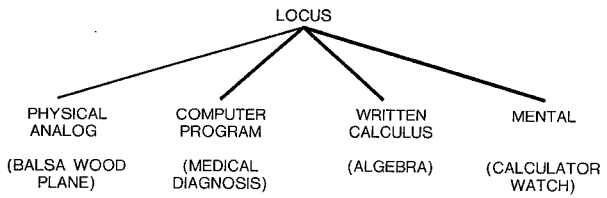

Figure 3 Where models can exist.

behavior of materials, predator-prey population correlations, and so on. Also, everyone is familiar with physical models, such as scale models of buildings used by an architect or plastic models of airplanes. Qualitative models, broadly put, are not numeric and not physical analogs; rather, they describe objects and processes in terms of spatial, temporal, and causal relations. Qualitative models may be written down in some notation [a reasoning calculus (De Kleer & Brown 1984, Sowa 1984)], believed by a person (a kind of mental model), or realized as a computer program (a kind of computational model).

Figure 3 classifies models according to locus, or where each exists. Student models, such as a model of how a student diagnoses a patient, may exist as computer programs. A student's model of the patient and how to do diagnosis is called a mental model; it exists in his mind. As a familiar example of a mental model, consider your understanding of what the buttons on a calculator watch do and your understanding of the procedure for setting the alarm (Young 1983). This model is not written down, it is "carried around in your head."

A computer program may be either quantitative, involving precise numeric calculation, or qualitative, characterizing trends and causal relationships, as mentioned above. We are concerned here with methods for representing qualitative models in computer programs. More specifically, we are interested in modeling processes, not static objects. Kinds of processes that are studied and formalized separately in order to construct instructional programs are shown in Figure 4. These include processes in the world, reasoning, learning, and communication. While learning may be construed as a form of reasoning, it is usually treated separately.

3.2. *Situation-Specific Models*

A situation-specific model is a description of some situation in the world, generally an explanation of how a situation came about or a plan for action. For example, in medicine, a situation-specific model describes a patient's current state (e.g. fever, inflammation) and the disease processes that brought this state about (e.g. a particular infection). In general, the process of solving a specific problem can be described in terms of forming a situation-specific model (Figure 5).

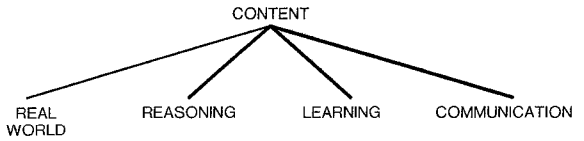

Figure 4 Processes modeled separately in studying instruction.

In this view of problem solving, a general model is related to the current situation by applying an inference procedure. The general model describes what is known about the world—for example, knowledge about stereotypic patients, diseases, and treatment plans. In some areas of AI, the term *domain model* refers to the general model; sometimes it refers to the combined general model and inference procedure.

The inference procedure is a program that focuses and orders gathering of problem information and making assertions about the solution. For example, in medicine, the inference procedure is generally called a "diagnostic strategy"; it is a procedure for gathering information about a patient and focusing on and testing disease hypotheses. Figure 6 gives common names for the general model and inference procedure in different domains.[1] The situation-specific model includes the specific problem information, transformed or reorganized in some way, depending on the nature of the task. For example, in diagnosis, the situation-specific model relates the symptoms to a description of processes that produced these symptoms. In medicine, this is called a patient-specific model (Patil 1981) (Figure 10). In geometry, the situation-specific model is the proof of the theorem, shown as a network (Anderson et al 1985) (Figure 11). In programming, the situation-specific model is the constructed program, as well as unwritten descriptions of the underlying design, relating the code to the goals the program is supposed to satisfy. In subtraction, the situation-specific model is commonly written on paper, with borrowing between columns indicated by scratch marks and the solution written below a horizontal line.

In a certain sense, the situation-specific model "copies over" part of the general model, though the form varies across domains and tasks. For example, programming-language constructs appear instantiated as particular statements in a program. In subtraction, the general fact that 5 minus 3 equals 2 appears in the situation-specific model for solving 452 minus 230. Similarly, geometry axioms and theorems appear in the proof. A medical di-

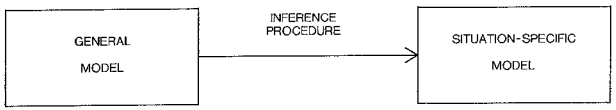

Figure 5 Problem solving: applying a general model to form a situation-specific model.

GENERAL MODEL	INFERENCE PROCEDURE	PROGRAM NAMES	REFERENCE
math table	subtraction procedure	BUGGY ACM	(Burton, 1982) (Langley, et al., 1984)
geometry axioms	rules of inference and proof procedure	GEOMETRY	(Anderson, et al., 1985)
algebra axioms	simplification procedure	PIXIE	(Sleeman, 1984b)
MACSYMA commands	programming procedure	MACSYMA ADVISOR	(Genesereth, 1982a)
PASCAL programming language	programming procedure	MENO PROUST	(Soloway, et al., 1981) (Johnson and Soloway, 1894)
electronic circuit theory and model of given circuit	circuit diagnosis procedure	SOPHIE	(Brown, et al., 1982a)
game rules and rules of probability	game playing strategy	WEST WUSOR	(Burton and Brown, 1982) (Goldstein, 1982)
physics of pressure/temperature and meterological patterns	qualitative causal reasoning	METEROLOGY WHY	(Brown, et al., 1973)
physiology, anatomy, disease processes	medical diagnostic strategy	NEOMYCIN	(Clancey and Letsinger, 1984)

Figure 6 General model and inference procedure for different problem domains.

agnosis will mention general concepts and their relations, such as the link between infection and meningitis (Figure 10).

A related but different idea is the derivation trace by which the problem was solved (VanLehn & Brown 1979). This shows how the situation-specific model is modified over time, as particular inferences are made. For example, in algebra this is the familiar sequence of transformations of an equation in the course of solving for a particular variable. Similiarly, a geometry proof, diagnostic explanation, or computer program can be written as a sequence of transformations. This is discussed in more detail in Section 5.1.

Each of the models shown in Figure 1 has a general and situation-specific form. For example, there is a general model of how to communicate (discourse processes) and a situation-specific model of the current dialog. The general model describes what typically occurs during a teaching dialog, how to detect these situations, what to do when they occur, what transitions are likely or advisable, and so on. Examples of communication processes are interrupting, assisting, orienting, explaining, evaluating, hypothesizing, probing, summarizing, and so on. In an interaction between student and teacher, both parties are using such a model to interpret and respond to each other's goals (Appelt 1982). For example, a teacher must recognize when the student is confused about what to do (perhaps because he says so) and respond appropriately.

An inference procedure, a program, "reads" the general communication model to produce a state description of the current dialog: what has been discussed with the student, what the teacher and student are currently doing, what the student is trying to understand now, etc. Given this situation-specific model, the program will further interpret the general model to decide among alternative communication processes that achieve instructional goals.

Similarly, the general model to be taught is paired with a situation-specific description of the problem now being worked on, such as a partial diagnosis of an electronic circuit. The basic job of the inference procedure is to gain additional information and make connections from the general model that improve the situation-specific model, making it coherent, consistent, and specific enough for its purpose. The student model has the same components: a general model of the subject matter, an inference procedure, and a situation-specific model describing the problem being solved. (While a recursion of models can be imagined, including the participants' models of each other's models, this level of complexity is not considered in today's instructional programs.)

3.3. *Simulation or Executable Models*

In summary, problem solving, described at a high level, involves both a general model and some inference procedure. Together, they constitute a simulation model of reasoning. When the agent is a person, a simulation model of reasoning is often called a cognitive model. It is a simulation model because it can be used to simulate the reasoning process, to solve actual problems. That is, it doesn't just describe intelligence in a static way by traits or numeric measures of aptitude. It includes a program, the inference procedure, by which problems can be solved.

The inclusion of a simulation model of reasoning in an instructional

program is of significant practical benefit. It means that the program can solve the problems that it gives to the student, providing a basis for interpreting his problem-solving behavior, evaluating his partial solutions, and assisting him.

The student model is also a simulation model, paralleling the program's domain model and inference procedure. It doesn't just describe the probability that a student will do something or not, but describes the process over time by which he gathers problem information and makes assertions. If we think about the student as some phenomenon in the world that the instructional program is trying to understand, the student model bears the same relation to the student as the general model to be taught bears to the world:

general model :: real world system (e.g. the patient)

student model :: cognitive system (the student)

The primary characteristic of a simulation model is that it can be used to predict subsequent behavior of the system being modeled. For example, a simulation of an electronic circuit can predict the internal states of its various components as well as the behavior (e.g. voltage) of its output ports.

A simulation model of a student predicts what he will do next. More importantly, it can be used to work backwards from student behavior to explain the basis of his behavior—i.e. to infer his general model and inference procedure. To highlight the contrast with a static description of the student's knowledge, interests, or preferences, a simulation student model is often called an executable model. The term is also used to describe simulation models of physical systems, such as an electronic circuit.

A classification model is an example of a qualitative model that is not a simulation model. For example, a classification model of diseases might only account for pain and other symptoms in a superficial way and might not explain in much detail how the symptoms occur. We can also use classification models to describe students and account for their reasoning. However, a simulation model allows us to construct an explanatory accounting that is complete on at least some level of detail and that relates behavior to the many specific facts in the student's domain model. Inferring errors from a constructed simulation of reasoning is also more efficient than preenumerating them in a classification. The distinction between classification and simulation qualitative models is elaborated upon in Section 6.6.

3.4. *Representation Requirements*

The clean separation between the general and the situation-specific model, and subsequent identification of the inference procedure, is an ideal that is not always explicit in computer programs. The advantages of separability and further constraints imposed by instruction are considered here.

3.4.1. THE ARTICULATE EXPERT From early on, researchers realized that instructional programs cannot be constructed on top of arbitrary problem-solving programs (Self 1974, Brown et al 1977, Brown 1977b). Expressing knowledge in a simple, uniform language facilitates its multiple use for generating questions, evaluating partially correct responses, and responding to student questions. This is one advantage of the semantic network in SCHOLAR and the rule base of MYCIN used in GUIDON (Clancey 1979b). Moreover, if the network can be interpreted to solve problems, it can be used both to evaluate a student's problem solving and to provide assistance. Brown characterized this kind of problem-solving program, which could provide the basis for evaluation and explanation, as an articulate expert. Such a program has also been called a glass-box expert because its reasoning can be inspected by students or other programs (Goldstein & Papert 1977). With further experience, we can be more precise today.

First, it must be clear from the encoding how problem information is being used to make situation-specific assertions, what the program's goals are, and how they relate to subgoals. It is precisely these characteristics that can be made evident in a production rule language and that were used to good effect in GUIDON, WUSOR, and more recently in the LISP and GEOMETRY tutors. Compared to an arbitrary computer program, the data, conclusions, and goals of these programs are well indexed. Put in a simple way, variables have a consistent meaning, corresponding to entities in the world, and there are no implicit side-effects.

Second, it is advantageous for the domain-specific qualitative model (the general model) to be separated from the inference procedure. This enables an explanation program to articulate the problem-solving strategy and domain model separately, and provides the basis for reconstructing a student's domain model (Section 7).

Third, after separating the domain model and inference procedure, for further explanation and student modeling capabilities, it is useful to represent (*a*) why the domain model is believed; and (*b*) what efficiency considerations, assumptions about data-gathering, memory, and problem situations, lie behind the design of the inference procedure. Both questions focus on the origin of knowledge and reasoning and ask about support for beliefs and constraints that are satisfied by the inference procedure.

The discussion in Section 6 explains in more detail the importance of making these distinctions and gives an example of such an analysis.

3.4.2. BEYOND TODAY'S EXPERT SYSTEMS Many readers of this review are familiar with the idea of an expert system, a computer program for reasoning about complex tasks such as system design, assembly, diagnosis, and control (Hayes-Roth et al 1983). These programs use qualitative models

for representing world knowledge and the inference procedure, with varying degrees of separation and generality of these components. What requirements does teaching place on a model of problem solving, beyond that it must adequately solve problems? These requirements are reconsidered in the concluding section.

- **Problem solving:** Solve problems in multiple ways that allow syntactic variations in situation-specific models. Cope with, and to some degree explain, discrepant behavior. The program cannot be an arbitrary model of intelligence; it must reflect human ways of thinking. Unlike other areas of AI, instructional research is unabashedly psychological.
- **Explanation:** Go beyond "audit trail" statements of how the problem is solved. Articulate the general and specific model, why both are believed, and relate them to the student's models and underlying beliefs. Address the listener's expectations, persuade him to modify his model, and help him debug it. In particular, evaluate the student's explanations of how he solves a problem. Again, the research emphasizes that explanation is an act of teaching. Explanation is not just saying what you did when you solved a problem but also relating it to what the listener expected you to do and what he would have done himself.
- **Learning:** Explain a student's beliefs in terms of how he learned from experience ["developmental epistemology" (Piaget 1971, Goldstein 1982)] and predict what he is ready to learn next. Again, this is not arbitrary "machine learning"; it must be psychological, a model of how people learn.

Summarizing broadly, instruction requires being able to use models in multiple ways—in solving problems, in explaining, and in learning from experience. Early research indicated that using knowledge in multiple ways is benefited by meta-knowledge: knowledge of the extent of what the program knows, its inference methods, representation, and reasons for failure (Davis et al 1982, Barr 1979, Barr et al 1979, Schank 1981, Kolodner 1982, Kolodner & Simpson 1984). This again motivates the separation of the general model and inference procedure. The instructional program must be able not only to say what it did but also to reflect on the regularities in its behavior, articulate them, reason about their validity, and determine what alternatives are possible.

A complicating factor is that human learning involves not just incorporating new facts but also becoming more efficient and faster through practice (Anderson et al 1981, Laird et al 1984). Some researchers believe that experts lose meta-knowledge as they gain the ability to solve problems automatically, without conscious thought. In this respect, expert systems often reflect the expert's automatic way of thinking, without the articulation and abstraction

(separation of model and inference procedure and attendant generality) useful for teaching. This makes it more difficult to construct instructional programs with the capabilities listed above.

Models of problem solving useful for teaching are thus inherently empirical (they must relate to what students do) and open (they must account for original, unexpected student behavior). Recognizing and evaluating alternative models requires that knowledge be expressed at a high level of abstraction, as general theories, in sharp contrast with specially engineered programs.

We start with the premise that students will have some means, perhaps incomplete, of solving problems. Forming a model of the student means learning about his general model and inference procedure. The variance of student behavior from the ideal and the extent to which a program can learn a model different from its own ideal model crucially determine the extent to which the program can understand the student's behavior and adapt its instruction to his approach. AI representations such as semantic networks and production rules generally offer the kind of indexing that enables a program to adapt to a student's order of inferences for solving a problem. For example, each relation between a symptom and diagnosis might be represented as a single production rule in the general model. Thus, regardless of the order in which the student gains information about the problem, the program can predict what he knows by applying the production rules.

However, to recognize a different inference procedure (the strategy that determines the order in which inferences are made) requires that the procedure be represented separately in what is called a functional model (Section 6). While expert systems generally encode knowledge to fit the "indexing" criteria, very few separate out the inference procedure or encode it so that it can be reasoned about by a modeling program. That is, most of today's expert systems could not be used as a representation of subject matter to provide the modeling assessments discussed in Section 4.

Of course, we might take an even broader view of the modeling problem. Insofar as we expect our students to perform and learn in ways that we cannot mimic on a computer today, the instructional program must have some way of articulating its limits and the assumptions upon which it is based, thus enabling students to relate its embedded theory to the larger world (Winograd & Flores 1986, Suchman 1985).

3.4.3. QUALITATIVE REASONING The representation of qualitative models is a burgeoning topic in artificial intelligence today (Bobrow 1984); this review can only provide a very high-level introduction. The term *qualitative reasoning* is usually associated with simulation models. However, in this review, I adopt the view that most of AI is concerned with the representation

of qualitative models. In stating this pattern, I draw distinctions different from those that appear in the literature of the 1970s:

- In the literature, a distinction is not generally drawn between the programming language, the inference process, and the general model. A paradigmatic example is describing MYCIN in terms of goals and chaining through rules (Davis et al 1977). This abstract description was very valuable for defining a new method of computer programming for implementing expert systems. However, in describing MYCIN as a model, we move up to the "knowledge level" (Newell 1982) and describe its diagnostic inference procedure and the classification of diseases embedded in the rules.
- The term "qualitative model" has been restrictively applied to simulation models of processes based on a description of functional components (Bobrow 1984). Other models of processes are given domain-specific names (such as "nosology" in medicine) or are described only in terms of general constructs used to represent many different kinds of concepts [such as "script" (Schank 1975) or "state-transition network" (Brown et al 1973, Stevens et al 1982)]. Classification and behavioral models of processes are not an inferior version of functional descriptions. They properly represent a certain form of knowledge that is common in domains such as medicine and everyday reasoning (Norman et al 1976). While this knowledge may not provide the same level of explanatory detail as a simulation model, it still functions as a model by enabling useful predictions and explanatory accounting. It is now generally believed that this is the form of expertise associated with quick, routine problem solving (e.g. see Feltovich et al 1984). Indeed, this is the kind of model described by Minsky in his frame theory (Minsky 1975).

Separately identifying the model and the inference procedure in a knowledge base makes it clear that there are several kinds of qualitative models (Section 6). It is convenient to call them all qualitative models because, first, they are all non-numeric, and, second, they are models describing situations in the world, so they can be recognized and acted upon. The specific computational formalisms used to represent qualitative models of processes are described in Section 6.6 (see also Clancey 1986b).

3.5. *Use of Models: the Central Role of Diagnosis*

Diagnosis plays a special role in instructional programs. First, it is a common problem-solving task that we attempt to teach—e.g. debugging computer programs and diagnosing patients or circuits or equipment. More significantly, diagnosis plays a central role in teaching itself. An analogy can be drawn with the process of monitoring the behavior of a physical system (such as an automobile engine), looking for discrepancies from an ideal specification, tracking discrepancies back to structural faults or misadjustments, and repairing by changing the device. In instruction, we monitor the behavior of the student (a cognitive system), look for discrepancies from the ideal specification (target problem-solving model), track discrepancies back to faults in the student's presumed world model or inference procedure, and "repair" the student by instruction. This process of causally tracking backwards from discrepant reasoning behavior to hidden faults in a cognitive system is called diagnostic modeling.

Note that there are several levels of causal description: the discrepant behavior, the system fault or malfunction (called a bug), and the explanation of how this bug came about. For example, in a computer program, we distinguish between the incorrect output (what the program did wrong), the bug in the code (what statements are missing or incorrect), and the explanation for this bug (why the programmer did not use the correct code). In medicine, we distinguish between a behavioral symptom such as a headache, the disease (or bug) in the body, and how this person happened to have this disease (e.g. why the immuno-response system failed to function to prevent the disease). In computer system diagnosis, we distinguish between a system crash and the underlying cause of the problem (e.g. a queue that exceeded its resources because an I/O controller stuck) and the environmental problem (e.g. heat) or design flaw that brought about this situation. At each point we proceed from system behavior, to a mechanistic explanation of how the system produced the behavior (bugs), to causes outside the system affecting system input or internal structure.

3.5.1. BUGS

Figure 7 summarizes the different levels of analysis involved in cognitive modeling, with an example from computer programming to make it concrete. The number of sources of error is somewhat surprising when it is laid out this way.

In general, a bug is some structural flaw (faulty part) manifested in faulty behavior (a process). Thus, the term *bug* is used to refer to the incorrect part of a constructed procedure (e.g. incorrect statement in a computer program). It also refers to an incorrect inference procedure (e.g. error in student's subtraction procedure) and, by extension, an error in the student's general model (e.g. believing that $9 - 4 = 6$). An incorrect general model is commonly called a "misconception" (such as a misconception about the cause of a disease). Errors can be combined, because an incorrect computer program may involve a combination of a misconception about the operators of

the computer language and an error in the inference procedure by which the student has pieced together these operators to accomplish some goal (i.e. how he writes a program).

Note that Figure 7 expands the definition of an inference procedure given earlier. It shows that inference procedure controls interaction with the outside world (e.g. to make observations or write things down). The inference procedure also applies general knowledge to the problem at hand. Parts of the general model applied to the current problem might be inferred rather than explicitly stored. For example, a programmer might infer that an "integer statement" should be placed at the start of a program because that is where "declarations" are placed. Thus, specific facts are inferred from properties of classes. The figure also indicates that part of the general model may have been learned through processes such as analogy with other models (Matz 1981, Gentner & Stevens 1983b). For example, knowledge about programming languages in general might be applied to form a model of a particular language (Johnson & Soloway 1984). Whether this occurs during problem solving or as part of the initial learning, and to what extent this background knowledge consists of unformalized "raw experiences" (Winograd & Flores 1986), may be relevant considerations, but they have not played a part in student modeling research in AI.

3.5.2. THE ORIGINS OF BUGS The instructional programs we are studying are particularly fascinating because of the researchers' attempts to understand the origins of bugs. This interest stems from several goals. First, if a program is constructed with a preenumerated set of bugs, not all students may fit the program's fixed models. A more capable program would attempt to generate a description of bugs from patterns in a particular student's behavior and a model of how bugs come about, called a generative theory of bugs. Second, there may be too many possible bugs to preenumerate practically. For example, the possible misconceptions in medical reasoning are countless. A generative theory makes it possible to engineer an adaptive program more efficiently. Third, as part of the science of instruction, researchers want to understand the origin of bugs so they can more appropriately design instructional sequences, to prevent bugs from forming or to catch them before they become ingrained.

The prevalent model, holding strong sway in the community, is that reasoning is not random; there is a causal explanation for every error. These explanations fall into three categories:

1. **Mis-learning:** The student's model of the world, his believed situation-specific information, or inference procedure is incorrect because of a learning error: a faulty textbook, an over-generalization, a false analogy, etc (Matz 1981, Sleeman 1984a).

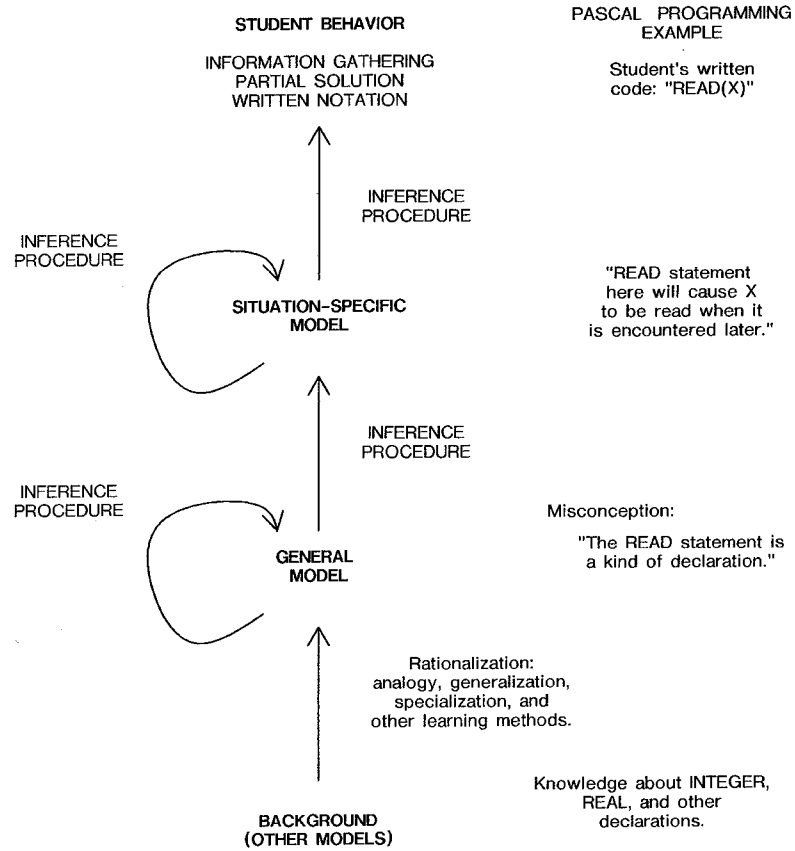

Figure 7 Levels of analysis in cognitive modeling (example from Soloway et al 1982).

2. **Construction:** The student's model and inference procedure did not allow complete solution of a problem. According to repair theory, weak problem-solving methods allow resolution of an impasse; the core deficiency is manifested by different constructed bugs, according to the context. Bugs may "migrate" and not be explainable by the student (Brown & VanLehn 1980).
3. **Slip:** The student knows better, he just didn't perform properly. Slips are caused by fatigue, emotional problems, conflict in memory, "cognitive overload," reverting to a previous error, or even unarticulated wishes (Freud 1965, Norman 1979, Norman 1980, Norman 1982, Matz 1981).

Understanding the origin of bugs can be equated to articulating a "deep structure" that accounts for implicit regularities in reasoning (Brown et al

1977). Notice that models of physical systems have a deep structure, too. So this idea of detecting regularities and giving a causal accounting plays a role in both first-order modeling (e.g. medical diagnosis) and the second-order modeling of cognitive diagnosis, modeling problem-solving itself. This commonality is fortunate, for it provides us with even more examples to study: A better understanding of cognitive diagnosis models in instructional programs helps us to understand diagnostic models in expert systems and vice versa.

A cognitive diagnosis drives the instructional program's explanations and selection of new examples and problems. However, we do not understand the "repair process" (Brown & VanLehn 1980) that is going on in the student's mind (a mechanistic account of how a misconception or gap leads the student to revise his model). Consequently, it is unclear just how good a cognitive model must be for instruction to be effective. To some extent, the research of the past decade constitutes a scientific foundation that we will adopt in the future to construct pragmatic instructional programs, perhaps with less complex cognitive modeling capabilities than our theories allow (Rouse & Morris 1985, Wenger 1986).

The reader may want to refer back to Figure 7 at this point to review the relationship between observed student behavior, bugs in the situation-specific model, and bugs in the general model.

4. STUDENT MODEL ASSESSMENTS

The simplest kind of assessments classify the student in terms of behavior, preferences, or background. Such a stereotype user-model (Rich 1979) is useful for "priming" the modeling process when there is no prior experience on which to base expectations (Carr & Goldstein 1977, Clancey 1982a). Beyond this, what do existing programs learn about students as a result of watching them solve problems and evaluating their performance? What can these programs infer about a student? Recalling the distinction between a general (domain) model and a situation-specific model (Section 3.2) and the relation among the ideal model, bugs, and underlying misconceptions (Section 3.5.1), a logical spectrum of interpretations can be laid out and illustrated by existing programs (summarized in Figure 8):

1. **Determine whether statements about the problem to be solved and the general model are correct.**
   a. Determine whether a student's statements about the problem are correct.
      - Example: In GUIDON, a student supports a medical diagnosis by listing facts about the current patient. The program indicates if the facts are correct before evaluating the hypothesis or its justification.

	ASSESSMENT	ASSOCIATED TERMS	EXAMPLE PROGRAMS
1.	statements about general model	quiz question-answer	SCHOLAR
2.	consistent situation-specific model	hypothesis evaluation	SOPHIE
3.	general model and problem information used to form partial solution	overlay or differential model	GUIDON WEST
4.	general model	errors called misconceptions	WHY MACSYMA ADVISOR PROUST
5.	inference procedure	errors called mal-rules or bugs	BUGGY DEBUGGY PIXIE ACM
6.	origin of inference procedure	model of learning	SIERRA

Figure 8 Assessments and associated terms for well-known modeling programs (numbers correspond to the discussion in Section 4).

   b. Determine whether a student's statements about the general model are correct.
      - Example: In SCHOLAR, a student is quizzed about geographic and demographic facts. The program indicates if his answers are correct by examining a semantic net.
      - Example: In WHY, a student is quizzed about the necessary and sufficient causes of heavy rainfall (e.g. "Do you believe that every place with mountains has heavy rainfall?"). The program indicates if his model is correct.
      - Example: In GUIDON, when a student justifies a hypothesis by saying that a particular symptom is evidence for a particular disease, the program indicates whether this is part of the general model.

2. **Determine whether a student's solution (situation-specific model) is consistent with the general model and the situation-specific information he has received.**
   - Example: In SOPHIE, a student's hypothesis about a possible circuit fault is evaluated with respect to the measurements (voltage, etc) he has taken. His hypothesis may not be the actual fault known to be present, but it may be consistent with the available information and the general model of how the circuit components interact. An error here could be caused by a faulty inference procedure (not taking available information into account) or by a faulty general model (interpreting information inappropriately).
   - Example: In GUIDON, a student's hypothesis about the possible presence of a disease is evaluated with respect to the symptoms he knows about. Treating the symptoms independently, the program determines if they are consistent with his disease hypothesis, even if this is not the diagnosis that could be deduced from further information.
3. **Determine what knowledge (domain model and inference procedure) the student has used to form a partial solution (make a hypothesis).** Generally reformulated as, What aspects of the program's general model are consistent with the student's partial solution?
   - Example: In WHY, a student's prediction about heavy rainfall occurring in a particular location (e.g. "Southern California has heavy rainfall") is related to the general model. One can then determine whether the student has considered a necessary factor (presence of mountains) but believes that it is sufficient, or whether he has omitted another necessary factor (e.g. presence of moisture-bearing air mass).
   - Example: In GUIDON, a student's diagnosis is related to the general model, producing a list of symptoms (or combinations of symptoms) that are consistent with the diagnosis.
   - Example: In WEST, the program interprets a sequence of student moves in terms of the available operators, and determines which aspects of the general model (spinner combination operators) are consistent with the student's play. For example, the student may be avoiding certain operators (e.g. use of parentheses) whose use is consistent with the general model of winning the game, while his use of other operators (e.g. division) is consistent with the model of how the game should be played.

4. **Determine which incorrect general model is consistent with the student's behavior.** This is often called a student bug model.
   - Example: In WHY, if the student incorrectly believes that heavy rainfall occurs in a particular place (making a false prediction), the program extracts a general rule of the form "Known factor is a sufficient cause" (e.g. mountains are a sufficient cause of heavy rainfall). This general rule is then stated for the student, to see if he believes it.
   - Example: In the MACSYMA ADVISOR, a student's sequence of Macsyma operators is related back to his goal to determine his beliefs about the prerequisites of these operators and what they produce as output. A consistent interpretation may involve positing a statement about an operator that is incorrect, a bug in the student's general model about Macsyma.
5. **Determine what nonoptimal or incorrect inference procedure is consistent with the program's general model and the student's behavior.**
   - Example: In DEBUGGY, a student's subtraction solution is related to incorrect subprocedures (operators), such as an incorrect procedure for borrowing.
   - Example: In ODYSSEUS (Wilkins et al 1986), a student-modeling program based on HERACLES (Clancey 1985b), the sequence of student requests for data is related to tasks for making a diagnosis; this allows detection of reordered and deleted conditional tasks in the ideal inference procedure.
   - Example: In WEST, a student uses mathematical operators to combine numbers on a spinner, thus determining how many spaces his player will advance on a game board. Assuming that the goal is to reach the final place on the board first, the program determines if the student's choice of operators is optimal—that is, consistent with the inference procedure dictated by the goal.
6. **Determine what underlying misconceptions explain the student's incorrect general model or buggy inference procedure.**
   - Example: In REPAIR THEORY, impasses are generated from a subtraction procedure that is missing certain subprocedures. Inference procedure bugs are generated by weak problem-solving methods that repair the impasse, based on some assumptions about the correct form of a solution, such as putting a number in every column. Thus, improper behavior is explained in terms of a missing

part of the general model and a procedure for working around the gap.

- Example: In PROUST, programming errors (incorrect lines of code) are explained in terms of bugs in the general model (incorrect beliefs about what an operator, such as a READ statement, will do); these are in turn explained in terms of a mechanism, such as a false analogy [relating READ to an INTEGER declaration, and thus concluding that READ(x) is a general declaration about the variable x]. Thus, improper behavior is explained in terms of an incorrect general model and (informally, not by a simulation program) by a process for how this incorrect model may have been learned.

In subsequent sections I consider how representation requirements and methods for making these assessments vary across domains.

5. CONTRAST BETWEEN FORMAL AND PHYSICAL DOMAINS

The point of this section is to compare problem solving in different domains to determine to what extent methods developed in one problem area can be successfully applied to other kinds of problems. First I consider how problems across domains can be classified. Then I show how modeling problems that arise in one domain are minimized or do not occur in others. This analysis is useful for someone who wants to apply the modeling methods described in this paper to a new problem area or to define a problem so that it fits a given method, with an understanding of what aspects of student reasoning may be consequently incompletely modeled.

To compare problems in different domains, I describe inference procedures in terms of model-manipulation operators. That is, the solution, the situation specific-model, is viewed as an object that is modified during problem solving until a certain form is attained. For example, the form of a correctly simplified linear algebraic equation is "<variable> = ..."; the form of an executable computer program consists of validly composed expressions in a language, which perform the desired computation; the form of a diagnosis is a parsimonious network that causally relates the manifestations that need to be explained. An inference procedure, abstractly stated, consists of operations for critiquing and improving a partial situation-specific model on the basis of its form, thus calling into play selected aspects of the general model. This view of problem solving, elaborated upon below, provides a basis for comparing modeling methods and difficulties in mathematics, programming, and diagnosis.

Briefly, modeling is made much easier if the inference procedure consists

of well-defined operators, if the operators are applied in a "written notation" (i.e. using pencil and paper, other physical objects, or equivalent computer graphics) and if the domain is axiomatized. For example, all of these are satisfied by subtraction, but none are satisfied by medical diagnosis. In subtraction, misconceptions are apparently irrelevant; in medical diagnosis they are unbounded. The subtraction inference procedure is well known, to say the least, and is precisely what we attempt to teach; in programming and diagnosis we don't even understand what experts can do. We mainly attempt to teach facts about the programming language or the physical system to be diagnosed (the general model) and patterns for using these facts in common problems. Viewed more generally, instruction focuses on functionality of the domain, and we find it difficult to talk about the inference procedure.

Figure 9 places domains of instructional programs in a two-dimensional space. Along the top, I contrast an algorithmic, formally derivable inference procedure with one that is heuristic and provides no guarantee of constructing the correct answer. For example, the procedure for subtraction is algorithmic; the procedure for writing programs is heuristic. The distinction exists because of uncertainty in the problem information (incomplete or uncertain measurements), uncertainty in the general model (as in medicine), or because the space of possible solutions is combinatorially too large to search systematically (Buchanan et al 1969). Note that the matrix describes the ideal subject matter, not its form in a particular person or how it may have been modeled. A given person might do algebra heuristically or a computer program might have an algorithm for diagnosis. In general, the matrix describes the "real world," as opposed to particular models of it. However, I mention scientific laws parenthetically to show where they would appear.

By "system functionality," the second axis, I mean "how the domain works." We can draw an analogy among mathematics operators, programming language operators, circuit operations, physiological functions, etc. These operators define how the domain works, what can be computed, and what processes can occur.

The question is, what enables this functionality to occur? Is the system formally defined—that is, something someone designed (such as an electronic circuit) or something that someone asserted (such as geometry axioms)? Or is the functionality of the system part of the world, with an indeterminate basis (such as the operation of the human body) or unbounded and subject to change (such as the environmental loads that might be placed on a computer system)? The boundary between formal and indeterminate functionality is not sharp. For example, electronic circuits have defined functionality (unlike components of the human body), but they rest upon a natural system of physical interactions that is not formal or constructed by some person.

To put this another way, referring to general and background levels shown

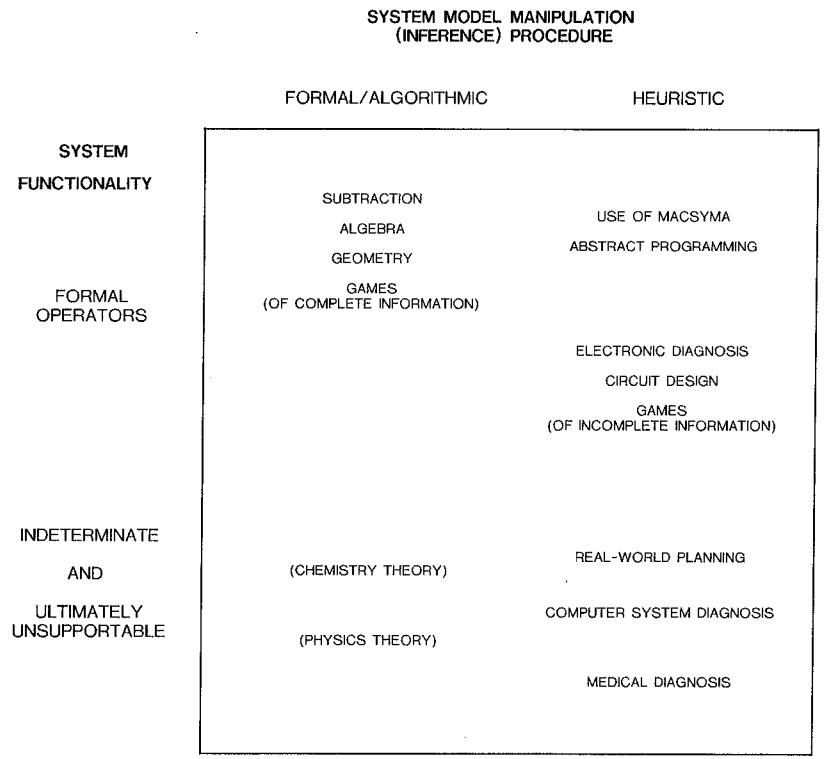

Figure 9 Functionality of domain operators related to model manipulation.

in Figure 7, if the functionality is formal, the general model consists of axioms, with well-defined terms. The model is complete, constituting a closed world (Carbonell & Collins 1973, Collins et al 1975, Reiter 1978). For physical or natural systems, there are no formalized axioms, defined terms, or syntactic inference operators. There is no bottom; we can always inquire about more detailed causal interactions. We may describe functionality in terms of physical laws, but these are hypotheses, not axioms; they may have a basis in deeper theory and are subject to error.

For completeness, I include "real world planning and programming" as an example of an indeterminate domain. In particular, I contrast the problem of writing a program to do abstract operations (such as list manipulation) with programming that involves interacting with the world (such as an airline reservation system). We may rigorously specify how a real world program must work, but we cannot practically anticipate (formally define) all of the problem situations that may occur. The system is not closed. There is of course a spectrum here, with highly repetitious and isolated problems at one

end (such as accounting) and one-of-a-kind, noisy problems at the other end (such as battlefield management).

5.1. *Written Notation and Operators*

Modeling reasoning involves formalizing an inference procedure. For subtraction, this is trivial because the inference procedure is a well-known algorithm and there is a written notation for describing the situation-specific model. However, parallels can be drawn that make it possible to model heuristic procedures and systems of indeterminate functionality.

Recall that an inference procedure constructs a situation-specific model (Section 3.2). When the model is represented in a written notation, as in mathematics, the primitive operators of the inference procedure are notational—that is, in terms of syntactic manipulations of the representation. In subtraction, these primitive operators correspond to decrementing a column, adding ten to a column, shifting left, finding a difference, etc (Ohlsson & Langley 1985). The subtraction inference procedure is guaranteed to produce a correct answer.

What are the corresponding primitive operators in medical diagnosis? There are no notational operators because there is no written notation for solving diagnostic problems. The inference procedure used by people for generating, relating, and testing hypotheses is heuristic and could not even be stated as a simulation model until the last decade (e.g. see Pauker et al 1976, Elstein et al 1978).

Although physicians do not speak of or teach a formal set of "diagnostic operators," we can now model the inference procedure in a way that parallels the subtraction algorithm—that is, as a program for manipulating a representation, the situation-specific model.

Figure 10 shows a simplified representation for diagnosis, as used by NEOMYCIN. According to this view of medical diagnosis, the object is to construct a well-formed graph that represents a situation-specific model. All of the symptoms that need to be explained (e.g. seizures) are linked to internal states that are causing them (e.g. increased intracranial pressure) and the disease processes that caused these states (e.g. an infection). Proceeding upwards in the diagram, the "explanation" becomes more specific in terms of causes and subtypes of processes. Thus, the graph takes the form of a "proof." Causal links correspond to theorems, so that each link has a corresponding graph that "proves" its validity in this case. Of course, the sense of "proof" here is in terms of the logical form of the diagnostic explanation, as an argument, rather than a sound derivation.

For comparison, a partial situation-specific model, as expressed in a written calculus, is shown for geometry theorem proving in Figure 11. In contrast

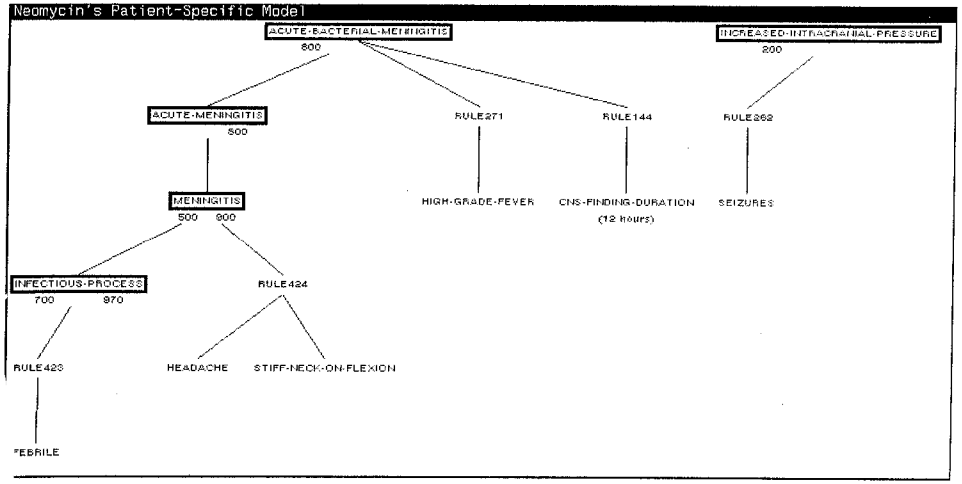

Figure 10 Written notation for partial situation-specific model in medical diagnosis (from Clancey 1986b).

with geometry theorems, causal associations are uncertain. They are heuristic associations, relying on mechanisms that might not hold in the given case. In general, they are more like axioms or laws, because they tend to be empirically supported hypotheses rather than theorems derived from a general model.[2] Also, there may be separate, disconnected parts of the situation-specific model; the problem is to explain all of the abnormal findings. This is almost backwards from theorem proving, in which only some of the givens need be used and the root of the proof, the thing to be proven, is known. Finally, geometry theorem proving does not deal with "instances" in the world, in the sense that each human body is an instance of a general system. Thus, all inferences in geometry are general, and what is proved can be directly applied to new problems.

Using this proof-like notation, the reasoning steps of diagnosis can be expressed as graph construction operators. For example, "test hypothesis" corresponds to growing support links down from a process or state description. "Group" corresponds to linking states or processes by a containing category (subsumption) or a common cause (Pople 1982). Algebraic operators for transforming equations are similar: "collect like terms," "multiply through by denominators to remove fractions" (Sleeman & Smith 1981). Operators for geometry are much simpler, involving just forward and backward inference. Moreover, algebra and geometry, unlike subtraction, involve searching memory for operators to apply. The algorithm allows for some discretion in the order in which operations are done.

In summary, an inference procedure can be described as operators for

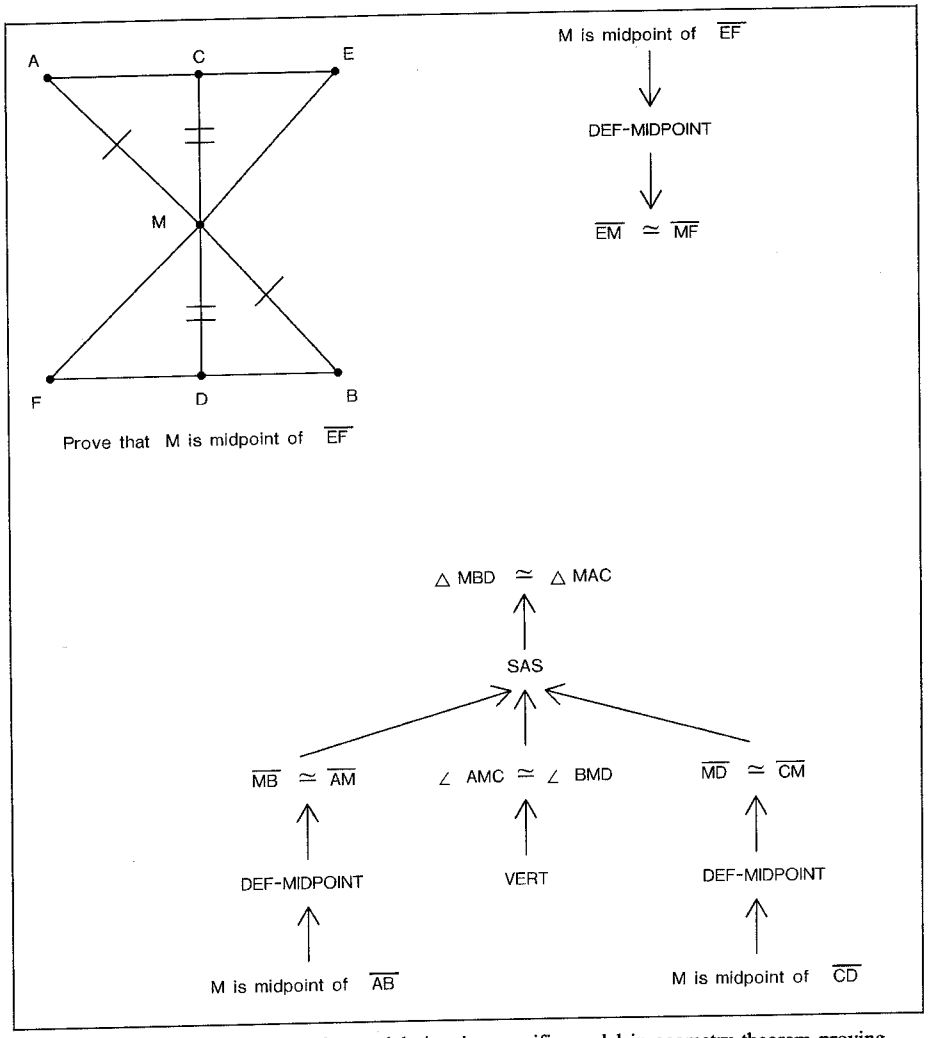

Figure 11 Written notation for partial situation-specific model in geometry theorem proving (from Anderson et al 1985).

manipulating a situation-specific model. This is a constructive process that involves posting intermediate results, searching for relevant associations in memory, and perhaps backtracking to try alternative approaches. The important, unifying idea is that the solution at any point, the situation-specific model, must have a certain form. This form can be stated as constraints that relate problem information to the solution (a diagnosis, a theorem, a computer program, a value for an algebraic variable). In diagnosis and geometry

theorem proving, these are the constraints of completeness and consistency in a proof. In programming, the constraints are computational, relating to the goals of the program. In algebra, the value for the variable must satisfy the original equation.[3]

5.2. *Subtraction Compared to Medical Diagnosis*

On this basis, I summarize the differences between subtraction and diagnosis, contrast the nature of bugs, and articulate modeling difficulties:

- There is no established written calculus for diagnosis. The inference procedure used by people constructs a model that is almost exclusively mental and is not expressed in a written notation (Rouse & Morris 1985). Nevertheless, people sometimes speak in terms of operators [e.g. "split the hypothesis space" (Brown et al 1982a), "go down the tree of diagnoses" (Clancey 1984b)], and some programs have modeled human reasoning in terms of formal operators (Newell & Simon 1972, Larkin et al 1980, Clancey 1984b).
- The inference procedure of diagnosis is heuristic, not algorithmic:
  - It is inherently based on activation of associations in memory, in contrast with subtraction, in which the organization of memory plays no apparent role (Rouse & Morris 1985);
  - more than one operator may apply, with a problem of establishing focus and reasoning efficiently;
  - reasoning may involve working backwards from a hypothesized part of the solution; and
  - multiple, situation-specific models may be constructed and compared (Pople 1982, Patil 1981).
- The number of domain facts in medical diagnosis is tremendous, so that most errors for students are in the domain model (Feltovich et al 1984). By comparison, most errors in subtraction involve incorrect manipulation of the notation—that is, errors in the inference procedure, not mathematics facts (Young & O'Shea 1981).
- Impasses occur when domain facts are insufficient to construct an adequate situation-specific model. However, adults stop and ask questions or seek help; they are less likely to make the unsupportable inferences posited by repair theory in the problem solving of children. Rationalizations (to argue away data or revise the general model) may be necessary to achieve a consistent model (Clancey 1984a), but they are deliberate causal arguments.
- Heuristic operations to ensure completeness, such as asking general questions early and gathering circumstantial evidence before requesting laboratory results in order to ensure completeness, are explicitly taught and sometimes forgotten. Thus, procedure errors corresponding to subtraction bugs can occur. However, if a student does not form a hypothesis, we cannot necessarily say that it is because he didn't try; automatic processes of memory are involved.
- The domain facts of diagnosis are not based on known axioms. Incorrect domain beliefs may derive from other incorrect facts, ultimately caused by mislearning, constructions, or slips. Identifying a misconception is just the first step; the teacher must always reason about its cause and correct that as well. It is assumed that errors in the math table or notational procedure of subtraction are not based on misconceptions. Furthermore, because domain facts are not axiomatized, they are always contextual, based on assumptions about the world. "Causal theorems" do not always apply and may interact, so they cannot be applied independently.

From this discussion, it should be clear that "bugs in subtraction" and "bugs in diagnosis" are quite different beasts. The idea of a bug has been considered perhaps too generally in the literature, particularly when different domains are compared, resulting in failure to properly distinguish between domain model and inference procedure errors. Formal domains, in contrast with natural domains, have

- an axiomatized general model;
- algorithmic, syntactic inference procedure;
- written representation of the situation-specific model; and
- minimal concern with focusing attention.

A mathematics modeling program has a written notation to interpret, and bugs are errors in its manipulation. In contrast, in medical diagnosis there is traditionally no written notation and "bugs" are predominantly misconceptions about processes in the system being diagnosed. Programming and electronic diagnosis fall in between, and are considered next. I return to the confusion about bugs in Section 6.9, after explicating in more detail how qualitative models of processes can differ, even within a particular domain.

5.3. *Artifactual vs Natural Functionality*

In general, misconceptions may lie in the functionality of the subject system or in how this functionality is realized by underlying mechanism. For formal domains, there is no underlying mechanism. The functionality of operators in formal systems is given by axioms. Operators in computer programs (such as a "read statement") are well-defined in a similar way.

In physical systems, we have two cases. For artifactual (man-made) systems, such as electronic circuits, functions are well-defined but they rest on

natural, physical models. For example, the desired behavior of a transistor (excepting timing conditions) is well-defined, but how the materials in a transistor bring about this behavior is open-ended and involves knowledge of quantum mechanics. This is important because the student may have misconceptions about the faulty behavior of a transistor based on misconceptions about how a correctly functioning transistor works, though there is a practical level below which normal problem solving does not go. Electronic circuit design is similar.

In other domains, such as medical, cognitive, and computer system diagnosis[4] and most real-world planning tasks (Hayes-Roth and Hayes-Roth 1979), the functionality of the system to be interpreted or constructed is not fully known or predictable. For example, a therapy may not work; a store may not have the desired goods. Variations from the desired behavior are not enumerable, and external influences are unlimited (though there may be patterns). Student misconceptions derive from misunderstanding system function or how the behavior is derived. It is relatively easy to determine that the student is applying an incorrect fact (e.g. "if gargling improves a sore throat, then the patient had a viral infection") by asking the student or by contextual inference. However, it would go beyond human capability to construct a program capable of always explaining such bugs in the general model in terms of underlying misconceptions. The same observation holds for domains of physics, chemistry, meteorology (Stevens et al 1982), etc. There is a practical level below which routine problem solving need not go, but, unlike artifactual systems, this level is not formal and is not based on deliberate design.

Thus, proceeding from formal to natural domains (Figure 9), there is an increase in the number of possible interpretations and amount of world knowledge that might be involved.

Consequently, while both formal and physical systems admit to construction of simulation models of ideal (correct) behavior, the approach for explaining bugs must be fundamentally different. When errors are not just incorrect or forgotten axioms but involve complex causal models, it is insufficient simply to point out the errors, as in the MACSYMA ADVISOR. Instead, the program must have some way to understand a student's explanation and ferret out the underlying misconceptions. No cognitive modeling program today can do this, except by the use of a bug library (Soloway et al 1981). It is tempting to criticize this approach for being ad hoc, but domain differences prevent simply applying the generative approach used for modeling mathematics errors.

Moreover, experiments with naive subjects indicate a proclivity for misconceptions based on "causal reasoning," where no underlying processes are involved, or the attribution of complex mechanism, where all functionality is formal (Bott 1979). These results suggest that the "closed world" approach of mathematics modeling programs may be missing the strong, nonformal metaphors people use in understanding new problems, even in formal domains. Analysis of strategies for playing the WEST game supports this view (Burton & Brown 1979).

5.4. *Algorithmic vs Heuristic Inference*

Returning to Figure 9, I briefly consider the difficulties a heuristic inference procedure imposes on modeling.

Problems in subtraction, algebra, programming, and diagnosis, in contrast to geometry theorem proving, do not start with a conclusion that must be shown to be correct (Greeno & Simon 1984). However, all domains have constraints on the form of the model (e.g. in medical diagnosis, to explain all serious findings). Once a partial solution is hypothesized, it may be supported by a theorem-proving, top-down approach (Murray 1985, Genesereth 1984). This form imposes constraints on the inference procedure:

- Procedural errors include: making unuseful or inadequately supported inferences, refining a line of reasoning before contrasting alternatives, and failing to discover situation-specific facts (symptoms, specifications) that must be incorporated in the model.
- The nature of the inference procedure is orthogonal to the nature of the operators. Formal operators may still be applied by a heuristic inference procedure. Programming, as performed by people, is not algorithmic, but heuristic. Similarly, operators in board games are also well defined, but the situation-specific model may be inherently uncertain, as in bridge, or too numerous and complicated to fully explore, as in chess.
- Formal problems occur in a closed world, so the problem solver can assume that he has been given all relevant information. Problems in natural domains are open.

Coupling these observations with what has been accomplished in student modeling, we must temper the successes of mathematical modeling with sober awareness that we have just begun to model the complexities of human reasoning for nonalgorithmic problem solving. Our models of what experts do in controlled situations are meager (Chi et al 1981, Larkin et al 1980, Chi & Glaser 1982, Chi et al 1986, Glaser 1985); thus our ability to model what a student is doing is all the more uncertain. Results in modeling heuristic inference procedures are tentative and incomplete; they barely indicate what might be possible and do not cope with the difficulties in any general way.

6. TYPES OF QUALITATIVE PROCESS MODELS

The spectrum of "explanation" spanned by student-modeling programs ranges from detecting discrepant behavior, to relating this behavior to incorrect beliefs, to accounting for these incorrect beliefs. However, it is possible to

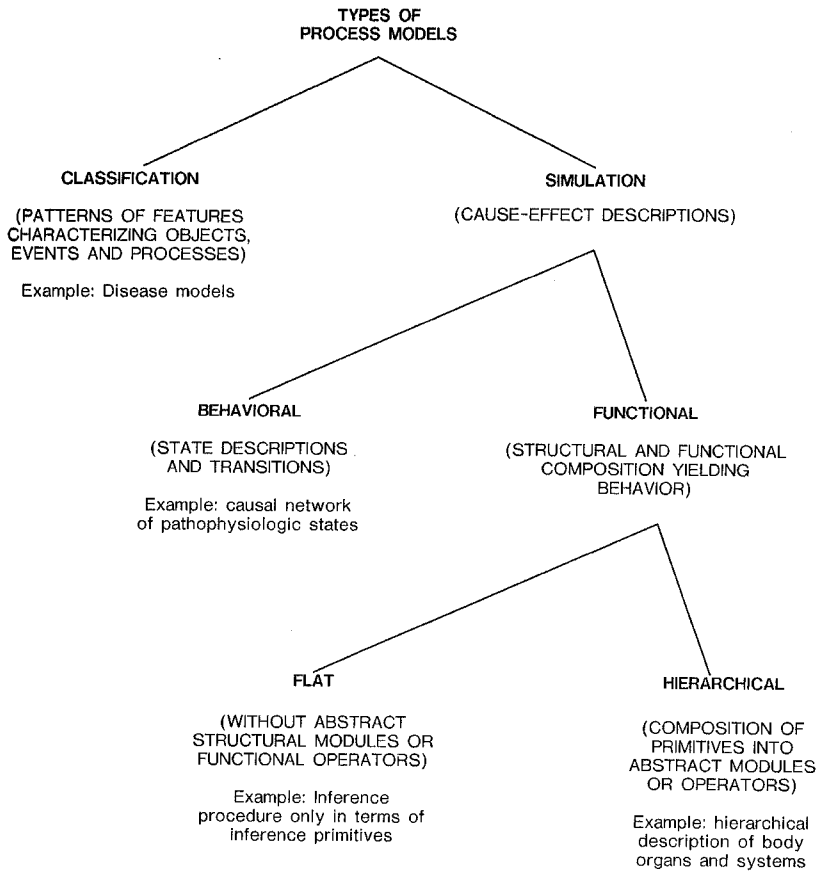

Figure 12 Types of qualitative models of processes.

construct such interpretations without modeling the inference process itself. In this section, I consider what a process model of reasoning is, how behavioral and functional process models differ, and how a functional model is represented. Figure 12 summarizes the distinctions made here.

6.1. *Process Models of Reasoning*

A process model of reasoning is a simulation model that explains behavior in terms of both the student's general model and his inference procedure (refer to Figure 5.) The simulation not only says what the situation-specific model is and how it relates to the general model, but it describes the process, over time, by which the solution was derived. A mark of a process model is that it describes problem solving in terms of shifting focus of attention. The model describes a sequence of focusing on situation-specific information and the

evolving situation-specific model, plus a procedure that determines this shifting focus, which I have termed the inference procedure. A process model is a strong argument for completeness of the explanation that relates the situation-specific model to the general model. Many researchers believe that accurate process models will provide a basis for improving instruction (Glaser 1983).

Many instructional programs do not construct process models of student reasoning. For example, SOPHIE checks to see if a solution is consistent with the given information, but it does not explain the order in which information is gathered or the order in which hypotheses were inferred.

BUGGY, ACM, LMS, and PROUST provide an intermediate case; they reconstruct the student's reasoning process but do not match this against his actual, step-by-step behavior. For BUGGY and ACM, working in subtraction, the notation is so constraining that a procedure can often be inferred just from the final solution, without even considering scratch marks (though this assumption may also account for part of the failure of these programs to explain some solutions). PROUST constructs a situation-specific model that accounts for the student's code and shows how it might be derived from beliefs about PASCAL operators. But it also works from a final solution, so it cannot verify that the inference procedure is the same as that used by the student. Emphasis is on detecting and explaining incorrect behavior in terms of errors in the general model, not the programming process. Here the formal domain operators and complex written notation greatly facilitate inferring the student's beliefs. The MACSYMA ADVISOR goes a step further and relates intermediate steps to an idealized inference procedure.

In simulating the inference process, programs like the LISP TUTOR, SPADE, and ODYSSEUS attempt to explain behavior in terms of the order of inference for modifying the situation-specific model. Here an attempt is made to match the step-by-step process that the student goes through as he gathers more information about the problem and/or writes down partial solutions [called model-tracing (Anderson et al 1986)]. Thus, ODYSSEUS explains why the student is making a hypothesis at a certain point, while SOPHIE only explains if it is consistent. The LISP TUTOR determines the student's goal for writing each piece of code at the time he writes it. Of course, what variations from the ideal model these programs can recognize is a different issue.

6.2. *Behavioral vs Functional Process Models*

Two types of models simulate the inference process: those stated in domain-specific terms, which I call behavioral models, and those stated in terms of abstract operators for manipulating models, which I call functional models. Behavioral models describe how a system appears in itself; functional models place behavior in a larger context, indicating the role it plays in achieving the properties of a larger system.

In general, a functional model describes the purpose of a system. Thus, a physical system would be described in terms of its functional components and how they interact, not just according to the behavioral relations among states. For example, a functional model of a radio would refer to amplification and locking on to a station, while a behavioral model alone would only describe current flow and changing voltages. We move from seeing behavior in isolation to accounting for patterns in behavior and finally to the larger goals these patterns satisfy.[5]

A functional cognitive model describes reasoning in a domain-independent way that focuses on model construction and use. Thus, NEOMYCIN'S goals relate to explaining/diagnosing a system's behavior: e.g. "trying to test a hypothesis." Rather than describe what the program is doing in terms of specific requests for information and inferences about particular diagnoses, we abstract a sequence of information requests, then relate this pattern to the more general objectives of forming a model of the patient (Clancey 1984b, Clancey 1983a).

Similarly, PROUST'S goals relate to designing/constructing a system that exhibits particular behavior: e.g. "implementing desired functionality" describes a pattern of inferences during program writing as a specification is transformed into the operators of the programming language.

Such a model describes not just what the problem solver is doing at the domain level. It characterizes the function of cognition in terms of operators for manipulating models. Necessarily, the general domain model is represented separately from the inference procedure, rather than being combined in specific situation/action patterns. That is, one separates "what is true about the world" (the general and situation-specific models) from "what to do" (the inference procedure), as shown in Figure 5. The inference procedure, as shown in the next section, is abstract and does not mention domain terms.

In contrast, behavioral models describe what a system does in particular situations (e.g. the symptoms of a physical system such as a malfunctioning circuit; the design decisions or diagnostic questioning of a cognitive system). A behavioral model may describe unobservable states (such as unarticulated inferences). Stated as situation-action patterns, a behavioral model may even simulate processes. This is how the diagnostic process is simulated in MYCIN (used by GUIDON) and how the programming process is simulated in the LISP TUTOR and the FLOW TUTOR (Gentner 1977). Note that a behavioral model of reasoning may articulate goals behind problem-solving steps, but they are always domain-specific (e.g. in medical diagnosis, trying to determine if the patient has an infection; in LISP programming, trying to add an element to a list). Figure 13 shows the possible combinations of models of the domain and reasoning, with examples of instructional programs. Note that the

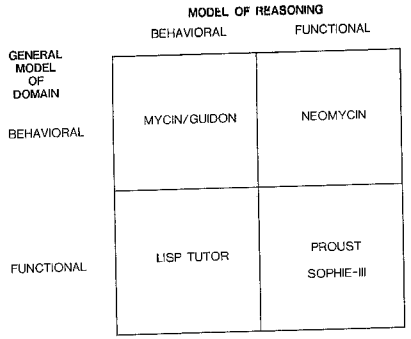

Figure 13 Comparison of programs in terms of model of the domain and model of reasoning.

implicit model of the domain in the LISP TUTOR, consisting of the input/output relations of procedural operators, is functional.

Again, to draw the contrast, the LISP TUTOR examines errors in terms of specific transformations of a partial computer program. It is still a process model, with the advantage of completeness shared by all simulation models, a result of reconstructing the student's situation-specific model. As a process model, it goes beyond evaluating consistency with the general model (correct use of LISP operators to accomplish goals) to model the order of programming transformations. But the model does not explain this ordering. The model is stated as a specific program that does not separate the domain model from the inference procedure, thus obscuring the fact that a model construction process is occurring at all.[6] Such a behavioral model should be contrasted with the nonsimulation, pattern-matching approach (Soloway et al 1982, Miller 1980), which makes no attempt to reconstruct the student's code and relies instead on a classification of errors to be detected (analogous to a behavioral, nonsimulation representation of diseases as simply prototypes to be matched).

The lock-step, "evaluate and correct each line of code" approach used in the LISP TUTOR is tractable because the problems admit to relatively linear, incremental transformation of the evolving program. More complex programs may have many different implementations and may be designed in too many ways to anticipate. A behavioral model describes what is done in a situation-specific way. Preenumerating the situation/action patterns is only tractable if the context of each situation can be narrowly construed, as in the simple abstract programming of the LISP TUTOR. Use of a separate, inspectable inference procedure—a functional model—provides greater flexibility for adapting to the student's order and syntactic variations and provides a parse of many alternative solutions without having to anticipate each situation beforehand (London & Clancey 1982, Johnson & Soloway 1984, Murray 1985).

If the model is written in an appropriate language, it can be used to articulate the process and provide assistance as well (Swartout 1981, Hasling et al 1984, Clancey 1985b).

6.3. *Levels of Abstraction*

The distinction between a behavioral and functional model can be laid out as a sequence of four levels of abstraction. Using a simple example from a hypothetical medical diagnosis program, we progressively separate the domain model from how it is applied and abstract the inference procedure.

2. **Instance-specific model**
(behavioral, domain is one patient)

   Pertains to particular problem instances; stated as a procedure of specific situation/action patterns.

   Example:

   `If (FEVER mary) then (INFECTION mary).`[7]

   If Mary has a fever, then Mary has an infection.

2. **Domain-specific, proceduralized model with variables** (behavioral, domain is one disease)

   Problem-instance objects replaced by variables; stated as a specific procedure of situation/action patterns. Situation descriptions are interpreted (supplied as data to the program).

   Example:

   `(PATIENT mary).`

   If (AND (PATIENT $PATIENT) (FEVER $PATIENT))
   then (INFECTION $PATIENT).

   If the patient has a fever, then the patient has an infection.

3. **Domain-specific, general model separated from inference procedure** (functional, domain is medical diagnosis).

   Domain concepts are classified as different terms (e.g. DISEASE and SYMPTOM). A general domain model consists of relations over these terms ("domain facts" such as CAUSES). A distinction is made between the general model and the situation-specific model (indicated by the PRESENT relation here, which strictly speaking should be probabilistically qualified). Both are referenced by the inference procedure, which is now expressed as a set of general situation/action patterns for accomplishing model construction tasks.

   Example:

   `(DISEASE infection) (SYMPTOM fever) (CAUSES infection fever)`
   `TASK: TEST-HYPOTHESIS`
   `FOCUS: (DISEASE $D)`
   `If (AND (CAUSES $D $S)`
   `(PRESENT $S $PATIENT))`
   `then (PRESENT $D $PATIENT).`

   If a disease causes a symptom and there is evidence that the symptom is present in the patient, then there is evidence that the disease is present in the patient.

4. **Inference procedure stated abstractly** (functional, domain is diagnosis using a classification general model). Domain terms and relations are more abstractly classified (e.g. DISORDER replaces DISEASE). Terms in the inference procedure now refer to systems, objects, and processes; relations refer to spatial, causal, temporal and probabilistic characterizations of terms.

   Example:

   `(SYSTEM patient) (DISORDER infection) (EFFECT fever) (CAUSES infection fever)`

   TASK: TEST-HYPOTHESIS

   FOCUS: (DISORDER $D)

   If (AND (CAUSES $D $E) (PRESENT $E $SYSTEM))
   then (PRESENT $D $SYSTEM).

   If a disorder causes an effect and there is evidence that the effect is present in the system being diagnosed, then there is evidence that the disorder is present in the system.

Here it should be clear that the domain relations (e.g. CAUSES) are not axioms but may summarize a complex line of reasoning about the process. Inferring this relation may involve a great deal of situation-specific reasoning, as opposed to simply "looking up the fact" in the general model.

Note that the transformation from steps 2 to 4 corresponds to the transformation from MYCIN to NEOMYCIN (Clancey & Letsinger 1984) to HERACLES (Clancey 1985c). Furthermore, we may continue the process of abstraction to produce more primitive relations for deriving the inference procedure itself. For example, we might derive the rule for TEST-HYPOTHESIS from some definition of what that task is supposed to do or we might derive the ordering of the clauses in this rule from efficiency constraints (Smith 1985). This level of specification may be useful for generating nonredundant prosaic explanations, as well as for making reasonable predictions

about what variations from this procedure might be expected in student reasoning (Clancey 1985b, Neches et al 1985).

6.4. *Flat Functional Models*

A special kind of functional model is nonhierarchical and expresses higher-order control by the ordering of situation/action patterns [e.g. ACM (Langley et al 1984) and LMS (Sleeman 1982)]. For example, such a model for algebra would represent the concept "collect/isolate variables" by an ordered set of primitive conditional actions. Similarly, the diagnostic procedure of NEOMYCIN could be flattened out by composing productions (e.g. the two productions "If A & B, then do G" and "If goal is G and C, then do P," where P is a primitive action, would become "If A & B & C, then do P"). Obviously, when the model is constructed by a machine-learning approach out of primitives, the common concepts we associate with macro or abstract operators cannot be assigned. In short, the functions of the functional model have not been specified. For constructing a diagnostic model (just trying to determine what the student's procedure is), this may not be a deficiency. However, it is striking that the procedures used by people in both formal and natural domains are hierarchical and make strong use of abstract functional terminology (e.g. in medicine, they group hypotheses into categories and differentiate among them). Unfortunately, one of the shortcomings of automatically-generated cognitive models (e.g. Langley et al 1984) is that they are flat and consequently leave out the abstract names that people use to group and understand complex operations.

6.5. *Idealized Competence vs Individual Models*

A further complication is introduced by Young in his arguments that hierarchical organization of rules or procedures and separation of the general model from the inference procedure are "psychologically unwarranted" (Young & O'Shea 1981). He argues that people do not necessarily know such well-formed procedures; their knowledge is more "anarchic." In evaluating this argument, we must distinguish between a computational representation that models what people are capable of doing (their competence) and a model of how reasoning is organized and activated in the human brain. In this sense, the separation of model from inference procedure and its hierarchical representation reveals something akin to a grammatical structure, expressing more of an ideal than what any given person knows. Like a grammar, a functional model is an abstraction of behavior, in this case relating to the problem-solver's larger goals and the constraints under which he operates. It does not necessarily correspond to what the problem solver is thinking about explicitly.

The thrust of my argument has been that the functional approach is tremendously valuable for generalizing results across domains. However,

while this might turn out to be a good software engineering principle for constructing instructional programs (and expert systems), we may discover that models of individual people must be inherently ill-structured, situation-specific, and even incoherent. Nevertheless, our theories of learning and communication need not be ill-structured and incoherent, and the hierarchical functional approach will serve us well there, even if a model of a given person's problem-solving knowledge is best expressed as flat, independent, situation-specific rules.

The argument is further complicated if the inference procedure reflects automatic processes of memory, as in medical diagnosis, which might be more properly modeled independent of the specific domain model. That is, a theory of diagnosis would properly describe the association and retrieval process abstractly, as a functional model, independent of particular medical facts (the general model). In this respect, we must be careful in applying Young & O'Shea's analysis of subtraction to other domains. As another example, it is possible for an experienced problem-solver to step out of an automatic mode of operating and systematically apply a procedure he knows explicitly (essentially moving from predominantly forward inference to means-ends, goal-directed reasoning). For further discussion, see Young & O'Shea 1981, Clancey 1984b, Langley et al 1984, Rouse & Morris 1985, Sleeman 1984b, Wenger 1986.

6.6. *Computational Representations for Qualitative Models*

Broadly speaking, qualitative models of processes are represented computationally in four distinguishable ways. These constitute the basic representational repertoire for modeling physical, inference, and communication processes in computer programs.

- **Process classification network**

  Description: Processes described as a hierarchy of input/output or cause/effect patterns.
  Examples: NEOMYCIN (diseases), MENO (bugs in code).

- **Causal-associational state-transition network**

  Description: Processes described by a graph of system states linked by cause and subtype.

  Examples: NEOMYCIN (pathophysiologic states), METEOROLOGY PROGRAM (Brown et al 1973) and WHY (rainfall temperature and moisture relations).

- **Functional/procedural-transition network**

  Description: Processes hierarchically described by functional composition of inputs and outputs.

Examples: GUIDON (discourse procedure) (Clancey 1979a, Woolf & McDonald 1984), NEOMYCIN (inference procedure), BUGGY (subtraction procedure) (Burton 1982), SOPHIE-III (electronic circuit module behavior lattice), PROUST (programming tasks and program schemas).

- **Structure/Function network**

  Description: Processes hierarchically described by structural composition of inputs and outputs within functional modules.

  Examples: QUAL (electronic simulation), DART (electronic simulation)

Behavioral networks (classification and causal-associational networks) are generally constructed to account for past behavior of a system, though they can be used to make predictions as well. They do not necessarily characterize the full state of the system being reasoned about on any level of analysis, and do not necessarily explain what state will follow from arbitrary initial conditions (for example, allowing that parts of the system are functioning normally).

A classification model allows accounting for behavior in terms of familiar patterns. Of course, even the simplest "explanation by naming" must be based on assumptions of what kinds of processes are possible and what the range of possible causes might include. For example, without further information, a new disease might be classified as a toxic effect, a deficiency of production, an infectious process, a congenital disorder, etc. Thus, a classification model, and the features thought to be causally related, might be based on an analogy with a better understood process.

A causal network, or script model, does allow simulation of behavior, but without accounting for how the behavior comes about. "Hidden" internal states may be described, but the purpose of transitions and how transitions follow from the structure of the system (the physical components) are not completely described.

In contrast, a functional network makes a claim about completeness of the explanation or system description. The purpose of the system is captured procedurally on multiple levels of abstraction, so that states can be related to functional goals and their subgoals. Each function can be expressed computationally as a subprocedure made up of ordered and controlled situation/action statements. Actions are either ordered invocations of subprocedures or primitive operations.

Finally, a complete structure/function model, to which the term "qualitative model" is usually applied in the literature, gives a full accounting for each component (structure) in the system in terms of its role in fulfilling the function of the system. Thus, a structural model of reasoning would relate descriptions of physical components in the brain to functional operations

(Kosslyn 1980). In general, cognitive modeling, and student modeling in particular, does not go down to this level; though of course structural modeling plays a crucial role in modeling other physical systems, such as an electronic circuit.

To better understand these representations of processes, consider the differences among a disease description, a network characterizing how manifestations are caused by internal states, and an anatomic model of some body system, characterizing its normal function. Notice how the inference procedure for modeling takes different forms as well: recognizing a pattern as a set of features; constructing a partial "historical" accounting for events (partial because perhaps only abnormal manifestations will be explained, not normal functioning of the system); and constructing a complete model of the system being diagnosed (accounting for both abnormal and normal system behavior in terms of structural components).

At this time, researchers are just beginning to identify this spectrum of qualitative representations for processes (e.g. Szolovits 1985). There is no single program that combines these models to reflect human reasoning in a robust, well-engineered way. Programs incorporate a model on one end of the spectrum or the other. For example, contrast the disease models of NEOMYCIN, which state causes and effects of abnormal processes, with the simulation model of QUAL in SOPHIE-III, which describes the physical structure of a system and how the components interact. Multiple representations, combining qualitative and quantitative models, are also possible and sometimes advantageous (Brown et al 1982a). Standards for representing qualitative models to facilitate computation, explanation, and acquisition are still in an early stage of formalization (de Kleer 1984, Neches et al 1985, Mitchell et al 1985).[8]

6.7. *Classification vs Simulation Models of Bugs*

As mentioned in Section 3.5.2, bugs may be preenumerated in a program as a classification, as in PROUST and WHY, or they may be generated. This generative process may be a simulation of how the student learned, as in STEP THEORY, or it may be constructed from the general model and inference procedure, as in PIXIE and ACM. The bugs might also be constructed by composing primitive bugs, as in BUGGY. The constructive process is further described in Section 7.

An instructional program's problem-solving model may also take the form of a classification model, as in FLOW and MENO. These programs cannot model the programming process; they can only evaluate the final code, matching it against patterns. Of course, it is much more difficult to model the programming process, and we should not expect that the methods that apply

in subtraction or algebra for representing the inference process and generating bugs will be adequate in this domain.

Classification models are useful for modeling inference when it is difficult to otherwise construct an interpretation of what the student is doing from primitive operators or when primitives might be difficult to define, as in programming and diagnosis. A compromise approach is to use a simulation model of the inference process that does not attempt to model human reasoning and instead just replicates the planning or "proof" structure of the situation-specific model, as in PROUST and the MACSYMA ADVISOR. A classification of frequently occurring bugs can then be used to account for inability to construct a consistent model, as in PROUST. In PROUST, bugs are classified abstractly, according to general patterns; for example, there are general rules for recognizing when goals for filtering data have been combined incorrectly.

It is worth repeating that the general model may be a classification model, as in NEOMYCIN, even though the inference process is simulated. Classification is always limited by restricting the situations that can be adequately modeled, but it may be the only approach possible for some domains.

6.8. *Arguments for Functional Models*

As previously stated, it is advantageous for the problem-solving model in an instructional program to be a functional qualitative model. The alternative approach is to express all knowledge as domain-specific situation/action patterns. As for the classification/simulation distinction, the advantages of the functional approach lie in different aspects of generality:

- **Explanation of reasoning:**

  Because the general model is not composed in situation-action patterns, it is possible to articulate it and the inference procedure explicitly (Clancey 1984a). That is, the program can say what is true and what to do in general, rather than being limited to saying what information to gather and what conclusions to make (as in GUIDON).

- **Robust engineering:**

  In a behavioral model, all situations must be specifically anticipated. As a result, combinatorial problems arise and there is no assurance of completeness. The possibility of interacting processes (e.g. nested loops in a program; interacting malfunctions in a circuit) makes it practically impossible to recognize every problem in terms of previously known patterns (e.g. program or disease schemas). With the inference procedure represented separately, a program can interpret the general model selectively and have some assurance that model construction goals are considered and accomplished in general.[9]

- **Modeling misconceptions:**

  By this separation, it is possible to construct a student model that makes the same distinctions. In particular, if one makes certain assumptions about the inference procedure one can reconstruct the student's general model and thus detect gaps and formulate misconceptions. An interaction of misconceptions and procedural differences might also be reconstructed, though the computational problems of doing this have not been explored.

It is well-established that first-order diagnostic performance is improved by use of a qualitative simulation model (Brown et al 1982a, Genesereth 1982b, Davis et al 1982). Just as programs like NEOMYCIN are limited in not having a functional model of the body, cognitive diagnosis systems are limited if they operate upon a behavioral model of reasoning. However, the difficulties for cognitive diagnosis are more severe because a behavioral cognitive model typically composes an inference procedure with a behavioral general model (refer to Figure 13). For example, MYCIN is a behavioral model of reasoning, with a behavioral model of physical processes embedded in it. Both cognitive and physical processes are modeled, but not functionally. This would be analogous to stating the subtraction model in terms of specific patterns of numbers—e.g. "If subtracting 9 from 5 and the top column to the left is a 2, then replace the 2 by 1 and write a 1 next to the 5...." The models of the LISP TUTOR and GEOMETRY TUTOR are similar. A student-modeling program based on a behavioral model can generate only behavioral explanations (e.g. "the student wrote down 1 next to the 2 because he was subtracting 9 from 5," etc); it is unable to distinguish factual errors from inference procedure errors. Even if a behavioral cognitive model had a functional model of the subject system embedded within it (which is unlikely for nonformal domains of any size), the instructional program would not be able to articulate student errors in terms of systematic errors in the general model.

For example, the GEOMETRY tutor cannot detect if a student always works backwards for the theorem to be proved. This inference principle is implicit in the program's specific rules for using geometry theorems. Because the principle is not independently represented, the program does not know when it is being applied either by the ideal model or the student. Also, the program cannot articulate the principle in an explanation to the student.

6.9. *Overlay vs Bug Models*

Typically in the literature one will see a distinction between an overlay model and a bug model. An overlay model relates student behavior to the internal problem-solving model; the student model is a subset of the idealized model ("overlay") and only notes missing or inappropriately used knowledge. In contrast, a student model incorporating bugs describes incorrect knowledge or reasoning (either by a "generative" construction as in the MACSYMA ADVISOR or by a bug library as in WHY).

This distinction is misleading because it suggests that the method of reconstructing reasoning by overlaying cannot be applied to infer student bugs. By overlaying a functional model, a program can reconstruct the student's model manipulation subgoals to detect gaps between what the student did and the general model.

For example, a pattern of inference might suggest that the student is trying to manipulate the model in some way, such as to test a hypothesis. However, according to the program's general model, his specific data request might not be relevant to the hypothesis he is testing. This inconsistency can be reformulated as facts that the student believes to be true—i.e. misconceptions. This is how the MACSYMA ADVISOR reconstructs the student's general model. The reasoning of BUGGY is similar when it explains an error in terms of a "math table bug" (these are not preenumerated but are inferred from the "gaps" between the reconstructed procedure and the student's solution). No similar reconstruction was possible in GUIDON because the cognitive model was behavioral and did not make a distinction between the domain model and inference procedure. The idea that the overlay method has nothing to do with bugs probably came about because the first overlay models were either behavioral (GUIDON) or did not explicitly represent the inference procedure (WUSOR and WEST) so that it could be reasoned about to infer errors in the general model.

There is also a tendency in the community to believe that modeling work in formal systems (subtraction, algebra) is more advanced because it isolates student inference procedure bugs. On the face of it, it appears inappropriate to base a student model on an "expert" inference procedure (such as NEOMYCIN). As I have discussed, this analogy fails to distinguish between a syntactic, algorithmic inference procedure and one that is unformalized and apparently subconscious, based on processes of memory. Errors in reasoning about physical systems seem to be predominantly factual (not "procedural bugs"). Isolating misconceptions by reconstruction from the inference procedure (as in the MACSYMA ADVISOR) is possible, but it requires substantial research just to identify what this procedure might be (Johnson & Soloway 1984, Clancey 1984b). It is not preformalized, as in subtraction, or easily constructed, as in algebra (Sleeman 1982) or symbolic integration.

The real work begins in explaining the origin of a misconception in terms of the background model that supports it (as in WHY) and what inference or learning process (e.g. false analogy, over-generalization) led to this model discrepancy (as in PROUST, summarized in Figure 7, or in REPAIR THEORY). While work in formal systems offers some general mechanisms for the production of misconceptions, their methods of enumeration (BUGGY), extraction of syntactic patterns (PIXIE), and generation from problem-space reduction (ACM) are inadequate. Bugs in BUGGY, ACM, the MACSYMA ADVISOR, PROUST, PIXIE, etc are either preenumerated or syntactic variations and combinations of formal operators. Misconceptions about physical models are unbounded in scope; they cannot be exhaustively preenumerated or assembled from a fixed set of primitive concepts.

The use of alternative models in teaching and student reasoning, particularly analogical reasoning, has been of great interest (Gentner & Stevens 1983a, Stevens & Collins 1978, Bott 1979). In WEST, the inability to consistently model the student (called cognitive tear), leads the program to try alternative inference procedures that are based on different strategies for playing the game (e.g. landing on squares that cause the attractive display to react). In PROUST, canned remediation text attempts to correct alternative general models that may have produced the coding error (e.g. confusing READ with a declaration).

The methods of syntactic variation used in formal systems might be extendable to modeling misconceptions about physical processes, not as specific misconceptions but as patterns in understanding processes in general. This is similar to the concepts of object primitives (Lehnert 1978) and fault models (Davis 1983). In general, reconstructive or explanation-based learning is a rapidly growing area of interest in AI (Kolodner & Simpson 1984, Mitchell et al 1986).

6.10. *Pragmatic Considerations*

I have distinguished between different kinds of process models of reasoning: classification vs simulation, behavioral simulation vs functional simulation, and different types of functional models (Figure 12). What kind of model is useful in different domains and how detailed a model must be for explanation and instruction are independent issues. For example, it may turn out that most computer programming errors are misconceptions about programming operators or procedures ("schemas") for relating operators. The order in which a student wrote the program may be irrelevant. Or, as assumed by the MACSYMA ADVISOR, the order may be the same (and correct) for the vast majority of students; errors are in the general model, not how it is used. On the other hand, information about what the student is trying to do (e.g. what diagnostic

hypothesis he is trying to confirm) may be useful for providing assistance in the context of solving a problem.

A complicating factor is that people do not always make conscious decisions about what they are trying to do. Or they may write down a solution automatically, without having to go through intermediate steps. Thus one student might be trying to use the fact that "M is a midpoint in segment AB," while a second immediately realizes that two triangles are congruent. Practiced reasoning might even be nondeterministic, consistent with several inference procedure principles, but not causally brought about by any one in particular (Brown & Newman 1985, Suchman 1985).

Finally, in saying that the function of the cognitive system is model construction and use, we may find it necessary to explain functional failures in terms of structural faults in the physical components in the brain (or computer) itself. However, based pragmatically on the population of students, cognitive models in today's instructional programs do not consider structural faults. Indeed, considering the nature of electronic diagnosis, it is a fascinating reflection on the nature of the mind that cognitive modeling is so little concerned with the structural level. References to memory and attention resources have been the exception (Ohlsson & Langley 1985, VanLehn 1983a).

7. CONSTRUCTING SITUATION-SPECIFIC PROCESS MODELS

All process modeling involves some kind of reconstruction. The assessments of student knowledge and reasoning described in Section 4 can be logically ordered from observations of student behavior to complex constructions of incorrect reasoning procedures. Here I briefly review how these inferences are made.

A modeling program generally works backwards from student behavior to construct a situation-specific model that may include

1. Elements of the program's domain model evident in student behavior (e.g. formal operators as in WEST);
2. Unarticulated domain-specific inferences (e.g. as in the hypothesis evaluation of GUIDON and WHY, and the move interpretation of WUSOR);
3. Unarticulated domain and inference procedure subgoals (e.g. programming intentions in PROUST, diagnostic tasks in IMAGE/NEOMYCIN, program methods in the LISP TUTOR);
4. Missing or inappropriate inferences and subgoals (e.g. incorrect implementation of programming intentions (deriving from misunderstanding operators in the programming language), wrong disease hypotheses, incorrect interpretation of circuit malfunctions, math table bugs, misstatements of geometry theorems);
5. Incorrect procedures for accomplishing reasoning subgoals (e.g. incorrect subtraction subprocedures, inefficient diagnostic reasoning, ineffective theorem proving).[10]

There is a logical progression here, proceeding from evaluation of actions to accounting for them in terms of the hypothesized situation-specific model of the student, current and past domain-specific goals, and inference procedures for accomplishing goals. Obviously, direct questioning and student-stated or observed partial solutions are an immediate source of information. Other assessments are made as follows:

- An idealized executable model (often called the "expert model") generates problem-solving behavior that is compared to the student's behavior.
- Differences in action are detected, producing a list of "appropriate," "omitted," and "inappropriate" behaviors. Those aspects of the model and inference procedure (perhaps combined, as in GUIDON, the LISP and GEOMETRY tutors), used by the expert to derive the "appropriate" and "omitted" behaviors, are marked as being "used" and "not used" by the student (Burton & Brown 1979, Goldstein, 1977, Clancey 1979b). This is called an overlay (Carr & Goldstein 1977) or differential (Burton & Brown 1979) model.
- Applying the accumulation of these markings, the subset of the expert model that is believed to be known (or typically used) by the student is run to produce a set of expectations of student actions. Discrepancies between these expectations and actual student behavior (surprises) drive the instructional interaction (e.g. ask the student to explain a surprising correct problem-solving step).

If the general model and inference procedure are separate, the program can proceed to explain inappropriate and omitted behavior by several means:

- **simulate variations of the inference procedure :**
  - an alternative (pre-specified) inference procedure, based on different problem-solving goals (WEST);[11]
  - switching in alternative buggy subprocedures from a library (BUGGY);
  - constructing a procedure from primitive operators by exhaustive search (LMS) or by discrimination learning (ACM);[12]
  - syntactic variations, by reordering or deleting subprocedure statements using a mixture of top-down prediction and bottom-up interpretation (IMAGE/NEOMYCIN);

- syntactic variations, by hypothesizing operators that bridge the gaps in a reconstruction [PIXIE (Sleeman 1983, Sleeman 1984b)].

- **simulate variations in the program's general (domain) model:**
  - variations of the general model that are simpler or coarser, taken from a library (WUSOR);
  - syntactic variations or transformations of the general model, supported by a theorem-prover approach for testing equivalence to the ideal model (Murray 1985);
  - inferring general model propositions consistent with the inference procedure and situation-specific model [MACSYMA ADVISOR (Genesereth 1982a)].

- **derivation of the inference procedure** in terms of constraints on model construction and use, to simulate/explain how the student copes with impasses [inability to accomplish subgoals because of an inadequate domain model (missing facts) or missing inference subprocedures (not knowing specifically what to do)] (REPAIR THEORY).

Methods for dealing with noise (Burton & Brown 1979), "coercing" partial matches (Burton 1982), and explaining "migration" of behavior (Brown & VanLehn 1980) constitute an important subarea of research (Ohlsson & Langley 1985).

8. CONCLUSIONS

I conclude by recapitulating the main arguments, summarizing the methodological lessons and the state of the art, and commenting on trends in the field.

8.1. *Recapitulation*

Surveying and evaluating a variety of existing instructional programs, I have stressed the following points in this review:

- **AI programming methodology provides new means to model processes** of physical systems, problem-solving procedures, how learning takes place, and methods and procedures for communication. In contrast with traditional CAI programs, these models are executable, so they can be used to solve the same problems presented to the student. They are also primarily qualitative and describe agents and processes by their causal, spatial, and temporal interaction. When subject matter is represented in this form, the internal model is called a "glass box" expert; it can both solve problems and be described in explanations by the teaching procedures.

- **A general model can be separated from the inference procedure by which it is applied to specific problems.** This is termed a functional model, in contrast with a behavioral model, which describes problem-solving behavior in a situation-specific way. Functional models are represented in terms of hierarchical composition of functional operators and/or structural components. Behavioral models are expressed as links between system states in a causal-associational network.

- **Both behavioral and functional models can be used to simulate physical and reasoning processes.** However, a functional model characterizes purposes and goals abstracted from particular situations. For example, in a functional model of reasoning, these purposes are described in terms of general operators for manipulating a situation-specific model.

- **A given functional model can be used in multiple ways.** In an instructional program, this allows a single encoding of the general model to be interpreted for evaluating student performance, communicating with him, and modeling his learning behavior. At the same time, this separation forces the problem-solving, communication, and learning models to be represented explicitly, in a well-structured way, enabling the formalized models to be properly communicated and shared in the scientific literature.

- **Problem domains differ in terms of the functionality of the system and the nature of the inference procedure.**
  - Formal or artifactual domain operators: Is functionality defined, rather than being inferred from other domain models?
  - An algorithmic inference procedure: Is the procedure explicitly taught, rather than involving predominantly automatic processes of memory and attention?
  - A written calculus: Can the problem-solving steps for constructing a situation-specific model be written down as a sequence of syntactic transformations? Or does the procedure involve predominantly mental operations or partial solution descriptions from multiple perspectives?

  Domains cut across these dimensions in different ways. For example, programming involves formal domain operators, but it has a nonalgorithmic inference procedure and an incomplete written calculus. Generalization of student modeling methods must take these dimensions into account. Formal models might be exploited in nonformal domains (e.g. Figure 10) as a simplification.

In applying modeling methods developed for mathematics to other domains, we face a number of difficulties:

- The functionality of the domain may be indeterminant, so misconceptions cannot automatically be generated or anticipated.

- The inference procedure may be heuristic and even unknown by human teachers; formalizing it explicitly so that it may be reasoned about may be beyond AI representational capabilities.
- Problem solving may be mostly mental, without a written calculus by which the student's situation-specific model may be observed and critiqued.

On the positive side, we have found that it is possible to understand different domains using common terminology; the ideas of a procedural bug, domain operator, and inference procedure, among others, carry over for relating concepts and modeling techniques in different domains.

8.2. *Methodology*

The research reviewed here has been remarkably successful in uncovering interesting representational problems, reconceptualizing AI results (in the areas of "expert systems," natural language, learning, knowledge representation), and formulating new computational methods. It is worthwhile to reflect on the methodology that has contributed to this progress. A number of patterns can be discerned:

- By constructing artifacts designed to exhibit intelligent behavior, we are proceeding experimentally and enhancing our opportunity to learn from failures. This is an effective heuristic for focusing research and effectively formulating good theoretical questions, an approach AI shares with traditional engineering (Petroski 1985).
- By formulating our domain and communication knowledge in programs in well-structured representations, we are constructing "first-draft" models that can be studied and reformulated. This idea of using knowledge representations to detect patterns, articulate principles, and improve the representations has proved extremely powerful (Clancey 1983b). Particularly when we are trying to model what experts know only tacitly, this provides a means for abstracting domain-specific statements about "what to do," allows us to separately state the general model (what is true about a system in general) and the inference procedure (how to construct and use a situation-specific model).
- By constructing complicated programs that must solve problems, learn, and communicate, we are providing a holistic test for our theories. In order to construct general programs, we apply our methods in different domains and to represent different kinds of processes. A systematic set of techniques, including classification and simulation approaches, is used to qualitatively model processes in the physical world and in inference, learning, and communication.

In summary, our approach is to tease apart interacting principles of representation, inference, modeling, and pedagogy by experimentally constructing general programs (Clancey 1982a). This AI research is distinguished from "knowledge engineering" and traditional computer-aided instruction by its use of programming as a vehicle for moving forward to more abstract (general) levels of qualitative model interpretation—in which models of processes are articulated in terms of interacting causal, spatial, temporal, mathematical, and social constraints.

8.3. *State of the Art*

Today's instructional programs, those that attempt to realize adaptive instruction in a general way, are among the largest and most complicated computer programs in the field of artificial intelligence.

Reviewing the requirements for qualitative models of problem-solving listed in Section 3.4.2:

- Experiments have verified the accuracy of some student models (Brown 1978, VanLehn 1984, Anderson et al 1985); results in the study of mental models, spawned by this research, have not been generally exploited in the design of current programs.
- There has been significant progress in explanation of the inference procedure (in SOPHIE-III, XPLAIN and NEOMYCIN) and minor attempts to tailor explanations to a listener's expectations and model (e.g. GUIDON, XPLAIN, and WUSOR). Results in natural language research far exceed the capabilities of current instructional programs. However, the generality of these methods has not been tested on knowledge bases of the complexity found in instructional programs.
- We have made significant progress modeling learning in formal domains [e.g. subtraction (SIERRA), geometry (GEOMETRY TUTOR), and abstract computer programming (Anderson et al 1984a)]. Otherwise, models of learning go little beyond classification of misconceptions and prerequisite ordering.

Some programs have been well-debugged, carefully engineered for ease of use, and tested with students (e.g. WEST, SOPHIE-I and II, BUGGY, REPAIR THEORY). Most programs exist only as experimental prototypes, perhaps taking years to construct, and are not intended for general use. Given the methodology outlined above, routine use of a program in a classroom today is not the only measure of success; designing a new language and implementing a program and testing it with a small number of cases may reveal enough deficiencies to keep a researcher busy for several years. Thus,

there are several stages of model validation: plausibility (can the program perform at all, established by a small number of cases) and completeness (does the program explain and cope with real-world behavior, established by a large number of empirical trials with students).

Regarding progress in explanation and learning, research is slowly proceeding from simply stating procedures to modeling how they can be constructed automatically. It is on this basis that models for learning procedures will evolve. Today, it is difficult enough to state heuristic inference procedures in some computational language (Clancey 1984b, Johnson & Soloway 1984, Genesereth 1982b). It is no surprise that modeling research has assumed that the student has the same inference procedure as the ideal model or some simple variation of it (composition, deletion, reordering).

Current instructional programs can neither articulate the rationale for abstract procedures (except on an ad hoc basis by canned text) nor understand the rationale for alternative designs. That is, even the current ideal model in today's instructional programs contains much implicit information, perhaps even unarticulated by the researchers themselves (Section 6.3) (Clancey 1984b, Neches et al 1985, Brown et al 1982b).

8.4. *Trends*

During the past decade, there have been several significant shifts in the field. First, in the transition from CAI to AI-based systems, emphasis shifted from constructing a curriculum of exercises (e.g. Barr, 1976) to representing the knowledge necessary to solve a restricted set of problems. The systems of the period from 1975 to 1980 were complex, with modeling, explanation, and problem generation facilities (WEST, SOPHIE-II, WUSOR, GUIDON). The second shift focused on the modeling of errors (in WHY, BUGGY, MENO) and their cause (Matz 1981, Brown & VanLehn 1980), and the construction of better models of knowledge and reasoning (in SOPHIE-III, NEOMYCIN, PROUST, and STEAMER). Finally, in the 1980s progress in knowledge representation and the involvement of researchers in cognitive psychology has brought about a renewed interest in pedagogy (Anderson et al 1984b) and discourse (Woolf & McDonald 1984).

Research on representation of qualitative models has progressed far enough to allow attention to return to construction of explanation and teaching programs. For example, we could exploit what we have learned about representing processes to decompose communication models in a way that makes discourse assumptions more explicit. First, improved representation languages now permit functional representations (as in the shift between GUIDON and the MENO-TUTOR). Second, there is a synergy between student modeling and communication modeling research: Better articulated

student models provide the opportunity for and demand finer-grained statements of what a teaching interaction is supposed to do.

At this point, pragmatic issues may begin to dominate. The programs must work consistently, students must be motivated to use them, and the interface must be carefully (and tediously) engineered. Constructing a good graphics interface may take longer than designing the underlying representation (Richer & Clancey 1985). For these reasons, it may be another five years before questions about completeness, usefulness, and reliability of AI-based instructional programs are even meaningful. Increasingly, there may be a shift from scientific issues of problem solving and learning to human-engineering issues. (How many buttons should be on mouse? How large should the screen be?) Without a convenient interface, the student modeling capability of the program may be lost. Just the idea of introducing new notations for solving problems (e.g. Figure 10) poses conceptual difficulties for communicating with students. For this reason, current research should be viewed as exploratory.

It is worth mentioning again that all of the modeling methods in existing programs can detect non-ideal behavior. The issue is whether they can explain it. While I have argued that a functional process model has certain advantages for explanation and modeling, particularly for developing a theory of problem solving and instruction, it is possible that practical instructional programs can be constructed that only indicate to a student that his behavior is inappropriate and state the correct problem-solving step (e.g. "use a CONS here"; "ask about fever"). Intuitively, some researchers believe that automatically inferring a student's misconception, especially in physical domains, is more than good teachers can do (Ohlsson & Langley 1985). In AI, the danger of expecting programs to do what people cannot generally do is called the "superhuman human fallacy" (Minsky 1982). Asking the student to explain what he is doing may be a more practical approach.

While we may find that there are conceptual barriers to accomplishing some of our goals for adaptive instruction, there is good reason to believe that useful instructional programs will be produced from this line of research. The confluence of learning, memory, and problem-solving research is a most exciting development; however, its pragmatic and even philosophic effects are generally yet to be realized in the design of instructional programs. In this rapidly changing field, there is reason to expect dramatic changes in future research goals and assumptions, concerning what instructional programs are in theory capable of doing and what kinds of aids are believed to be worth constructing (Rouse & Morris 1985).

It is now clear that representation, learning, and discourse, or any subarea of AI, cannot be studied in isolation. Instructional programs may provide the

strongest foundation for advances on all fronts, for they are designed on the very principle of solving problems by multiple models, learning from experience, and communicating to share problem-solving models. It is this synergy that may hold the greatest promise for uncovering the nature of intelligence, perhaps now best defined as the capability to acquire, use, and communicate qualitative models.

BRIEF GUIDE TO THE LITERATURE

The general reader might start with the chapter on educational applications in the *Handbook of Artificial Intelligence* (Clancey 1982b). See Sleeman & Brown 1982 for a description of more programs in greater detail; this book is the single most important technical reference in the field. Wenger's recent survey (Wenger 1986) provides a good over-all synthesis of the goals and methods of AI-based instructional programs. Other books, mostly about AI-based instructional programs, are Gentner & Stevens 1983a, Klahr 1976, Bobrow & Collins 1975, O'Shea & Self 1983.

Articles in the *Proceedings of Cognitive Science* and the *Joint Conference on Artificial Intelligence* are short and useful—for example, see Clancey 1979a, Clancey & Letsinger 1984, London & Clancey 1982, Woolf & McDonald 1984, Johnson & Soloway 1984, Anderson et al 1985. Excellent articles are published in *Cognitive Science*—e.g. Sleeman 1984b, Young & O'Shea 1981, Brown 1978, Brown & VanLehn 1980, Anderson et al 1984a. Relevant theses include Clancey 1979c, VanLehn 1983b, Johnson 1985, and de Kleer 1979. The best introduction to functional qualitative models of physical systems is Bobrow 1984. Three extremely useful surveys of related cognitive science ideas are Rumelhart & Norman 1983, Greeno & Simon 1984, Rouse & Morris 1985.

ACKNOWLEDGMENTS

I would like to thank the readers who added to the completeness and accuracy of this review: John Anderson, John Seely Brown, Lewis Johnson, Pat Langley, John Self, Derek Sleeman, Etienne Wenger, and Richard Young. Omissions and errors are of course my responsibility. I especially thank my colleagues at Stanford, Steve Barnhouse, Ted Crovello, David Leserman, Bob London, Mark Richer, Naomi Rodolitz, and David C. Wilkins, for their comments on the first draft. I also thank Etienne Wenger for sharing his own review (Wenger 1986) in manuscript form. Barbara Tall provided essential secretarial support.

On behalf of everyone participating in this research, I would like to thank the Office of Naval Research, Army Research Institute, and Advanced Research Projects Agency for funding this work and playing such a strong role in organizing and developing our community. In particular, we thank Susan Chipman, Marshall Farr, Henry Halff, Harry O'Neil, Joe Psotka, and Martin Tolcott for their support and encouragement. Research facilities have been provided by several universities; the support of Xerox-PARC and Bolt-Beranek and Newman is particularly noteworthy.

Funding for my own research has been provided in part by ONR Contract N00014-79C-0302 and the Josiah Macy Jr. Foundation (Grant B852005). Computational facilities have been provided by the SUMEX-AIM facility (NIH Grant RR00785).

FOOTNOTES

[1] The term "inference engine" (Davis 1986) often refers to a simple rule interpreter, in which the "knowledge base" that it interprets is a program that combines the domain model and inference procedure. This is the relation beween MYCIN's rules and rule interpreter. The trend is to identify the domain model with the knowledge base and to view the inference procedure as something much more complicated than a rule interpreter.

[2] The ABEL program (Patil 1981) attempts to prove the validity of causal associations in particular cases by using "deeper" or more detailed reasoning to determine resultant effects of interacting disease processes.

[3] Extension of this idea to computer programming poses additional problems. The written notation (the code) only partially expresses the situation-specific model and omits goal interactions and levels of abstraction (Johnson & Soloway 1984). Moreover, the refinement operator of the inference procedure for programming requires a notation for showing composition and specialization of partial process descriptions. See VanLehn & Brown 1979 and Abelson et al 1985 for examples. See also Rouse & Morris 1985 for a related discussion in terms of "implicit vs explicit model manipulation" and "behavioral discretion." My "written notation" and "algorithmic vs heuristic" dimensions are similar but draw the lines more sharply.

[4] In contrast with electronic circuit diagnosis, computer system diagnosis includes problems caused by software and the operating environment.

[5] See Kosslyn 1980 for a related discussion about levels of scientific explanation.

[6] Nevertheless, the tutor can articulate this process, through the use of ad hoc "planning rules" that recognize particular situations and offer strategic advice. Thus, the issue is how the model is expressed in the program—what is separate and explicit—not whether it is present at all.

[7] RELATIONS appear in all caps; ground terms—literals—are in lower case; VARIABLES are preceded by a dollar sign. *Translations* are in italics.

[8] Note that the term "simulation model" might also be applied to a program that is based on a quantitative general model, such as numeric equations describing a circuit's behavior. Also, a classification model might be expressed as a numeric function (Nilsson 1965). Thus the classification/simulation distinction is orthogonal to the qualitative/quantitative distinction.

[9] The problem of generality recurs at this level, of course. This is one reason researchers are attempting to abstract to the constraints underlying the inference procedure (Section 6.3).

[10] An incorrect programming process should not be confused with bugs in computer programs that follow from misconceptions about program operators. In early work (Miller & Goldstein 1976, Miller 1977), the functionality of the inference procedure is not separated from the functionality of the computer program. Thus, a hierarchy of "operators" includes both "iterate" (what the program does) and "decompose" (what the programmer does in writing the program, a model-manipulation operator).

[11] More precisely, in WEST the spinners constitute a subsystem whose desired behavior is determined by the desired behavior the student ascribes to the game board (e.g. bumping his opponent). This "external constraint" is analogous to the social interaction constraints of diagnosis. Thus, when we refer to the student's strategy in WEST we should distinguish between the interacting constraints of the spinner and board systems and the inference procedure the player applies to satisfy these constraints. Generalizing WEST or building upon its general design would require careful articulation of the concept of "strategy" along these lines.

[12] Here it is important not to confuse diagnostic modeling— inferring the student's procedure from a trace of his solutions, as in LMS and ACM—with modeling the human learning process. The former is concerned with completeness, efficiency, and generality (domain-independence); the latter with producing an explanatory accounting in terms of psychological assumptions and the instructional sequence. The ACT* and SIERRA models are attempts to satisfy both goals (see discussion in Langley et al 1984).

Literature Cited

Abelson, H., Sussman, G. J., Sussman, J. 1985. *Structure and Interpretation of Computer Programs*. Cambridge, Mass: MIT Press

Achinstein, P. 1983. *The Nature of Explanation*. New York: Oxford Univ. Press

Adelson, B. 1984. When novices surpass experts: the difficulty of a task may increase with expertise. *J. Exp. Psychol.: Learning, Memory, and Cognition* 10:483–95

Anderson, J. R., Boyle, C. F., Corbett, A., Lewis, M. 1986. Cognitive modelling and intelligent tutoring. *Interim ONR-86-1, Carnegie-Mellon Univ., Pittsburgh, Pa.*

Anderson, J. R., Boyle, C. F., Farrell, R., Reiser, B. 1984b. Cognitive principles in the design of computer tutors. In *Proc. 6th Ann. Conf. Cognitive Sci. Soc., Boulder, Colo.*, pp. 2–10

Anderson, J. R., Boyle, C. F., Yost, G. 1985. The geometry tutor. In *Proc. 9th Int. Jt. Conf. Artif. Intell., Los Angeles,* 1:1–7

Anderson, J. R., Farrell, R., Sauers, R. 1984a. Learning to program in LISP. *Cognitive Sci.* 8(2):87–129

Anderson, J. R., Greeno, J. G., Kline, P. J., Neves, D. M. 1981. Acquisition of problem-solving skill. In *Cognitive Skills and their Acquisition,* ed. J. R. Anderson, pp. 191–230. Hillsdale, NJ: L. Erlbaum

Appelt, D. E. 1982. Planning natural-language utterances to satisfy multiple goals. *Tech. Note 259, SRI Int., Menlo Park, Calif.*

Atkinson, R. C. 1972. Ingredients for a theory of instruction. *Am. Psychol.* 27:921–31

Atkinson, R. C., Wilson, H. A. eds. 1969. *Computer-Assisted Instruction*. New York: Academic

Barr, A. 1979. Meta-knowledge and cognition. *Proc. 6th Int. Jt. Conf. Artif. Intell., Tokyo,* pp. 31–33

Barr, A., Atkinson, R. C. 1975. Adaptive instructional strategies. Presented at *IPN Symp. 7: Formalized Theories of Thinking and Learning and their Implications for Science Instruction*

Barr, A., Beard, M., Atkinson, R. C. 1976. The computer as a tutorial laboratory: the Stanford BIP Project. *Int. J. Man-Mach. Stud.* 8:567–96

Barr, A., Bennett, J., Clancey, W. 1979. Transfer of expertise: a theme for AI research. *Work. Pap. HPP-79-11, Stanford Univ., Calif.*

Bobrow, D. G. 1984. Special issue on qualitative reasoning about physical systems. *Artif. Intell.* 24(1–3)

Bobrow, D. G., Collins, A., eds. 1975. *Representation and Understanding: Studies in Cognitive Science*. New York: Academic

Bott, R. A. 1979. A study in complex learning: theory and methodologies. *Tech. Rep. 7901, Cent. for Human Inf. Process., Univ. Calif., San Diego*

Brady, M., Berwick, R. C., eds. 1983. *Computational Models of Discourse*. Cambridge, Mass: MIT Press

Brown, J. S. 1977a. Remarks on building expert systems (Reports of panel on applications of artificial intelligence). In *Proc. 5th Int. Jt. Conf. Artif. Intell., Cambridge, Mass,* pp. 994–1005

Brown, J. S. 1977b. Uses of AI and advanced computer technology in education. In *Computers and Communications: Implications for Education*. New York: Academic

Brown, J. S. 1983. Process versus product—a perspective on tools for communal and informal electronic learning. In *Education in the Electronic Age, Proc. Conf. sponsored by Educ. Broadcast. Corp., WNET/Thirteen*

Brown, J. S., Burton, R. B. 1978. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Sci.* 2(2):155–92

Brown, J. S., Burton, R. R., Bell, A. G. 1974. Sophie: a sophisticated instruction environment for teaching electronic troubleshooting (an example of AI in CAI). *BBN Tech. Rep. 2790, Bolt, Beranek & Newman, Cambridge, Mass.*

Brown, J. S., Burton, R. R., de Kleer, J. 1982a. Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III. See Sleeman & Brown 1982, pp. 227–82. London: Academic

Brown, J. S., Burton, R. R., Zydbel, F. 1973. A model-driven question-answering system for mixed-initiative computer-assisted instruction. *IEEE Trans. Syst. Man Cybern.* SMC3(3):248–57

Brown, J. S., Collins, A., Harris, G. 1977. Artificial intelligence and learning strategies. In *Learning Strategies,* ed. H. O'Neill. New York: Academic

Brown, J. S., Goldstein, I. P. 1977. Computers in a learning society. Testimony for the House Science and Technology Subcommittee on Domestic and International Planning, Analysis, and Cooperation

Brown, J. S., Moran, T. P., Williams, M. D. 1982b. The semantics of procedures: a cognitive basis for training procedural skills for complex system maintenance. Palo Alto, Calif: *Xerox Corp., CIS Work. Pap.*

Brown, J. S., Newman, S. E. 1985. Issues in cognitive and social ergonomics: from our house to Bauhaus. *J. Human-Computer Interact.* 1(4):359–91

Brown, J. S., Rubenstein, R., Burton, R. 1976. Reactive learning environment for computer-aided electronics instruction. *BBN Tech. Rep. 3314, Bolt, Beranek & Newman, Cambridge, Mass.*

Brown, J. S., VanLehn, K. 1980. Repair theory: a generative theory of bugs in procedural skills. *Cognitive Sci.* 4(4):379–415

Buchanan, B. G., Sutherland, G., Feigenbaum, E. A. 1969. Heuristic Dendral: a program for generating explanatory hypotheses in organic chemistry. In *Machine Intelligence,* ed. B. Meltzer, D. Michie, pp. 209–54. Edinburgh: Edinburgh Univ. Press

Burton, R. R. 1976. Semantic grammar: an engineering technique for constructing natural language understanding systems. *BBN Rep. No. 3453, Bolt, Beranek & Newman, Cambridge, Mass.*

Burton, R. R. 1982. Diagnosing bugs in a simple procedural skill. See Sleeman & Brown 1982, pp. 157–83

Burton, R. R., Brown, J. S. 1979. An investigation of computer coaching for informal learning activities. *Int. J. Man-Mach. Stud.* 11:5–24. Reprinted in Sleeman & Brown 1982

Carbonell, J. R. 1970. Mixed-initiative man-computer instructional dialogues. *BBN Tech. Rep. 1971, Bolt, Beranek & Newman, Cambridge, Mass.*

Carbonell, J. R., Collins, A. M. 1973. Natural semantics in artificial intelligence. In *Proc. 3rd Int. Jt. Conf. Artif. Intell., Stanford, Calif.*, pp. 344–51

Carr, B., Goldstein, I. 1977. Overlays: a theory of modeling for computer aided instruction. *AI Memo 406, Artif. Intell. Lab., Mass. Inst. Technol., Cambridge*

CERL (Computer-based Education Research Laboratory). 1977. Demonstration of the PLATO IV computer-based education system. Univ. Ill., Urbana

Charniak, E., Riesbeck, C. K., McDermott, D. V. 1980. *Artificial Intelligence Programming*. Hillsdale, NJ: L. Erlbaum

Charniak, E., McDermott, D. 1985. *Introduction to Artificial Intelligence*. Reading, Mass: Addison-Wesley

Chi, M. T. H., Feltovich, P. J., Glaser, R. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Sci.* 5:121–52

Chi, M. T. H., Glaser, R. 1982. Final report: knowledge and skill differences in novices and experts. *Tech. Rep. UPITT/LRDC/ ONR/KBC-7, Learn. Res. Devel. Cent., Univ. Pittsburgh, Pa.*

Chi, M. T. H., Glaser, R., Farr, M., eds. 1986. *Contributions to the Nature of Expertise*. In preparation

Clancey, W. J. 1979a. Dialogue management for rule-based tutorials. In *Proc. 6th Int. Jt. Conf. Artif. Intell., Tokyo,* pp. 155–61

Clancey, W. J. 1979b. Tutoring rules for guiding a case method dialogue. *Int. J. Man-Mach. Stud.* 11:25–49. Republished in Sleeman & Brown 1982

Clancey, W. J. 1979c. *Transfer of rule-based expertise through a tutorial dialogue*. PhD thesis. Stanford Univ., Calif.

Clancey, W. J. 1982a. GUIDON. In *The Handbook of Artificial Intelligence,* ed. A. Barr, E. A. Feigenbaum, pp. 267–78. Los Altos, Calif: W. Kaufmann

Clancey, W. J. 1982b. Applications-oriented AI research: education. See Clancey 1982a, pp. 223–94

Clancey, W. J. 1983a. The advantages of abstract control knowledge in expert system design. In *Proc. Natl. Conf. Artif. Intell., Washington, DC,* pp. 74–78

Clancey, W. J. 1983b. The epistemology of a rule-based expert system: a framework for explanation. *Artif. Intell.* 20(3):215–51

Clancey, W. J. 1984a. Methodology for build-

ing an intelligent tutoring system. See Kintsch et al 1984, pp. 51–83
Clancey, W. J. 1984b. Acquiring, representing, and evaluating a competence model of diagnosis. *HPP Memo 84-2, Stanford Univ., Calif.* To appear in Chi et al 1986. Submitted for publication
Clancey, W. J. 1985a. Software tools for developing expert systems. In *Proc. Int. Conf. Artif. Intell. Med., Amsterdam,* pp. 155–78. Participants ed.
Clancey, W. J. 1985b. Heuristic classification. *Artif. Intell.* 27:289–350
Clancey, W. J. 1986a. Representing control knowledge as abstract tasks and metarules. In *Computer Expert Systems,* ed. M. J. Coombs, L. Bolc. New York: Springer-Verlag. In preparation
Clancey, W. J. 1986b. From Guidon to Neomycin and Heracles in twenty short lessons *ONR Final Rep. 1979–1985. KSL Tech. Rep. 86-11, Stanford Univ., Calif.*
Clancey, W. J. 1986c. The science and engineering of qualitative models. *KSL Tech. Rep. 86-27, Stanford Univ.*
Clancey, W. J., Letsinger, R. 1984. NEOMYCIN: reconfiguring a rule-based expert system for application to teaching. See Clancey & Shortliffe 1984, pp. 361–81
Clancey, W. J., Shortliffe, E. H., eds. 1984. *Readings in Medical Artificial Intelligence: The First Decade.* Reading, Mass: Addison-Wesley
Collins, A., Stevens, A. L. 1980. Goals and strategies of interactive teachers. *BBN Tech. Rep. 4345, Bolt, Beranek & Newman, Cambridge, Mass.*
Collins, A., Warnock, E. H., Aiello, N., Miller, M. L. 1975. Reasoning from incomplete knowledge. See Bobrow & Collins 1975, pp. 383–415
Crowder, N. A. 1962. Intrinsic and extrinsic programming. In *Proc. Conf. Appl. Digital Comput. Autom. Instruct., New York,* pp. 55–58
Davis, R. 1983. Diagnosis via causal reasoning: paths of interaction and the locality principle. In *Proc. Natl. Conf. Artif. Intell., Washington, DC,* pp. 88–94
Davis, R. 1986. Knowledge-based systems. *Science* 231:957–63
Davis, R., Buchanan, B., Shortliffe, E. H. 1977. Production rules as a representation for a knowledge-base consultation program. *J. Artif. Intell.* 8(1):15–45
Davis, R., Shrobe, H., Hamscher, W., Wieckert, K., Shirley, M., Polit, S. 1982. Diagnosis based on description of structure and function. In *Proc. Natl. Conf. Artif. Intell., Pittsburgh, Pa,* pp. 137–42
de Kleer, J. 1979. *Casual and teleological reasoning in circuit recognition.* PhD thesis. Mass. Inst. Technol., Cambridge. Also *Rep. No. AI-TR-529*
de Kleer, J. 1984. Choices without backtracking. In *Proc. Natl. Conf. Artif. Intell., Austin, Tex,* pp. 79–85
de Kleer, J., Brown, J. S. 1984. A qualitative physics based on confluences. *Artif. Intell.* 24(1–3):7–83
diSessa, A. A. 1984. The third revolution in computers and education. Rep. for Comm. Math., Sci. Technol. Educ., Comm. Behav. Soc. Sci. Educ., Natl. Acad. Sci.
Elstein, A. S., Shulman, L. S., Sprafka, S. A. 1978. *Medical Problem Solving: An Analysis of Clinical Reasoning.* Cambridge, Mass: Harvard Univ. Press
Feigenbaum, E. A. 1977. The art of artificial intelligence: I. Themes and case studies of knowledge engineering. In *Proc. 5th Int. Jt. Conf. Artif. Intell., Cambridge, Mass,* pp. 1014–29
Feltovich, P. J., Johnson, P. E., Moller, J. H., Swanson, D. B. 1984. The role and development of medical knowledge in diagnostic expertise. See Clancey & Shortliffe 1984, pp. 275–319
Fletcher, J. D. 1975. Modeling of learner in computer-assisted instruction. *J. Comput.-Based Instruct.* 1:118–26
Freud, S. 1965. *The Psychopathology of Everyday Life.* New York: W. W. Norton
Genesereth, M. R. 1982a. The role of plans in intelligent teaching systems. See Sleeman & Brown 1982, pp. 137–55
Genesereth, M. R. 1982b. Diagnosis using hierarchical design models. In *Proc. Natl. Conf. Artif. Intell., Pittsburgh, Pa.,* pp. 278–83
Genesereth, M. R. 1984. The use of design descriptions in automated diagnosis. *Artif. Intell.* 24(1–3):411–36
Gentner, D. R. 1977. The FLOW tutor: a schema-based tutorial system. In *Proc. 5th Int. Jt. Conf. Artif. Intell., Cambridge, Mass,* pp. 787
Gentner, D., Stevens, A., eds. 1983. *Mental Models.* Hillsdale, NJ: L. Erlbaum
Glaser, R. 1983. Education and thinking: the role of knowledge. *Tech. Rep. PDS-6, Learn. Res. Dev. Cent., Univ. Pittsburgh, Pa.*
Glaser, R. 1985. Thoughts on expertise. *Tech. Rep. 8, Learn. Res. Devel. Cent., Univ. Pittsburgh, Pa.*
Goldstein, I. P. 1977. The computer as coach: an athletic paradigm for intellectual education. *AI Memo 389, Artif. Intell. Lab., Mass. Inst. Technol., Cambridge*
Goldstein, I. 1978. Developing a computational representation for problem solving skills. In *Proc. Carnegie-Mellon Conf. Problem Solving and Education: Issues in Teaching and Research, Pittsburgh*
Goldstein, I. P. 1982. The genetic graph: a representation for the evolution of procedural knowledge. See Sleeman & Brown 1982, pp. 51–77. London: Academic
Goldstein, M., Goldstein, I. F. 1980. *How We Know: An Exploration of the Scientific Process.* New York: De Capo
Goldstein, I. P., Papert, S. 1977. Artificial intelligence, language and the study of knowledge. *Cognitive Sci.* 1(1):84–123
Greeno, J. G. 1980. Psychology of learning, 1960–1980: one participant's observations. *Tech. Rep. 5, Learn. Res. Dev. Cent., Univ. Pittsburgh, Pa.*
Greeno, J. G., Simon, H. A. 1984. Problem solving and reasoning. *UPITT/LRDC/ONR/APS 14, Univ. Pittsburgh, Pa.* To appear in *Stevens' Handbook of Experimental Psychology.* New York: J. Wiley. Revised Ed. Submitted for publication
Grosz, B. 1977. The representation and use of focus in a system for understanding dialogs. In *Proc. 5th Int. Jt. Conf. Artif. Intell., Cambridge, Mass.,* pp. 67–76
Hasling, D. W., Clancey, W. J., Rennels, G. R. 1984. Strategic explanations in consultation. *Int. J. Man-Mach. Stud.* 20(1):3–19. Republished 1984 in *Development in Expert Systems,* ed. M. J. Coombs. London: Academic
Hayes-Roth, B., Hayes-Roth, F. 1979. A cognitive model of planning. *Cognitive Sci.* 3:275–310
Hayes-Roth, F., Waterman, D., Lenat, D., eds. 1983. *Building Expert Systems.* New York: Addison-Wesley
Hollan, J. D., Hutchins, E. L., Weitzman, L. 1984. STEAMER: An interactive inspectable simulation-based training system. *Artif. Intell. Mag.* 5(2):15–27
Johnson, P. E., Durain, A. S., Hassebrock, F., Moller, J., Prietula, M., et al. 1981. Expertise and error in diagnostic reasoning. *Cognitive Sci.* 5:235–84
Johnson, W. L. 1985. *Intention-based diagnosis of errors in novice programs.* PhD thesis. Yale Univ., New Haven, Conn. Also *Rep. No. YALEU/CSD/RR#395*
Johnson, W. L., Soloway, E. 1984. Intention-based diagnosis of program errors. In *Proc. Natl. Conf. Artif. Intell., Austin, Tex.,* pp. 162–68
Kahn, G., Nowlan, S., McDermott, J. 1985. MORE: an intelligent knowledge acquisition tool. In *Proc. 9th Int. Jt. Conf. Artif. Intell., Los Angeles, Calif.,* pp. 581–84
Kieras, D. E. 1984. A simulation model for procedure inference from a mental mode for a simple device. *Tech. Rep. UARZ/DP/TR-84/ONR-15, Univ. Ariz., Tucson*
Kimball, R. B. 1973. Self-optimizing computer-assisted tutoring: theory and practice. *Tech. Rep. 206, Stanford Univ., Inst. Math. Studies in Soc. Sci.*
Kintsch, W., Miller, J. R., Polson, P. G., eds. 1984. *Method and Tactics in Cognitive Science.* Hillsdale, NJ: L. Erlbaum
Klahr, ed. 1976. *Cognition and Instruction.* Hillsdale, NJ: L. Erlbaum
Kolodner, J. L. 1982. The role of experience in development of expertise. In *Proc. Natl. Conf. Artif. Intell., Pittsburgh, Pa.,* pp. 273–77
Kolodner, J. L., Simpson, R. L. 1984. Experience and problem solving: a framework. In *Proc. 6th Ann. Conf. Cognitive Sci. Soc., Boulder, Colo.,* pp. 239–43
Konolige, K. 1984. *A deduction model of belief and its logics.* PhD thesis. Stanford Univ., Calif. Also *Rep. No. STAN-CS-84-1022*
Kosslyn, S. M. 1980. *Image and Mind.* Cambridge, Mass: Harvard Univ. Press
Laird, J. E., Rosenbloom, P. S., Newell, A. 1984. Towards chunking as a general learning mechanism. In *Proc. Natl. Conf. Artif. Intell., Austin, Tex,* pp. 188–92
Langley, P., Ohlsson, S., Sage, S. 1984. A machine learning approach to student modeling. *Tech. Rep. CMU-RI-TR-84-7, Robotics Inst., Carnegie-Mellon Univ., Pittsburgh, Pa.*
Larkin, J. H., McDermott, J., Simon, D. P., Simon, H. A. 1980. Models of competence in solving physics problems. *Cognitive Sci.* 4:317–48
Laubsch, J. H. 1975. Some thoughts about representing knowledge in instructional systems. In *Advance Pap. 4th Int. Jt. Conf. Artif. Intell., Tbilisi, Georgia, USSR,* 1: 122–25
Lehnert, W. G. 1978. Representing physical objects in memory. *Tech. Rep. 131, Comput. Sci. Dep., Yale Univ., New Haven, Conn.*
Lesgold, A. M. 1983. Acquiring expertise. *Technical Report UPITT/LRDC/ONR/PDS-5, Learn. Res. Dev. Cent., Univ. Pittsburgh, Pa.*
London, B., Clancey, W. J. 1982. Plan recognition strategies in student modeling: prediction and description. In *Proc. 2nd AAAI,* pp. 335–38
Matz, M. 1981. Towards a generative theory of high school algebra errors. See Sleeman & Brown 1982, pp. 25–50
Miller, M. L., 1980. Using plans to generate hints in an intelligent tutoring system for elementary graphics programming. *Tech. Rep. Proposal 16-R80, Intell. Interact. Syst. Branch, Texas Instrum.*

Miller, M. L., Goldstein, I. P. 1976. PAZATN: a linguistic approach to automatic analysis of elementary programming protocols. *AI Memo 388, Artif. Intell. Lab., Mass. Inst. Technol., Cambridge*

Miller, M. L., Goldstein, I. 1977. Problem solving grammars as formal tools for intelligent CAI. In *Ann. Conf. ACM, Seattle, Wash.*

Minsky, M. 1975. A framework for representing knowledge. In *The Psychology of Computer Vision,* ed. P. H. Winston, pp. 211–77. New York: McGraw-Hill

Minsky, M., ed. 1982. *Semantic Information Processing*. Cambridge: MIT Press

Mitchell, T. M., Keller, R. M., Kedar-Cabelli, S. T. 1986. Explanation-based generalization: a unifying view. *Machine Learning* 1(1): In press

Mitchell, T. M., Mahadevan, S., Steinberg, L. I. 1985. LEAP: a learning apprentice for VLSI design. In *Proc. 9th Int. Jt. Conf. Artif. Intell., Los Angeles, Calif.,* pp. 573–80

Moore, R. C. 1982. The role of logic in knowledge representation and commonsense reasoning. In *Proc. Natl. Conf. Artif. Intell., Pittsburgh, Pa,* pp. 428–33

Murray, W. R. 1985. Heuristic and formal methods in automatic program debugging. In *Proc. 9th Int. Jt. Conf. Artif. Intell., Los Angeles, Calif.,* pp. 15–19

Neches, R., Swartout, W. R., Moore, J. 1985. Explainable (and maintainable) expert systems. In *Proc. 9th Int. Jt. Conf. Artif. Intell., Los Angeles, Calif.,* pp. 382–89

Neves, D. M., Anderson, J. R. 1981. Knowledge compilation: mechanisms for the automization of cognitive skills. See Anderson et al 1981, pp. 57–84.

Newell, A. 1982. The knowledge level. *Artif. Intell.* 18(1):87–127

Newell, A., Simon, H. A. 1972. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall

Nilsson, N. J. 1965. *Learning Machines: Foundation of Trainable Pattern-classifying Systems*. New York: McGraw Hill

Norman, D. A. 1979. Slips of the mind and an outline for a theory of action. *Tech. Rep. 7905, Cent. Hum. Inf. Process., Univ. Calif., San Diego*

Norman, D. A. 1980. Errors in human performance. *Tech. Rep. 8004, Cent. Hum. Inf. Process., Univ. Calif., San Diego*

Norman, D. A. 1982. Five papers on human-machine interaction. *Tech. Rep. ONR-8205, Cent. Hum. Inf. Process., Univ. Calif., San Diego*

Norman, D. A., Gentner, D. R., Stevens, A. L. 1976. Comments on learning schemata and memory representation. In *Cognition and instruction,* ed. Klahr. Hillsdale, NJ: L. Erlbaum

Ohlsson, S., Langley, P. 1985. Identifying solution paths in cognitive diagnosis. *Tech. Rep. RI-TR-85-2, The Robotics Inst., Carnegie-Mellon Univ., Pittsburgh, Pa.*

O'Shea, T., Self, J. 1983. *Learning and Teaching with Computers*. Englewood Cliffs, NJ: Prentice-Hall

Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books

Patil, R. S., Szolovits, P., Schwartz, W. B. 1981. Causal understanding of patient illness in medical diagnosis. In *Proc. 7th Int. Jt. Conf. Artif. Intell., Vancouver, BC,* pp. 893–99

Pauker, S. G., Gorry, G. A., Kassirer, J. P., Schwartz, W. B. 1976. Toward the simulation of clinical cognition: taking a present illness by computer. *Am. J. Med.* 60:981–95

Perrault, C. R., Allen, J. F., Cohen, P. R. 1978. Speech acts as a basis for understanding coherence. *Theor. Iss. Nat. Lang. Process.* 2:125–32

Petroski, H. 1985. *To Engineer is Human: The role of failure in successful design*. New York: St. Martin's

Piaget, J. 1971. *Genetic Epistemology*. New York: W. W. Norton

Pople, H. 1982. Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnostics. See Szolovits & Long 1982, pp. 119–90

Quillian, M. R. 1968. Semantic memory. See Minsky 1982, pp. 227–70

Raphael, B. 1982. SIR: Semantic information retrieval. See Minsky 1982, pp. 33–145

Reiser, B. J., Anderson, J. R., Farrell, R. G. 1985. Dynamic student modelling in an intelligent tutor for lisp programming. In *Proc. 9th Int. Jt. Conf. Artif. Intell., Los Angeles, Calif.,* 1:8–14

Reiter, R. 1978. On reasoning by default. In TINLAP-2 *(Theoretical Issues in Natural Language Processing),* pp. 210–18

Rich, E. 1979. User modeling via stereotypes. *Cognitive Sci.* 3:355–66

Richer, M. H., Clancey, W. J. 1985. GUIDON-WATCH: A graphic interface for viewing a knowledge-based system. *IEEE Comput. Graphics Appl.* 5(11):51–64

Rouse, W. B., Morris, N. M. 1985. On looking into the black box: prospects and limits in the search for mental models. *Tech. Rep. 85-2, Sch. Indust. Syst. Eng., Ga. Inst. Technol.*

Rumelhart, D. E., Norman, D. A. 1983. Representation in memory. *Tech. Rep. CHIP-116, Cent. Hum. Inf. Process., Univ. Calif., San Diego*

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. *Artif. Intell.* 5(2): 115–35

Schank, R. C. 1981. Failure-driven memory. *Cognition and Brain Theory* 4(1):41–60

Schank, R. C., Abelson, R. P. 1975. *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: L. Erlbaum

Self, J. A. 1974. Student models in computer-aided instruction. *Int. J. Man-Mach. Stud.* 6:261–76

Sleeman, D. 1982. Assessing aspects of competence in basic algebra. See Sleeman & Brown 1982, pp. 185–99

Sleeman, D. 1983. Inferring (MAL) rules from pupil's protocols. In *Proc. Int. Mach. Learn. Workshop,* Monticello, Ill. pp. 221–27

Sleeman, D. 1984a. Mis-generalization: an explanation of observed mal-rules. In *Proc. 6th Ann. Conf. Cognitive Sci. Soc., Boulder, Colo.,* pp. 51–56

Sleeman, D. 1984b. An attempt to understand students' understanding of basic algebra. *Cognitive Sci.* 8(4):387–412

Sleeman, D., Brown, J. S., eds. 1982. *Intelligent Tutoring Systems*. New York: Academic

Sleeman, D. H., Smith, M. J. 1981. Modelling student's problem solving. *Artif. Intell.* 16(2):171–87

Smith, D. E., Genesereth, M. R. 1985. Ordering conjunctive queries. *Artif. Intell.* 26(2):171–215

Snow, R. E., Frederico, P. A., Montague, W. E., eds. 1980. *Aptitude, Learning and Instruction: Cognitive Process and Analyses*. Hillsdale, NJ: Erlbaum

Soloway, E. M., Rubin, E., Woolf, B., Bonar, J., Johnson, W. L. 1982. Meno-II: an AI-based programming tutor. *Tech. Rep. CSD/RR No. 258, Yale Univ., New Haven, Conn.*

Soloway, E. M., Woolf, B., Rubin, E., Barth, P. 1981. Meno-II: an intelligent tutoring system for novice programmers. In *Proc. 7th Int. Jt. Conf. Artif. Intell., Vancouver, BC,* pp. 975–77

Soo, V., Kulikowski, C. A., Garfinkel, D. 1985. Qualitative modeling and clinical justification for experimental design. In *Proc. Int. Conf. Artif. Intell. Med., Amsterdam,* pp. 21–36. Participants ed.

Sowa, J. F. 1984. *Conceptual Structures*. Reading, Mass: Addison-Wesley

Stefik, M. 1985. Review of intelligent tutoring systems (D. Sleeman and J. S. Brown). *Artif. Intell.* 26(2):238–45

Stevens, A. L., Collins, A. 1977. The goal structure of a Socratic tutor. *BBN Tech. Rep. 3518, Bolt, Beranek & Newman, Cambridge, Mass.*

Stevens, A. L., Collins, A. 1978. Multiple conceptual models of a complex system. See Snow et al 1980, pp. 177–98

Stevens, A., Collins, A., Goldin, S. E. 1982. Misconceptions in students' understanding. See Sleeman & Brown 1982, pp. 13–24

Suchman, L. A. 1985. Plans and situated actions: the problem of human-machine communication. *Tech. Rep. ISL-6, Xerox Parc*

Suppes, P. 1979. Current trends in computer-assisted instruction. In *Advances in Computers,* pp. 173–229. New York: Academic

Swartout, W. R. 1981. Explaining and justifying in expert consulting programs. In *Proc. 7th Int. Jt. Conf. Artif. Intell., Vancouver, BC,* pp. 815–23

Szolovits, P. 1985. Types of knowledge as bases for reasoning in Medical AI Programs. In *Proc. Int. Conf. Artif. Med. in Med., Pavia, Italy,* pp. 31–48

Szolovits, P., Long, W. J. 1982. The development of clinical expertise in the computer. In *Artificial Intelligence in Medicine,* ed. P. Szolovits, pp. 79–117. Boulder, Colo: Westview

Taber, J. I., Glaser, R., Schaefer, H. H. 1965. *Learning and Programmed Instruction*. Reading, Mass: Addison-Wesley

VanLehn, K. 1983a. Human procedural skill acquisition: theory, model, and psychological validation. In *Proc. Natl. Conf. Artif. Intell., Washington, DC,* pp. 420–23

VanLehn, K. 1983b. Felicity conditions for human skill acquisition: validating an AI-based theory. *Interim Rep. CIS-21, Cognitive Instruct. Sci. Group, Xerox Corp./Palo Alto Res. Cent.* Also PhD thesis. Mass. Inst. Technol., *Cambridge*

VanLehn, K., Brown, J. S. 1979. Planning nets: a representation for formalizing analogies and semantic models of procedural skills. See Snow et al 1980, pp. 95–138

VanLehn, K., Brown, J. S., Greeno, J. 1984. Competitive argumentation in computational theories of cognition. See Kintsch et al 1984, pp. 235–62

Von Wright, G. H. 1971. *Explanation and Understanding*. Ithaca: Cornell Univ. Press

Weber, J. C., Hagamen, W. D. 1972. ATS: a new system for computer-mediated tutorials in medical education. *J. Med. Educ.* 47: 637–44

Webster, N. 1983. *Webster's Ninth New Collegiate Dictionary*. Springfield, Mass: Merriam-Webster

Weiss, S. M., Kulikowski, C. A., Amarel, S., Safir, A. 1978. A model-based method for

computer-aided medical decision making. *Artif. Intell.* 11:145–72

Wenger, E. 1986. *Artificial Intelligence and the Communication of Knowledge: An Over-view of Intelligent Teaching Systems*. Los Altos, Calif: Morgan-Kaufmann. In preparation

Wilkins, D. C., Clancey, W. J., Buchanan, B. G. 1986. An overview of the Odysseus learning apprentice. In *Machine Learning: a Guide to Current Research*, ed. T. M. Mitchell, J. G. Carbonell, R. S. Michalski, pp. 369–74. New York: Academic

Winograd, T., Flores, C. F. 1986. *Understanding Computers and Cognition: a new foundation for design*. Norwood, NJ: Ablex

Woolf, B., McDonald, D. D. 1984. Context-dependent transitions in tutoring discourse. In *Proc. Natl. Conf. Artif. Intell., Austin, Tex*, pp. 355–61

Young, R. M. 1983. Surrogates and mappings: two kinds of conceptual models for interactive devices. In *Mental Models*, ed. D. Gentner, A. Stevens. Hillsdale, NJ: L. Erlbaum

Young, R. M., O'Shea, T. 1981. Errors in children's subtraction. *Cognitive Sci.* 5(2): 153–77