ED 278 362 IR 012 468

AUTHOR	Ohlsson, Stellan
TITLE	Some Principles of Intelligent Tutoring.
INSTITUTION	Pittsburgh Univ., Pa. Learning Research and Development Center.
SPONS AGENCY	National Inst. of Education (ED), Washington, DC.; National Science Foundation, Washington, D.C.
PUB DATE	85
GRANT	MDR-8470339
NOTE	50p.
PUB TYPE	Information Analyses (070) -- Viewpoints (120)
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	*Cognitive Style; Computer Assisted Instruction; *Diagnostic Teaching; *Epistemology; *Individualized Instruction; Research Needs; *Systems Development; *Teaching Methods; Tutoring
IDENTIFIERS	*Intelligent Tutoring Systems

ABSTRACT

Arguing that the main promise of computers lies in their potential for moment-by-moment adaptation of instructional content and form to the changing cognitive needs of individual learners, this paper clarifies the implications of this potential for tutoring research, and points out overlooked relations between different strands of research relevant to the construction of intelligent tutoring systems. The discussion focuses on four major topics: (1) cognitive diagnosis; (2) subject matter analysis; (3) teaching tactics; and (4) strategies for teaching. The implications of the goal of providing dynamically adaptive instruction for each of these factors are analyzed, and the results of the analyses are stated as principles about intelligent tutoring. The major conclusion is that, in order to provide adaptive instruction, a computer tutor must have a strategy which translates its tutorial goals into teaching actions, and that, as a consequence, research on teaching strategies is central to the construction of intelligent tutoring systems. A four-page list of references concludes the document. (DJR)

ARNING RESEARCH AND DEVELOPMENT CENTER

SOME PRINCIPLES OF INTELLIGENT TUTORING

1986/2

STELLAN OHLSSON

SOME PRINCIPLES OF INTELLIGENT TUTORING

Stellan Ohlsson

Learning Research and Development Center
University of Pittsburgh

1985

Preparation of this manuscript was supported in part by Grant MDR-8470339 from the National Science Foundation (NSF), and in part by funds from the National Institute of Education (NIE), U. S. Department of Education. The opinions expressed do not necessarily reflect the position or policy of NSF or NIE, and no official endorsement should be inferred.

I thank my colleagues Jeff Bonar, Glynda Hull, and Gaea Leinhardt for useful comments on both content and style.

ABSTRACT

Research on intelligent tutoring systems is discussed from the point of view of providing moment-by-moment adaptation of both content and form of instruction to the changing cognitive needs of the individual learner. The implications of this goal for ***cognitive diagnosis***, ***subject-matter analysis***, ***teaching tactics***, and ***teaching strategies*** are analyzed. The results of the analyses are stated in the form of principles about intelligent tutoring. A major conclusion is that a computer tutor, in order to provide adaptive instruction, must have a strategy which translates its tutorial goals into teaching actions, and that, as a consequence, research on teaching strategies is central to the construction of intelligent tutoring systems.

Introduction

Proponents of individualized instruction, or, as it is sometimes called, adaptive education (Glaser, 1976, Wang & Walberg, 1985), have sought to tailor instruction to such characteristics of the student as initial competence, educational goal, learning style, and, most often, learning rate, in recognition of the fact that interindividual differences in cognition implies that different learners need different instruction. Neither the history (Grinder & Nelsen, 1985) nor the current state (Bangert, Kulik, & Kulik, 1983) of such efforts are encouraging.

The birth of the computer tutor alters this situation drastically. The computer offers the potential for adapting instruction to the student at a finer-grain level than the one which concerned earlier generations of educational researchers. First, instead of adapting to global traits such as learning style, the computer tutor can, in principle, be programmed to adapt to the student *dynamically*, during ongoing instruction, at each moment in time providing the kind of instruction that will be most beneficial to the student at that time. Said differently, the computer tutor takes a longitudinal, rather than cross-sectional, perspective, focussing on the fluctuating cognitive needs of a single learner over time, rather than on stable interindividual differences. Second, and even more important, instead of adapting to content-free characteristics of the learner such as learning rate, the computer can, in principle, be programmed to adapt both the *content* and the *form* of instruction to the student's understanding of the subject matter. The computer can be programmed, or so we hope, to generate exactly that question, explanation, example, counter-example, practice problem, illustration, activity, or demonstration which will be most helpful to the learner. It is the task of providing dynamic adaptation of content and form which is the challange and the promise of computerized instruction.[1]

[1] In this article, the terms "teaching," "tutoring," and "instruction" will be used as synonyms.

When we envisage computer tutors helping out in the classroom, it is easy to be trapped into thinking in terms of particular tutors. If we want good tutoring in, say, arithmetic, what more is needed, we might ask, than a good arithmetic tutor? This perspective forgets, in the concretion of the vision, that the task of tutoring is forever changing. School systems undergo reforms, student populations follow the rhythms of the surrounding society, topics are added to, or deleted from, the curriculum, courses are moved from one age-level to another, and course contents vary from textbook to textbook, from classroom to classroom. What we need, then, are not particular, quickly outdated, computer tutors but the know-how of tutor construction. As everyone knows who have written a large computer program, it is possible to proceed by accumulating tricks and patches, ending up with a system that is so irregular that, in a sense, one cannot say why it works. But even a pedagogically successful tutor has limited interest if it leaves system designers without any information about how to design the *next* system. The output of research into computer tutoring should consist, not of particular systems, but of principles which allow specifications in terms of course content and student characteristics to be turned into effective tutors. The main points of the present article will be formulated as informal principles of intelligent tutoring; some of these principles summarize beliefs which are commonly held in the field, others indicate new directions.

In summary, the main promise of computer tutors, I claim, lies in their potential for moment-by-moment adaptation of instructional content and form to the changing cognitive needs of the individual learner, and our task, as I see it, is to find principles which can guide the construction of tutors which fulfill that promise. My main goal in the following is to clarify the implications of this view for intelligent tutoring research. I break the discussion into four sections, dealing with *cognitive diagnosis*, *subject matter analysis*, *teaching tactics*, and *teaching strategies*, respectively. In each section, I propose one or more principles which, I argue, constitute minimal criteria of adequacy for a tutor which is to provide dynamically adaptive instruction. My secondary goal is to point out some often overlooked relations between different strands of research relevant to the construction of tutoring systems. I end with a summary of the main argument.

Cognitive Diagnosis

If the purpose of intelligent tutoring systems, their *raison etre*, is to provide adaptive instruction, then such a system must know something about the cognitive state of the student, what he knows and how he thinks, and preferably how he learns. I will call the process of inferring a person's cognitive state from his performance for *cognitive diagnosis*.[2] To the extent that the major failure of current educational practices is the failure to provide adaptive instruction, then the limits on the sensitivity and resolving power of our diagnostic procedures are also the limits on the amount of pedagogical progress attainable through intelligent tutoring systems.[3] Therefore, we should look carefully at current methods for cognitive diagnosis.

Research on cognitive diagnosis has been conducted from several viewpoints. Educators have developed tests for what they call assessment or normative diagnosis (Ginsburg, 1983; Nitko, in press), and there are materials available for diagnostic teaching, materials which emphasize the diagnosis of particular kinds of errors (e. g., Hill, 1980). The problem of describing the mental state of single individuals has also been addressed by psychologists, most notably developmental psychologists (e. g., Smedslund, 1969), even though the mainstream of academic psychology has been more interested in finding regularities across individuals. Information-processing psychology has contributed new methods for inferring the mental states and processes of single individuals, as

[2]In research on tutoring systems, the term "student modelling" is often used, but I will avoid it, since it is unnecessarily restrictive. Students are not the only human beings who can be diagnosed, and a model is not the only possible kind of diagnosis.

[3]An immediate corollary is that commercially available educational software which, as a rule, do not have any diagnostic capabilities, will not, in fact, produce more learning than traditional teaching materials.

well as a new formal rigor (Newell & Simon, 1972; Williams & Hollan, 1981). Research on cognitive diagnosis from the point of view of intelligent tutoring systems goes one step further in that it aims for computer implemented diagnostic methods (e. g., Burton, 1982). This exciting development, originating, somewhat paradoxically, from within computer science, adds potential to this methodological tradition, which so far has not held center stage in psychological research.

What kind of diagnostics do we need in a computer tutor which is to provide adaptive education? In order to approach this question, we need a framework within which various types of diagnosis can be compared. In the next section, I review some types of diagnostic methods. I propose to discuss them in terms of (a) the kinds of *empirical observations* they operate upon, as well as the kinds of *knowledge structures* they need to carry out the diagnosis (i. e., their "inputs"), (b) the *procedures* by which they infer the cognitive state of the student, and (c) the *types of descriptions* they generate (i. e., their "outputs"). In addition, we ought to consider (d) the *theoretical commitments* of the various methods, and (e) the *teaching actions* they are intended to support. Of these four categories, the type of description generated by the diagnostic system is the more fundamental. There are characteristic ways of deriving each type of description, as well as typical ways of using it in instruction. Also, the theoretical commitments of a diagnostic method usually reside in the type of description it generates (rather than, say, in the procedure by which it is generated). The review in the next section has four parts, corresponding to four types of diagnostic descriptions: performance measures, overlays, error descriptions, and simulations. In a later section, I describe a new perspective on diagnosis which, I believe, is more relevant for instruction.

Before getting started, we need to be clear about terminology. In the following, "error" is used as a generic term for misconceptions, false beliefs, procedural bugs, etc. An error resides in the head, as it were. The observable expression of the error is referred to as an "incorrect answer," or "incorrect performance."

VARIETIES OF COGNITIVE DIAGNOSIS

Performance measures

The simplest way to describe a person in relation to an area of knowledge is to measure how successfully he solves problems in that area. Performance measures appear in educational contexts under a variety of labels, such as "test scores," "examination results," "assessment levels," etc. Computation of such a measure requires nothing but the student's answers - a type of data which is readily available in most contexts - and some statistical aggregation procedure. It may seem as if a performance level, being an observational construct, is blessed with theoretical innocence. However, use of the performance level as a basis for instruction assumes that we need not know ***what*** knowledge a student has acquired in order to tutor him effectively, only ***how much***. Since such a measure is a global representation of the student, it can only support global actions on the part of the tutor. For instance, a tutor can upgrade or downgrade the difficulty of practice items, it can speed up or slow down a presentation, it can provide either terse or full explanations, etc. But performance measures do not, as one authority agrees (Glaser, in press), provide the level of detail necessary to decide what ***this*** student needs ***right now*** in order to learn ***that*** concept, procedure, fact, or principle.[4]

Overlays

To provide dynamic adaptation of the content of instruction, we must move from a concern with how much knowledge a student has to a concern with what he knows. Given an analysis of the subject matter to be acquired, we can represent a student by the set of subject-matter units he has mastered, for instance, the geometric theorems which he knows how to apply. Such a

[4]**Some educators are thinking hard about how to use recent research results from cognitive psychology to construct more informative educational tests (Glaser, in press; Nitko, in press), but such tests still belong to the future.**

description is called an *overlay model* in the literature on intelligent tutoring systems (Carr & Goldstein, 1977). The student is represented as a network of tick-marks which, as it were, is laid over the representation of the subject matter to show which parts of it he already knows. From the overlay point of view, the student knows a subset of what a domain expert knows. Learning is the process of acquiring a progressively more complete subset of the expert's knowledge units, but different learners can acquire those units in different orders, so that two learners who have roughly the same amount of knowledge can nevertheless know very different things.

Overlays are particularly natural models of students when the subject matter to be learned is represented as a *prerequisite hierarchy*. In such hierarchies, which have great intuitive appeal in mathematics and other problem-solving domains, each knowledge unit is broken down into its components or prerequisites, e. g., addition and subtraction are prerequisites for long division. Prerequisite hierarchies have been investigated by educators and psychologists (e. g., Gagne, 1962; Resnick, 1973; Resnick, Wang, & Kaplan, 1973) and have been re-invented by computer scientists. They are used quite frequently in intelligent tutoring systems, without the kind of empirical verification which educational researchers regard as necessary (Resnick & Wang, 1969).

The typical procedure for inferring an overlay works by relating the student's performance to the prerequisite structure of the subject matter. If a student reliably succeeds on a particular type of task, then he can be inferred to have acquired whatever knowledge items are prerequisites for succeeding on that task. If he fails reliably, at least one of the prerequisites can be assumed to be missing from his competence. Since the various knowledge items are needed in different combinations for different problem types, one can puzzle out which items a student has mastered by a judicial choice of problems. Sandra Marshall (1980; 1981) has worked out a precise theory for how to compute an overlay model for a prerequisite hierarchy. The essence of her diagnostic procedure is to compute a probability distribution over the set of possible diagnoses, to revise that distribution after each response from the student on the basis of Baysian decision theory, and to use statistical information theory for choosing maximally informative problems to present. This kind of diagnosis is widely used in intelligent tutoring systems, usually without the mathematical sophistication introduced by Marshall.

Like the performance level, the overlay at first seems to be innocent of theoretical commitments. In fact, overlays make two assumptions which limit their usefulness for instruction. A prerequisite hierarchy - or any other type of analysis - views the subject matter in a particular way, from a particular vantage point. Since the overlay representation of the student is parasitic on the representation of the subject matter, the overlay shows how far the student has progressed in acquiring *a particular view of the subject matter*. However, rarely, if ever, is there only one possible view, and, unfortunately, if the view used by the tutor is not the view which the student is trying to attain, as it were, then the overlay might be useless as a basis for instruction.

The second detrimental property of an overlay is that it assumes that a student is a subset of an expert. The learner is seen as knowing the same things as an expert, only fewer of them. This perspective ignores the possibility of *distorted knowledge*. However, recent research into arithmetic, algebra, and physics has revealed that different learners do not just differ in which subset of the subject matter they have acquired, but also in how they misunderstand that part of the subject-matter which they have not yet mastered (see below). There are no resources within the overlay to represent this fact.

An overlay model, particularly when coupled with a prerequisite analysis of the knowledge domain, can guide the decision of what topic to tutor next: In general, the next topic should be chosen from those subject-matter units which are not yet mastered, but which have all their prerequisites ticked off as known. Thus, an overlay allows a tutor to solve the traditional pedagogical problem of sequencing the subject matter, and, moreover, to solve it *adaptively*, for each student, and *dynamically*, while instruction is happening.

Error Descriptions

The empirical evidence is now overwhelming that learners do not only fail to acquire the content presented to them during instruction, they also *misrepresent* it. Students acquire erroneous procedures, false principles, and incorrect facts, both inside and outside school (e. g., Gentner & Stevens, 1983; VanLehn, 1983). The most spectacular progress in cognitive diagnosis in the past years is the appearance of computerized diagnostic methods for identifying the errors of individual learners. Such methods have so far been successful in a small number of knowledge domains, most notably arithmetic (e. g., Burton, 1982), algebra (e. g., Sleeman, 1982), and elementary programming (Johnson & Soloway, 1983; 1984; Soloway, Rubin, Woolf, Bonar, & Johnson, 1982).

These diagnostic systems usually work with performance data, i. e., the pattern of correct and incorrect answers on a set of problems, or, in the case of programming, a single answer. The inferential procedure which is used in this type of diagnosis is based on an *error library*, i. e., a list of possible errors. Given a particular student, it is possible to compute which error (or combination of errors) best account for the incorrect answers of that student. The basic difficulties in constructing such systems are (a) to establish the error library, which is a labour-intensive empirical undertaking, and (b) to invent methods which can compute the best-fitting error combination efficiently enough to deliver a diagnosis in a reasonable time.

The basic idea of error identification through an error library has been significantly extended in the PROUST system by Johnson and Soloway (1983; 1984), which takes a programming problem, i. e., a specification of a program, and a (typically incorrect) program as inputs, and tries to understand the incorrect program. PROUST has several different knowledge structures to draw upon, including an error library, a library of programming plans, knowledge of programmers' goals, etc. Its inferential procedure attempts to develop a goal structure for the problem which accounts for all the parts of the code in the observed program, using both correct, and, if necessary, incorrect programming plans as bridges between a particular goal decomposition and the faulty code. PROUST delivers a hypothesis about the programmer's intentions, and about how he tried to realize them, i. e., it generates an account of how the faulty program was *created*. Besides the programmer's intentions in writing the program (the goal hierarchy), the account also includes the domain-specific knowledge the programmer used (the correct programming plans), and the mistakes he did (the errors that must be posited to make the account complete). PROUST has been successful in diagnosing incorrect programs by novice programmers.

The theoretical commitments of error identification systems hide in their descriptions of errors. An error library reifies the errors, as it were, by making a decision as to what is a primitive error and what is a compound error. Furthermore, since errors are unavoidably described as deviations from correct knowledge, error descriptions presuppose a particular representation of the subject matter. Different analyses of the subject matter may result in different error libraries, and so in different diagnoses of particular students.

The pedagogical promise of an error description is that it might enable ***remedial instruction***. Given knowledge of which particular error a student is suffering from, we should be able to help him overcome it. However, deciding on the best remedy for a particular error is not a trivial matter, and, in fact, error descriptions have not been used in many tutoring systems. The approach taken by Attisha and Yazdani (1983; 1984) is to have the tutoring system type out a prestored natural language description of an error. The same idea has been used in the tutors constructed by John Anderson and his co-workers (Anderson, Boyle, & Reiser, 1985; Anderson, Boyle, & Yost, 1985; Reiser, Anderson, & Farrell, 1985), with the extension that they link specific errors with natural language ***templates*** containing variables which are filled with situation-specific information at the time they are output to the student. Reaching for the next level of flexibility, Woolf and McDonald (1984) are designing a system to do remedial teaching through a didactic dialogue. As we shall see in the section on teaching tactics, talking to a student about his error is not the only kind of remedial teaching.

Simulations

A description of a student's cognitive state can consist of a simulation model which performs like him in the relevant knowledge domain. The main characteristic of such a representation is that it is ***runnable***, i. e., that it will generate behavior when applied to a task. The behavior is a prediction of what the simulated person would do, if he solved the same task. With this kind of representation, we can explain the student's steps towards his answer to a problem, as well as the answer itself.

The first systematic method for constructing simulation models was proposed by Newell and Simon (1972). Their method takes think-aloud protocols as input, and the inferrence from data to simulation model proceeds in three steps. First, the protocol is scanned for evidence concerning the student's[5] mental encoding of the task, as well as for the cognitive operations he is applying to it. An encoding of a task and a set of operators together define a ***problem space***; consequently, we call this the problem space approach to diagnosis (Ohlsson & Langley, in press; 1985). Second, the sequential information in the protocol is used to construct the student's path to solution. Third, problem-solving rules are invented which can generate the solution path. The rule set is a model for the student's performance on the relevant task. This is a very labour-intensive diagnostic method. Early efforts to computerize it were not successful (Waterman & Newell, 1972), because, one might guess, of the need for such a system to understand the utterances in the think-aloud protocol, a task that was beyond the natural language capabilities

[5]The method was conceived in the context of basic research on human problem-solving, but in keeping with the terminology in the rest of this article, I will write about "students" rather than "subjects."

of computers at the time.

Pat Langley and I have recently collaborated on a different effort to computerize the problem space approach to cognitive diagnosis Langley, Ohlsson, & Sage, 1984; Ohlsson & Langley, in press; 1985). We constructed a computer program which takes an answer as input, rather than a think-aloud protocol, plus a problem space. The system searches the given problem space for a path which leads to the observed (possibly incorrect) answer, chosing between alternative paths on the basis of psychological criteria. Like the PROUST system discussed in the previous subsection, our program creates an account of how the incorrect answer was *created*. However, our type of account - a solution path - is much simpler than the type of generated by account used by PROUST. Once a path has been found, so-called machine learning methods can be applied to it in order to generate problem-solving rules which will reproduce the observed answer. Our program has successfully diagnosed a small set of subtraction errors.

The output from the problem space approach to diagnosis does not, in a sense, *describe* the errors of the diagnosed student. Rather, the method generates a procedure which performs the relevant tasks in the same way as the student. Whether that procedure is to be classified as correct or incorrect cannot be determined with reference to the way in which it was generated. The diagnostic method does not make use of the concept of "correctness," nor of an error library. Errors are *implicitly* described in the way in which the rules are stated, rather than explicitly described, as in the error identification methods described previously.[6]

It is interesting to contrast our diagnostic method, with the *model-tracing* method used by John Anderson and his co-workers (Anderson, Boyle, & Yoost, 1985; Reiser, Anderson, & Farrell,

[6]The explicit description of the error can be recovered by comparing the rules generated by the diagnostic procedure with the rules for correct performance. However, this would introduce a difficulty which is now familiar: *which* representation of correct performance should the student's rules be compared to? Different representations of the correct procedure will lead to different characterizations of the differences.

1985). The model-tracing method operates in the context of intelligent tutoring systems which monitor students' problem- solving activity. In that context, the diagnostic component need not compute the student's solution path, because it can *observe* what the student does, step by step. Also, the diagnostic system has a knowledge base consisting of several hundred problem-solving rules which encode both correct knowledge about the domain and typical errors. The system builds a model of the student by locating, for each problem-solving step he takes, which rule (or rules) corresponds to that step. By operating in a context where the solution path is observable and by having a large rule library, model-tracing diagnosis essentially *side-steps* the diagnostic problem that the problem space approach tries to solve. There is a lesson to be learned here with respect to intelligent tutoring which will be spelled out in detail in the next section.

The product of a model-tracing diagnosis is similar to an overlay in that it consists of a selection of knowledge units from a pre-established set. However, since the rule base can include incorrect as well as correct rules, the model-tracing diagnosis can represent both the student's knowledge and his errors. Also, since the rule base can contain alternative versions of the correct problem-solving rules, model-tracing diagnosis can model a learner with respect to several alternative versions of the correct problem-solving skill. Finally, since the knowledge units are executable rules, the rule set which represents the student is, in principle[7], a runnable model.

The theoretical commitments of simulations are more visible than the commitments of the other types of descriptions discussed. A simulation presupposes a performance theory. It must specify the mental representation of both declarative and procedural knowledge, it must assume something about memory stores and their properties, it must take a stand on whether human cognition is goal driven or not, and, if so, how goals are managed, it must run under some particular cognitive architecture, etc. The problem space approach to diagnosis is based on the

[7]Anderson and co-workers do not make any strong claims about the runnability of their student models.

theory that thinking consists of heuristic search through a problem space (Newell & Simon, 1972). The model-tracing methodology is strongly influenced by the general idea of heuristic search, but builds in its specifics on the ACT* theory (Anderson, 1983). Other performance theories might lead to yet other diagnostic methods.

A simulation model is a powerful description. Intuitively, we would expect such a description to buy us dramatic advantages in instruction, particularly with respect to individualized instruction. It is therefore suprising to realize that it is not clear what one can do with a simulation of the student that one cannot do without it. Planning and designing instruction which is maximally useful for a student is not a simple task, even with the help of a powerful description of that student.

IMPROVING COGNITIVE DIAGNOSIS

The different types of cognitive diagnosis discussed in the previous section form a regular progression towards more and more powerful descriptions of the student. A performance measure indicates the proportion of the subject matter which the student has mastered; an overlay model specifies *which* portion, and thereby, indirectly, which knowledge units the student still needs to learn. Error descriptions add information about what we might call negative subject matter, i. e., the distorted and twisted knowledge units which the student needs to *un*learn. Finally, simulations enable us to run the representation of the student, and thereby to make detailed predictions about his performance.

Although they differ with respect to completeness of description, all the diagnostic systems reviewed in the last section, with the exception of the model-tracing method, share a common perspective. They are based on a view of diagnosis as "research in miniature," to use Ginsburg's (1983) happy phrase. The task of diagnosis is, according to this perspective, to explain, or account for, a set of observations (performances). These methods (again, with the exception of model tracing) do not take into account how the diagnosis they produce is going to be *used*. But we need to know what the purpose of diagnosis is in order to decide how powerful a diagnostic method we should incorporate into our tutoring systems.

Let us consider the decision situation of a tutor, and what role the diagnostic component can play in it. A computer tutor is usually able to present practice problems and to provide feedback; it might be able to demonstrate correct solutions, to provide explanations, give hints, ask questions, etc. But no matter now sophisticated and versatile the tutoring system is, it will only have a finite number of actions to choose from, and its only task, one might claim, is to select one of those actions as the next thing to do. (This is a fair description of how some tutoring systems work.) The tutor's only use of the diagnosis, it seems, is to help select that next action. This perspective tempts us to view the diagnostic component of the tutor as a mechanism for mapping tutorial situations onto instructional actions, i. e., as imposing an equivalence-classification on the set of tutorial situations, one class containing all situations in which action **A** (e. g., explaining) is pedagogically superior, another class containing all situations in which action **B** (e. g., demonstrating), is superior, etc. This kind of diagnosis seems ripe and ready to be attacked with the well-established methodology of expert systems (e. g., Waterman, 1986).

Before falling for the temptation to develop this perspective further, we should remember that intelligent teaching does not consist of sequences of unrelated actions. A tutoring effort is structured; it coordinates the individual teaching actions, subsumes them under a plan for how to transmit the relevant knowledge. The moment-to-moment behavior of the tutor originate in the execution of that plan, rather than in successive decisions about what to do next. If the student model is to be useful, it has to contribute in some way to the construction and execution of instructional plans. This perspective is sufficiently different from the idea of "research in miniature" to be stated as a principle:

The Principle of Pragmatic Diagnosis. The purpose of the diagnostic component of an intelligent tutoring system is to support the execution of its instructional plan.

Considering diagnosis from this point of view, we are lead to ask what function cognitive diagnosis could have in the construction and execution of instructional plans. The crucial link between an instructional plan and a student model is, I propose, that instructional plans contain (implicit) *expectations* about what the student will do in response to the tutor's actions. For instance, consider the following simple, two-step plan for teaching the solution to problem Y: "first review problem X (e. g., addition of proper fractions), then introduce problem Y (e. g., addition of mixed fractions), and then show how the solution to X can be applied to Y as well." This plan implicitly presupposes that the student knows how to solve problem X correctly, so that X can be used as a pedagogical resource or lever in the teaching of Y. If this expectation is violated - the student fails to solve X - then the plan becomes irrelevant and has to be revised. Every instructional plan, I believe, contains implicit expectations about the student, expectations which, if violated, render the plan irrelevant.

This observation suggests that one function of cognitive diagnosis in tutoring is to test the viability of the expectations underlying the tutor's instructional plan.

The Principle of Expectation Testing. The function of the diagnostic component of an intelligent tutoring system is to test whether the expectations presupposed by the tutor's current plan are consistent with its current model of the student.

One procedure for making such a test is to predict, from the current model of the student, how he will respond to the tutor's actions, and then compare the predictions with the expectations. If they diverge, then the plan is not viable *vis-a-vis* current knowledge of the student. The plan should then be revised, and the expectations of the new plan tested, etc., until a viable plan is found. If it turns out that the *actual* (as opposed to the predicted) behavior of the student does not conform to the expectations/predictions, then, of course, both the instructional plan and the student model have to be revised. This argument speaks strongly in favor of runnable student models.

In summary, a number of diagnostic methods exist which generate more or less powerful representations of the cognitive state of the student. They typically perform the task of diagnosis independently of the pragmatics of teaching. By changing the diagnostic question from "What goes on in the head of the student?" to "What does the tutor need to know in order to teach?" we turn our attention to the nature of teaching strategies. A crucial property of such strategies is that they contain implicit expectations about student behavior, which, in turn, suggests that one important function for cognitive diagnosis in instruction is to help decide whether those expectations are viable or not.

Subject-Matter Analysis

Teaching, as Wenger (1985) reminds us, is an act of knowledge communication, even when it is carried out by an artifact. Its goal is to transmit a particular subject matter to the student. We need to ask what the implications of computer tutoring and adaptive instruction are for representations of subject-matter. I will argue (a) that subject-matter analysis is important because an adaptive computer tutor needs to make a distinction between the subject matter itself and its surface manifestations, and (b) that a central obstacle to intelligent tutoring is the fact that different learners may acquire different representations of the target knowledge.

In our discussion of cognitive diagnosis, we noticed that three different communities of scholars have been involved in research on diagnosis, namely educators, psychologists, and computer scientists. With respect to subject-matter analysis, we would have to mention those three, and then add that domain experts often engage in such analyses as well, particularly in areas like mathematics and physics. But commonalities in goals, methods, and results are often obscured by differences in terminology. What is known as "subject-matter analysis" among educators often goes under the name "task analysis" among psychologists (e. g., Gardner, 1985); a mathematician might prefer the term "didactic analysis" (e. g., Steiner, 1969), while a computer scientist is certain to use "knowledge representation" (e. g., Brachman & Levesque, 1985). The term "subject-matter analysis" is used here without any ulterior motive; either term could have served equally well.

WHY IS SUBJECT-MATTER ANALYSIS IMPORTANT?

The traditional educational task of finding the right representation of the subject matter has been taken up with a vengeance by the community of scholars involved in intelligent tutoring systems research. A glance at recent research (e. g., Sleeman & Brown, 1982) shows cognitive scientists hard at work giving their artificial tutors explicit representations of the relevant subject matter. Indeed, Wenger (1985), in his comprehensive review, argues that the major research questions addressed in intelligent tutoring research are issues of knowledge representation, and even draws the line between the prehistory and the history of intelligent tutoring systems at the moment when the tutors began to acquire an internal representation of the subject matter.

From the point of view of individualized instruction, the importance of subject-matter analysis is not immediately obvious. The knowledge to be taught is that which remains the same across students. Like gasoline, a substance which can drive many different kinds of vehicles without adapting itself to them, the content of a course may drive the learning of many different students, while remaining the same course content. If so, how does subject-matter analysis contribute to individualized instruction, if at all?

We can approach this question through a distinction between a knowledge item, a piece of subject matter, and the *form* in which it is presented to the student. For instance, consider a mathematical theorem, such as the familiar Cancellation Law for fractions:

$$\frac{a * N}{b * N} = \frac{a}{b}$$

This piece of knowledge can be presented in at least the following formats: (a) as a principle (e. g., "A fraction in which the numerator and the denominator share a factor ... ."), (b) as a mathematical formula (e. g., the one above), (c) as an algorithm (e. g., "First find the factors of the numerator, then ... ."), (d) as a demonstration of how to execute the algorithm on a particular problem, (e) as a demonstration of the algorithm with respect to some illustrative materials (e. g., Dienes blocks), (f) as a collection of solved examples in which the algorithm has been applied, and (g) as a proof that the theorem is correct. The range of possible presentation formats is no more restricted for other types of knowledge units.

A student struggling to learn the Cancellation Law may benefit most from one or the other of these presentations, depending upon the nature of the obstacle to his learning. A justification may enhance memory for the theorem; a demonstration might facilitate its application. It follows that a tutor, to be maximally helpful, must generate the presentation of the theorem on the spur of the moment, in response to the cognitive needs of the student at that moment. This implies, in turn, that the tutor must possess a core representation of the theorem, a deep structure, as it were, which is independent of the different presentation formats, but from which the specific presentations, the surface structures, can be derived. There is thus a strong connection between individualized instruction, on the one hand, and the separation between content and form, between subject matter and its tutorial shape, on the other.

> **The Principle of Generative Interfaces**. In order to provide adaptive instruction, a tutor must distinguish between the subject matter and the formats in which it can be presented, and be able to generate different presentations of each subject-matter unit as needed, at each moment in time choosing the form which is most beneficial for the learner at that moment.

The above principle spells out the essential difference between intelligent tutoring systems and other types of teaching materials. The printed page, the audiotape, and the movie are all equally incapable, in principle, of separating content from form. Older, so-called frame-based CAI systems also failed to separate the two, and so constituted a very limited advance upon traditional teaching materials, if any at all.[8] However, the computer has the potential to teach the way a good human tutor would teach, adapting the form of his presentation to the individual student by applying generative procedures, guided by diagnostic information, to a deep representation of the subject matter.

We can now understand why designers of intelligent tutoring systems pay so much attention to subject-matter analysis. Implementing a presentation-neutral representation of domain knowledge, plus the generative mechanism which is to produce the surface presentations of it, usually requires considerable clarification and extension of existing representations of the subject matter. The reader is referred to the retrospective report by Brown, Burton, and DeKleer, 1982; (particularly pp. 244-279) on the SOPHIE trouble-shooting tutor for an illustration of just how much thinking may be needed to implement a so-called domain expert. In short, the Principle of Generative Interfaces is not a new insight; instead, it explains why the field of intelligent tutoring concerns itself almost exclusively with intelligence, but hardly at all with tutoring: The need for a deep representation of the subject matter puts the issue of knowledge representation at the top of the tutoring agenda, and the difficulty of that issue ensures that it stays there.

THE PROBLEM OF ALTERNATIVE LEARNING TARGETS

A computer tutor typically embodies *a* representation of the subject matter, *a* domain expert. The target knowledge has only one representation inside the tutoring system. Considering the

[8]It is worth noticing that most commercial educational software is of the frame-based variety.

effort required to implement even a single domain expert, this is not surprising.

But there are no canonical representations of knowledge. Any knowledge domain can be seen from several different points of view, each view showing a different structure, a different set of parts, differently related. This claim, however broad and blunt - almost impolite - it may appear when laid out in print, is, I believe, uncontroversial. In fact, the evidence for it is so plentiful that we do not notice it, like the fish in the sea who never thinks about water. For instance, empirical studies of expertise regularly show that human experts differ in their problem solutions (e. g., Prietula & Marchak, 1985); at the other end of the scale, studies of young children tend to show that they invent a variety of strategies even for simple tasks, (e.g., Svenson & Hedenborg, 1980; Young, 1976). As a second instance, consider rational analyses of thoroughly codified knowledge domains such as the arithmetic of rational numbers. The traditional mathematical treatment by Thurstone (1956) is hard to relate to the didactic analysis by Steiner (1969), which, in turn, does not seem to have much in common with the informal, but probing, analyses by Kieren (1976; 1980) - and yet, they are all experts trying to express the meaning of, for instance, "two-thirds." In short, the process of acquiring a particular subject matter does not converge on a particular representation of that subject matter. This fact has such important implications for instruction that it should be stated as a principle.

The Principle of Non-Equifinality of Learning. The state of knowing the subject matter does not correspond to a single, well-defined cognitive state. The target knowledge can always be represented in different ways, from different perspectives; hence, the process of acquiring the subject matter have many different, equally valid, end states.

The non-equifinality of learning causes severe difficulties for intelligent tutoring, some of which have been discussed by Burton, 1982; (see in particular pp. 94-95) in connection with the WEST tutor. As discussed in the section on diagnosis, if the representation of the student is parasitic on the representation of the subject matter, as in the case of overlay models, then the diagnosis interprets the student in terms of how far he has developed toward acquiring a particular view of the subject matter, which may or may not be a useful representation of him. The corresponding problem occurs on the output side of the tutor: Which analysis of the subject matter should be used in deciding which topic to tutor next? Which breakdown of the subject matter into topics is relevant for the particular student?

One partial solution to the non-equifinality problem is to implement more than one representation of the subject matter, and allow the tutor to use all of them both when trying to understand the student and when making instructional decisions. This solution might work in domains where a small number of different representations of the target knowledge cover a large proportion of students. For instance, in the domain of subtraction, there might be only a small number of psychologically plausible encodings of the correct skill, and a computer tutor could try to judge which of these encodings a particular student is *en route* toward, and base its instruction on that hypothesis. The solution to implement more than one represenation of the domain knowledge might also work well in domains where the knowledge can be expressed in very modular units, e. g., production rules. The representation of the domain could in this case consist of a highly *redundant* rule set, with several different versions of each correct rule. A particular representation of the subject matter can then be created on-line, as it were, by selecting a subset of rules. This is the solution used in the tutoring systems by John Anderson and his co-workers (Anderson, Boyle, & Yost, 1985; Reiser, Anderson, & Farrell, 1985). Its success depends upon the procedure used for selecting the right subset of rules.

An entirely different approach to the obstacles caused by non-equifinalty in learning is to try to minimize the effects of the subject- matter representation on the tutoring effort, an idea which goes at right angles to the current direction of the field of computer tutoring, where most researchers seem intent on getting as much out of their domain experts as possible, once implemented. It is unclear to what extent a tutor can be made independent of its representation of the subject matter. It seems possible to design a diagnostic method which is not parasitic on the subject matter by using data-driven techniques. But it does not seem possible to design a tutoring strategy which can choose which topic to teach next without accessing a representation of the subject matter; indeed, such an idea seems self-contradictory.

However, one might imagine that we could program a general notion of ***improvement of knowledge***. Such a notion would make it possible to construct a tutor which dispenses with a representation of the ***target*** knowledge, but bases its tutoring on a represenation of the student's ***initial*** knowledge instead. Such a tutor would try to make the student improve his knowledge, in any direction, applying its general criterion of improvement to guide the student's learning, not caring, as it were, where the student is going, as long as he is going somewhere. Whether it is possible to define a notion of knowledge improvement which is content-free enough to support such tutoring we do not know.

In short, the non-equifinality of learning poses serious problems in the construction of intelligent tutoring systems. It affects central aspects like diagnosis of the student and pedagogical decision making. No principled solution to this problem has been proposed yet.

Teaching Tactics

Consider a pianist playing a complicated piece, score spread out in from of him. In a sense, the player is trying to ***follow*** the score. He has a number of actions to choose from - all the key-presses, with the appropriate variations - and for each symbol in the score, each note, there is one action, one key-press, which is the right one; that single note is played in the context of the key, the tempo, and the adjacent notes. Consider how much more difficult it would be to play the piano, if the player were guided by a screen, on which the notes were projected one at a time, at the rate at which the piece should be played. The situation of the one-on-one tutor is similar.

The tutor is following the learner, trying to respond to each action or performance or utterance with the right tutorial action.

Pushing the analogy one step further, consider what would happen if we deprived the piano player of the lower and the upper octaves, and perhaps of every other key in between as well. With such a narrow range of keys the player could not always follow the score. There would be many pieces of music that he could not play on such a mutilated instrument. As with playing the piano, so with one-on-one tutoring. If the tutor has a limited range of actions to choose from, it cannot adapt its teaching to the cognitive needs of the student. Where one student needs a definition, another needs an explanation, while a third student might learn better from a practice problem. This idea is simple but important, and so is expressed in the following principle.

> **The Principle of Versatile Output.** In order to provide adaptive instruction, a tutor must have a wide range of instructional actions to choose from.

An obvious source of information about useful instructional actions is the performance of teachers, in particular, expert teachers. (The computer tutor is not limited to the actions used by human teachers, but neither is there any reason to believe that it can make do with a *narrower* range of actions.) Inspired by Leinhardt and Greeno's (in press) careful analysis of the skill of teaching, let us think through the major types of teaching tactics needed for adaptive instruction.

In order to keep the following remarks within reasonable bounds, I will only consider tactics for teaching cognitive skills, like those involved in elementary mathematics.[9] The procedure to be taught will be referred to as the *target procedure*. I will define six classes of tactics and give some examples of each class. For the teaching of facts and principles, we have to imagine yet other tactics.

[9]No stand is implied on the old issue of whether mathematics should be taught through understanding or through drill.

Category 1: Tactics for Presenting the Target

The actions in this category have to do with ways of presenting the target procedure. Obvious actions of this sort are to ***define terms*** needed to talk about the procedure, to ***describe*** the procedure, and to ***prompt recall*** of the procedure. A procedure can also be ***demonstrated***, i. e., executed. In fact, there are several different kinds of demonstrations, such as ***annotated*** demonstrations, where the tutor explains and justifies each step as it is being executed, and ***interactive*** demonstrations, where the the student specifies the steps to be taken, but the tutor executes them. A procedure can also be ***applied***, i. e., executed on a concrete example (e. g., the standard procedure for subtraction can be applied to, say, Dienes blocks). A procedure can also be ***practiced***, of course, and there are many different types of practice: ***guided*** practice, in which the tutor specifies the steps one at a time, but the student executes them; ***annotated*** practice, in which either the tutor or the student justifies the steps as they are being executed; ***corrected*** practice, in which the tutor immediately corrects any incorrect step on the part of the student; practice in which the tutor ***hints*** at the correct steps; and sheer ***drill***, in which the tutor does not intervene unless the student asks for help.

Category 2: Tactics for Presenting Precursors

A precursor to a target procedure is a skill that the learner is expected to have mastered before he attempts to learn the procedure. A precursor is usually, but not always, a prerequisite, i. e., a skill which constitutes a component of the target skill. The multiplication of integers is a precursor to the multiplication of fractions; it is also a prerequisite. But the multiplication of fractions is, in most schools, a precursor to the multiplication of decimals, but it is not a prerequisite. Actions in this category include the following: ***priming***, in which the tutor reminds the student of the precursor by naming it or by locating it (e. g., "remember the theorem we learned last time"); ***reviewing*** the precursor; and ***marking*** those steps in a presentation which should be familiar, and distinguish them from those steps which are expected to be unfamiliar.

Category 3: Tactics for Presenting Purposes

The actions in this category serve to explain what the target procedure is ***for***, what it is supposed to achieve. The tutor can describe the purpose of a new procedure by ***giving a goal***, i. e., describing a desired result, and then introducing the target procedure as a method for achieving that result. Other tactics include to ***criticize*** the precursors by showing a problem for which they are insufficient, but which the target procedure can solve; to describe the target as a ***generalization***, or, more generally, ***replacement***, of the precursor (e. g., the arithmetic of decimals is, in a sense, a replacement for the arithmetic of fractions).

Category 4: Tactics for Presenting Justifications

The tutor can justify a procedure by ***annotating*** it, i. e., by relating each step in its execution to some principle. Another way is to ***give transparent cases***, i. e., the tutor executes the target procedure on problems in which the steps taken are intuitively obvious (e. g., using unit fractions to show how multiplication of a fraction with an integer works). Yet another way is to relate the target procedure to some ***equivalent procedure***, which already is justified; the target procedure then inherits the justification. The various ways of ***verifying*** the results of executing the target procedure constitute a useful subcategory. One can verify the outcome of a procedure by computing it with an ***alternative*** procedure, in some cases by applying the ***inverse*** procedure, or by making an ***empirical*** test of it.

Category 5: Tactics Related to Errors

The programming term "bug" is often used in the literature on intelligent tutoring systems to refer to systematic procedural errors. A tutor can ***reveal*** a bug in a procedure by applying the procedure on a problem where it will generate an obviously incorrect, absurd, or impossible result, or he can explicitly *mark* the buggy step as it is generated in execution. The tutor can also choose to *explain* a bug, i. e., to show which principle is being violated by the bug.

Category 6: Tactics Related to Student's Solutions

The most common tactics in this category are to *give feedback* and to *locate* the error. But a tutor can also *prompt a self-check* on the part of the student, *prompt a self-review* in which the student tries to describe a sequence of steps he has taken, or *prompt self-annotation*, i. e., exhort the student to justify his own steps.

The six categories of teaching tactics presented above is only an initial attempt to list the various actions an intelligent tutor needs in its behavioral repertoire in order to provide adaptive instruction in the acquisition of procedural knowledge. Further reflections and more detailed observations of expert teachers will no doubt lead to extensions of the list. Even as it stands, however, the above list is an order of magnitude richer than the behavioral repertoire of any existing intelligent tutoring system.

There are several reasons why current computer tutors tend to have limited output repertoires. First, many computer tutors are restricted in their *pedagogical scope*. For instance, some tutors do not teach or tutor in the ordinary sense of the word. Instead, they monitor practice, i. e., they generate practice problems and give feedback. Second, the richness of tutorial actions derives to a large extent from the variety of ways in which a particular knowledge unit can be presented. Unless the tutor can present its subject matter in more than one way, this richness will not be reflected in its action repertoire. Third, many tutorial systems have a narrow behavioral repertoire because they have a primitive diagnostic component. Unless the tutor can distinguish between many different cognitive states of the learner, it does not need a rich output repertoire. Fourth, many tutoring systems have a narrow behavioral repertoire because they have a simplistic teaching strategy. Unless the system designer knows how to make use of a particular teaching tactic - when, under what circumstances, to evoke it - then there is little purpose in implementing it.

> **The Principle of Strategic Repertoires.** The range of teaching tactics in a tutoring system is ultimately limited by the conditionality of the teaching strategy of the system; unless the strategy can identify circumstances under which a particular tactic is to be evoked, the tactic will not increase the power of the system.

Like the diagnostic component, the behavioral repertoire of an intelligent tutoring system is logically secondary to its teaching strategy.

Strategies for Teaching

Cognitive diagnosis and subject-matter analysis generate the "inputs" to a tutoring system, the information which forms the basis for its tutorial decisions. The teaching tactics discussed in the previous section, on the other hand, represent its "output," i. e. its behavior *vis-a-vis* the student. However, so far I have said nothing about *how to teach*, about how to generate a sequence of teaching tactics which will successfully transmit the subject matter to a particular student.

In spite of the fact that millions of teachers spend millions of hours every year teaching millions of students, nobody seems to know how to teach. There is no *Handbook of Pedagogical Methods* which we can take down from the shelf and from which we can read off the correct teaching strategy for some important part of the curriculum, such as arithmetic. Educators often invent interesting strategies for specific topics (e. g., Beede, 1985), but they have not made it part of their professional commitments either to empirically validate their proposals, or to explain, through theoretical analyses, why we should expect those proposals to be effective.

Psychologists are, of course, concerned with empirical verification, but, as a recent review reminds us (Bell, Costello, & Kuchemann, 1983, chap. 7), they have chosen to work with global ideologies of teaching, rather than with specific teaching strategies. The section headings of Bell, Costello, and Kuchmeann (1983, chap. 7) reads like so many old friends: exposition vs. discovery, meaningful vs. rote learning, learning hierarchies, advance organizers, etc., until, at the end, we reach the Grand Daddy of all topics in experimental educational research, the spacing of repetitions. Even if this research was in the habit of generating strong conclusions, such high-level ideas would only go part of the way towards the construction of specific teaching strategies. Cognitive scientists could have made it their business to analyze teaching, but, as we have seen, they believe that the greatest profits are to be found in the area of knowledge representation. In short, none of the three communities of scholars who are involved in research on education have made it their primary task to study teaching strategies.

The question to be asked here is how the goal of providing dynamically adaptive instruction constrains the method of teaching. What teaching strategies are compatible with this goal and which are not? I approach this question, first, by presenting an example of teaching methods invented by teachers, and, second, by analyzing the teaching strategies used in the most successful current computer tutors, primarily the tutors designed by John Anderson and his co-workers. In a third section, I outline a view of teaching which, I claim, is powerful enough to support adaptive instruction.

PLANS IN THE CLASSROOM

The classroom is an obvious source of inspiration when thinking about strategies for instruction. The methods being employed in classrooms have passed a stringent field test of sufficiency, since most school children do learn what they are supposed to learn. Leinhardt (1985) observed one expert teacher using the following plan for teaching multicolumn subtraction with regrouping:

1. Mix subtraction problems for which borrowing is necessary with the nonborrowing

problems, and make the student observe and recognize them before he knows how to solve them, indeed, even before he has completely mastered nonborrowing subtraction.

2. Once subtraction without borrowing is understood, i. e., set the goal of learning how to solve the borrowing problems.

3. Leave the subtraction context, and teach the regrouping procedure in isolation. Prove its validity by presenting cases in which the learner can check that it works, i. e., that the value of the regrouped number is not changed by the procedure. Use concrete illustrations as well as numerical examples. Provide a large number of cases.

4. Move back to the context of subtraction, and show how the regrouping procedure can be applied to borrowing problems, i. e., to the class of problems that was set aside previously.

This plan emphasizes the fact that intelligent instruction does not consist of sequences of *unrelated* actions, but, on the contrary, coordinates several actions in the service of a particular pedagogical goal, in this case the goal of overcoming the common difficulties with the borrowing operation. The justification for a plan consists of means-ends relations, e. g., the plan above seems to be based on the idea that distinguishing between borrowing and nonborrowing problems from the beginning of instruction in subtraction helps the child learn the correct conditions for when to apply the regrouping operation. Such relations are not explicitly stated in the plan, but implicitly expressed in the way it orders the teaching tactics.

The means-ends structure of the plan is important, because some of the actions the teacher performs while executing the plan would serve no purpose, have no meaning, in isolation from the other actions. For instance, making the learner recognize and acknowledge borrowing problems before they can solve them has no meaning within the task of mastering nonborrowing subtraction. It is purposeful within the plan because it prepares the learner for the following step of learning how to borrow. As a second instance, the action of proving that the regrouping operation does not change the value of the regrouped number serves a purpose in the above plan only because the regrouping operation is later to be executed within the context of subtraction; outside that context, who cares whether the value of the number remains constant or not? In short, the teaching tactics in a plan are meaningful because of their means-ends relations to other parts of the plan.

The above plan has a number of other noteworthy properties. First, it extends over several lessons, coordinating a large number of what is known as *activity segments* (Leinhardt & Greeno, in press). Second, in all likelihood, the teacher proceeds according to the above plan every time she teaches subtraction with regrouping. The plan can be stored and used over and over again. Third, the plan presupposes a particular view of the subject matter, involving a cut between nonborrowing subtraction and borrowing subtraction and the identification of the regrouping operation as an educationally crucial subskill. This view is likely to be based on experience, rather than on rational analysis of the domain. Fourth, the plan seems to be based on some hypothesis about learning, although it is unclear how to state it. By way of illustration, we might conjecture that the teacher's hypothesis is that *integrating a subskill into its superskill is facilitated if both the subskill and the superskill are mastered before the integration occurs.*

As a second example, let us consider the following low-level plan, which applies when the teacher has discovered that a learner who has been taught the arithmetic of fractions nevertheless does not know how to multiply them correctly. The plan is taken from the teaching materials by (Hill, 1980).

1. Give the student graph paper that is twelve squares wide, and have him cut a strip that is twelve squares long and one square wide.

2. Review the definition of multiplication of whole numbers, explain the meaning of multiplication of fractions, and point out the similarities.

3. Demonstrate the meaning of 1/3 times 3/4 with the help of the strips. This is done as follows:

    a. Take a twelve-squares long strip, and apply 3/4 to it, i. e., cut it to make a strip nine squares long.

    b. Apply 1/3 to the nine-squares long strip, making a strip three squares long.

    c. Count the number of squares in the remaining strip, thus establishing that 1/3 times 3/4 is 3/12.

4. Next, let the student solve the problems 5/6 times 1/2 and 1/4 times 1/3 in the same way.

5. Let the student generate verbal descriptions of these three problems and their solutions.

6. Help the student induce the following rule for multiplying fractions from their own descriptions: "To multiply two fractions, multiply the numerators and then multiply the denominators."

The basic properties of the previous plan are exemplified here as well: the means-ends structure and the existence of steps which would not be meaningful in isolation. But in contrast to the first example, this plan should only take a few minutes to execute. It is worth noticing that the actions in it are specified in detail, including the exact digits used in the examples.

Could we construct an intelligent tutoring system by storing teaching plans in the machine? We can readily imagine a library of such plans culled from observations of what teachers do, from recommendations of what they ought to be doing, and from our own imagination. For each teaching situation, the tutor could retrieve the appropriate plan and execute it.

From the point of view of individualized instruction, the disadvantage of a teaching plan is obvious. Because it consists of a pre-assembled list of actions, the plan has no capacity to adapt itself to the learner. Perhaps some learners acquire the regrouping procedure easier and with more understanding if it is taught in the context of solving subtraction problems. Clearly, what we want to do is to teach regrouping either within the context of subtraction or in isolation, depending upon our beliefs about the student. What is the teacher who uses the second of the above plans supposed to do with the student who, at step 3b in the plan, produces a strip 6 squares long? Clearly, the rest of the plan then becomes irrelevant with respect to that student, but there is no information in the plan about what to do instead. There are no resources within a plan for accommodating changes.

The linear nature of the plan is both its beauty and its flaw. The sequential relations between the actions can encode complicated means-ends relations and so free the tutor from the need of computing those relations at run-time, as it were; but, exactly because they are not computed at run-time, those relations cannot be adapted to the changing cognitive needs of a particular learner. But the idea of a teaching plan should not be abandoned. The advantages of having a coordinating structure which maps out what tactics should be used in the service of a tutorial goal is very powerful. The idea of a plan is not faulty, only incomplete. Teachers do not execute their plans slavishly or without flexibility; they do not need to go on blindly to the plan's bitter end, if there is evidence that the student(s) are not benefiting from it. The idea of a teaching plan should be augmented with the knowledge teachers bring to the execution of their plans, so that the computer tutor can use its plans with the same flexibility.

The question of what is needed in order to use plans flexibly will be taken up again in a later section. The next section deals with the teaching strategies in current computer tutors.

RULES IN THE MACHINE

A second obvious source of inspiration with respect to teaching is the strategies which have been implemented in existing tutoring systems. For instance, the tutors constructed by John Anderson and his co-workers (Anderson, Boyle, & Reiser, 1985; Anderson, Boyle, & Yoost, 1985; Reiser, Anderson, & Farrell, 1985) are successfully implemented, state-of-the-art designs for teaching complicated subject matters like mathematics and programming; they are based on a theory of learning, empirically tested and - it is almost overwhelming - found to produce learning in ecologically valid settings. How do these tutors go about the task of teaching?

In brief, the strategy of the Anderson tutors is to watch the student closely and intervene when he makes an error. Let us see in some detail how this is done in, for instance, the Lisp tutor (Reiser, Anderson, & Farrell, 1985). The activity of the student while interacting with the Lisp tutor is to build a Lisp program (function), i. e., to write and combine calls to other programs (functions) according to the rules of Lisp. At the top level, the interaction between the tutor and the student consists of two parts: The tutor poses a programming task, and the student responds by typing in his solution, i. e., his code. The Lisp tutor has a knowledge base consisting of several hundred problem-solving rules which can solve the Lisp programming task, and several hundred more incorrect rules, representing typical novice programming errors. Every time the student takes a step towards completing his Lisp function, i. e., every time he types in a new part of his code, the tutor tries to map that step onto one of its rules. If the matching rule is a correct rule, the tutor does not intervene; if the matching rule is incorrect, the tutor explains the error. If the student nevertheless cannot continue on the correct solution, the tutor can show what the correct step is, or invoke a planning mode, in which the tutor demonstrates the algorithm to be programmed.

The geometry tutor (Anderson, Boyle, & Yoost, 1985) is built on the same basic design as the Lisp tutor. The student's activity in that case is to construct a geometry proof by successively applying geometric theorems, and the tutor compares his steps to a large knowledge base consisting of both correct and incorrect problem-solving rules. The tutor intervenes when the student applies an incorrect rule, i. e., makes illegal or useless moves in the space of possible inferences.

The teaching task performed by the Anderson tutors might be characterized as *the intelligent monitoring of student activity*. The student is struggling with an intellectual construction, and the tutor helps him complete it. To what extent is the help provided by the tutor adapted to the individual student, and how is this adaptability achieved? With respect to moment-by-moment flexibility, these tutors are hard to beat. The grain-size of the tutor's behavior is determined by the grain-size of the student's activity. For instance, the Lisp tutor can react to each Lisp atom as it is typed in; the geometry tutor can react to the selection of a theorem, or the application of it. Since the tutor decides whether to intervene, and, if so, how to intervene, after each action on the part of a the student, the flexibility of the tutor could not be greater without becoming irrelevant, as their designers point out (Reiser, Anderson, & Farrell, 1985, p. 12).

The question of whether the *content* of instruction is adapted to the cognitive needs of the learner is more complicated to answer. Recall that the tutor responds to each action on the part of the learner by mapping his action onto a problem-solving rule, and then retrieving the appropriate tutorial action connected with that rule. (For the sake of uniformity, I regard "doing nothing" as a kind of action.) This means that the student draws out of the tutor, as it were, a particular sequence of tutorial messages by generating a particular sequence of rule applications. Since the tutor knows several hundred different rules, the number of distinct lessons it is capable of delivering is astronomical. At ***the level of the lesson***, then, the adaptation to the student is impressive.

However, this adaptivity is achieved with the help of predefined units. Each incorrect problem-solving rule is paired with a particular tutorial action, typically a stored message.[10] Any student who takes a step which matches a particular incorrect rule receives the message associated with that rule. For instance, a geometry student who applies a congruence theorem incorrectly for the first time receives the same instruction as a student who makes the same error, but who have also made a large number of other errors in applying congruency theorems. There is no on-line design of a tutorial message on the basis of the entire model of that student, i. e., on the entire set of rules he has used, rather than on a single rule. At ***the level of the single tutorial action***, then, there is no adaptation to the current cognitive state of the learner other than the classification of his last step as an instance of a particular type of error.

[10]**The messages contain variables which can be bound to the objects in the particular situation in which they apply. Strictly speaking, then, what is prestored is not the surface form of the message, but its tutorial gist.**

It is instructive to consider how the Anderson architecture could be extended to allow tutors to base their tutorial decisions on more information about the student than his last step. The obvious solution is to write tutorial rules which test for a *sequence* of steps on the part of the student, e. g., "if the student has applied rules x, y, and z, then given him message A," "if x, y, w, then message B," etc. The fallacy in this solution becomes apparent when we consider the fact that the tutor knows many hundreds of rules. A set of, say, 400 rules implies 1600 two-rule sequences, 640,000 three-rule sequences, etc. The number of tutorial rules quickly grows beyond practical limits. The more radical solution is to dissolve the prestored links between the step-classifications, the rules, and the tutorial actions, the messages, and replace them with intelligent agents which compute what the appropriate tutorial message is, given a particular error by a particular student in a particular problem-solving situation. However, this solution is equally fallacious, because the term "intelligent agents" hides within it all the problems of intelligent tutoring. If we knew how to design such agents, we would construct one to *replace* the tutors John Anderson and his co-workers have constructed. These two fallacious extensions illustrate, I think, the boundaries of the Anderson tutoring architecture.

In summary, the Anderson tutoring architecture is geared towards the intelligent monitoring of practice, and it is build around performance-instruction pairs, where the performance part consists of a mechanism for classifying a student's problem-solving steps as instances of particular errors and the instruction part consists of prestored tutorial messages relevant to those errors. Given a large data base of such performance-instruction pairs, a tutor delivers a lesson by responding to each error on the part of the student as it occurs in the problem-solving context. The question of how well these tutors adapt to the student receives different answers, depending upon the level at which the question is asked. At the level of a particular (incorrect) problem-solving step, each student recieves the same tutorial message; at the level of an entire problem-solving attempt, each student recieves a unique sequence of tutorial messages, as a function of his own activity.

Other projects in the field of intelligent tutoring research have proposed strategic rules for instruction as well. For example, the tutoring rules of the GUIDON system (Clancey, 1982; 1983) are different in character from performance-instruction pairs, because many of them, rather than recommend specific tutorial actions, revise the quantitative parameters of the system, which, presumably, has global effects on how GUIDON carries out its tutorial dialogue. The author's statement of the "set of tutoring principles that appear implicitly in the tutoring rules" (Clancey, 1983, p. 6) does not explicitly treat the question of how to adapt the content of the instruction to the individual. His principles deal with global aspects - attitudes, as it were - of the tutor. As an example, consider the principle, "provide orientation to new tasks by top-down refinement." Such a principle does not tell us what to do with students who do not benefit from top-down refinement. In short, the adaptability of the GUIDON system seems to be of the same nature as the adaptability of the Anderson tutors: Each student gets a unique lesson by drawing out a unique sequence of locally determined responses from the tutor. As a second example, consider the eleven pedagogic principles which Burton (1982, pp. 90-92) propose in connection with the WEST tutoring system. They, too deal with global aspects, attitudes, on the part of the tutor, e. g., "Do not intervene on two consequetive problem-solving steps, no matter what" (Principle 5). The rule which comes closest to have something interesting to say about how to shape the instruction is Principle 2: "If the student made a bad problem-solving step, the tutor should present an alternative step only if there exists an alternative step which is dramatically superior to the step the student took." In general, these rules seem rather far removed from the teaching plans we discussed in the previous subsection.

TEACHING AS PROBLEM SOLVING

The notion of a teaching plan and the notion of a local performance-instruction unit are equally unsatisfactory as a basis for teaching. The reason for this is that neither notion helps us understand how a teaching goal (e. g., "explain borrowing") becomes translated into a sequence of teaching tactics designed to satisfy that goal. (e.g., "mark borrowing problems," "prove invaliance under regrouping," etc. Neither construct makes explicit the means-ends relations which connect what a teacher does with what he wants to achieve.

But understanding the connection between goals and actions is a prerequisite for the construction of artificial teachers capable of delivering adaptive instruction. As a conjecture, I propose that an intelligent tutor must have the following capabilities. First, it must be able to decompose its goals into subgoals. Second, it must be able to access descriptions of the student and of the subject matter and on their basis generate a plan for how to satisfy the tutoring goal. Third, it must be able to execute such a plan. Fourth, it must be able to detect mismatches between a student and a plan. As mentioned in the discussion of diagnosis, a teaching plan contains implicit expectations about what the student will do. When these expectations are violated, the plan is not quite right for that student. Finally, a tutor must be able to revise a plan, or generate a completely new plan for a particular tutoring goal. In summary, we should think of tutoring, I suggest, as proceeding through cycles, each cycle consisting of plan generation, plan execution, and plan revision. According to this view, the coordination of tutoring stems from a tutor's ability to follow a plan; the flexibility of tutoring stems from its ability to revise a plan.

> **The Principle of Teaching Plans.** A tutor needs to be able to generate a teaching plan on the basis of its representation of the student, its knowledge of the subject matter, and its current tutorial goal; furthermore, it should be able revise its plan if it discovers that the plan does not fit the student.

This principle is not a teaching strategy, but it describes the kind of strategies I suggest we should be trying to incorporate into computer tutors. Whether tutors with such capabilities produce more learning than other kinds of tutors is not known. The above principle is only a *design hypothesis.*

Of course, the kind of system I have outlined in the above design hypothesis is entirely familiar: it is a *problem solver*. The topics of subgoaling, of plan generation, and plan revision are all familiar items in the discussion of problem solving. Thus, my suggestion could be restated by saying that a teacher needs a strategy for how to solve the problem of bringing the student from a state of ignorance to a state where he has acquired the subject matter. What research is relevant to the construction of such strategies?

The literature on formal analyses of problem-solving strategies in general is of course extensive (e. g., Pearl, 1984). But with respect to teaching strategies, in particular, the pioneering work of Gaea Leinhardt is the only available analysis. Leinhardt and Greeno (in press) present a general analysis at the skill of teaching. Leinhardt and Smith (1985) have applied the notion of a planning net (VanLehn & Brown, 1980) to the phenomenon of teaching, showing how one can generate a particular teaching plan from a structure of goals and a set of teaching tactics. This is the kind of generative mechanism that an intelligent tutoring system needs.

The theoretical analysis of teaching strategies presuppose detailed empirical studies of the strategies used by teachers. The studies by Leinhardt (1985), Leinhardt and Smith (1985), and Leinhardt and Greeno (in press) have already been mentioned. As a second instance of the kind of empirical work we need, let me mention the study by Collins and Stevens (1982) of rules for guiding Socratic dialogues in geography. These rules make recommendations about what kind of case the tutor should present and what kind of question to ask in response to particular properties of the current dialogue situation, e. g., "if one or more factors (such as heavy rainfalls) have been identified with respect to a particular value of the dependent variable (such as amount of rice grown), then ask if those factors are necessary conditions for that value or not." These strategies were identified in detailed studies of expert teachers.

In summary, I suggest that we view a one-on-one tutor as a problem solver who is trying to execute a teaching plan which will help the student to progress in his learning, constantly updating the plan on the basis of the student's performance. The theoretical research needed to implement this view of tutoring systems is to invent and formalize problem-solving strategies which can connect tutoring goals with teaching tactics. Such analyses, in turn, would be much helped if we had available, as a source of inspiration, a large collection of field-tested teaching plans. The empirical research needed to progress towards intelligent tutoring systems consists of detailed studies of expert teachers.

Conclusion

The present discussion opened with the claim that the promise and the challenge of computer tutors lie in their potential for providing instruction which is dynamically adapted to the learner. What kind of tutor is capable of delivering such instruction? Let us recapitulate the main course of the argument.

A tutoring system must have an internal representation of the student, in order to adapt its teaching to him. However, a student can be described in many ways. The only way to decide what kind of description is needed is to consider the function of the description within the tutoring system. The idea was advanced that the function of diagnosis is to test the viability of expectations implicit in teaching plans, which implies that the design of a diagnostic component depends upon the teaching strategy of the system.

A tutoring system must also have an internal representation of the subject matter, because in order to adapt its teaching to the individual learner, it must be able to generate the particular presentation of the subject matter which suits the learner best. The main difficulty in the representation of subject matter is that the learner can acquire many different internal representations of the subject matter, all equally satisfactory from an educational point of view. This ambiguity in where the learner is going causes difficulties for both the diagnosis of the student and the representation of the subject matter. There is no principled solution to this problem yet.

A tutoring system must, in order to be able to adapt its teaching to the individual learner have a large number of teaching tactics available, so that it can choose, at any one moment, that tactic which will benefit the learner most. Thus, in principle, the richer the behavioral repertoire of the tutoring system, the more adaptive it becomes.

Finally, in order to adapt its instruction to the individual learner, a tutor must be able to compute means-ends relations between its goals and its actions, and to make those relations dependent upon its representation of the student. From this perspective, a tutor is a problem solver, a point of view which has been most clearly expressed by Leinhardt and Greeno (in press). Like other artificial problem solvers, a tutoring system should be build around its strategy. Precise teaching strategies can be defined by applying formal analyses to detailed observations of expert teachers. The ultimate conclusion of the present analysis is that in order to construct artificial teachers, we must first discover how to teach.

References

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

Anderson, J. R., Boyle, D. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science, 228,* 456-462.

Anderson, J. R., Boyle, C. F., & Yost, G. (1985, August). The geometry tutor. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 1-7).

Attisha, M., & Yazdani, M. (1983). A micro-computer based tutor for teaching arithmetic skills. *Instructional Science, 12,* 333-342.

Attisha, M., & Yazdani, M. (1984). An expert system for diagnosing children's multiplication errors. *Instructional Science, 13,* 79-92.

Bangert, R. L., Kulik, J. A., & Kulik, Chen-Lin, C. (1983). Individualized systems of instruction in secondary schools. *Review of Educational Research, 53,* 143-158.

Beede, R. B. (1985, October). Dot method for renaming fractions. *Arithmetic Teacher,* 44-45.

Bell, A. W., Costello, J., Kuchemann, D. E. (1983). *A review of research in mathematical education. Part A. Research on learning and teaching.* Berks, U.K.: NFER - Nelson.

Brachman, R. J., & Levesque, H. J. (Eds.). (1985). *Readings in knowledge representation.* Los Angeles: Morgan Kaufmann.

Brown, J. S., & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science, 2,* 155-192.

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

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

Carr, B., & Goldstein, I. (1977). *Overlays: A theory of modeling for computer-aided instruction. (Tech. Rep. A.I. Memo 406.* Cambridge: MIT.

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

Clancey, W. J. (1983). Guidon. *Journal of Computer-Based Instruction, 10,* 8-15.

Collins, A., & Stevens, A. L. (1982). Goals and strategies of inquiry teachers. In R. Glaser (Ed.), *Advances in instructional psychology* (Vol. 2, pp. 65-119). Hillsdale, NJ: Erlbaum.

Gagne, R. M. (1962). The acquisition of knowledge. *Psychological Review, 69,* 355-365.

Gardner, M. K. (1985). Cognitive psychological approaches to instructional task analysis. In E. W. Gordon (Ed.), *Review of Research in Education* (Vol. 12, pp. 157-195). Washington, DC: American Educational Research Association.

Genter, D. & Stevens, A. L. (1983). *Mental models.* Hillsdale, NJ: Erlbaum.

Ginsburg, H. P. (1983). Cognitive diagnosis of children's arithmetic. *Issues in cognition: Proceedings of a Joint Conference in Psychology* (pp. 287-300). Washington, DC: National Academy of Sciences and American Psychological Association.

Glaser, R. (1976). The processes of intelligence and education. In L. B. Resnick (Ed.), *The nature of intelligence* (pp. 341-352). Hillsdale, NJ: Erlbaum.

Glaser, R. (In press). The integration of testing and teaching. In *The redesign of testing for the 21st century. The Proceedings of the 1985 ETS Invitational Conference.* Princeton, NJ: Educational Testing Service.

Grinder, R. E., & Nelsen, E. A. (1985). Individualized instruction in American pedagogy: The saga of an educational ideology and a practice in the making. In M. C. Wang & H. J. Walberg (Eds.), *Adapting instruction to individual differences* (pp. 24-43). Berkeley, CA: McCutchen.

Hill, M. (1980). *Diagnosis. An instructional aid.* Chicago: Science Research Association.

Johnson, W. L., & Soloway, E. (1983, August). *PROUST: Knowledge-based program understanding* (Report No. YaleU/CSD/RD#285). New Haven, CT: Yale University, Dept. of Computer Science.

Johnson, W. L., & Soloway, E. (1984). Intention-based diagnosis of programming errors. *Proceedings of the AAAI Conference* (pp. 162-168).

Kieren, T. E. (1976). On the mathematical, cognitive, and instructional foundations of rational numbers. In R. A. Lesh (Ed.), *Number and measurement* (pp. 101-145). Columbus, OH: ERIC Clearinghouse for Science, Mathematics, and Environmental Education.

Kieren, T. E. (1980). The rational number construct - its elements and mechanisms. In T. E. Kieren (Ed.), *Recent research on number learning* (pp. 125-149). Columbus, OH: ERIC Clearinghouse for Science, Mathematics, and Environmental Education.

Langley, P., Ohlsson, S., & Sage, S. (1984). *A machine learning approach to student modeling* (Tech. Rep. CMU-RI-TR-84-7). Pittsburgh: Carnegie-Mellon University, The Robotics

Institute.

Leinhardt, G. (1985, March). *The development of an expert explanation: An analysis of a sequence of subtraction lessons.* Paper presented at the meeting of the American Educational Research Association, Chicago.

Leinhardt, G., & Greeno, J. G. (in press). The cognitive skill of teaching. *Journal of Educational Psychology.*

Leinhardt, G., & Smith, D. A. (1985). Expertise in mathematics instruction: Subject-matter knowledge. *Journal of Educational Psychology, 77,* 247-271.

Marshall, S. P. (1980). Procedural networks and production systems in adaptive diagnosis. *Instructional Science, 9,* 129-143.

Marshall, S. P. (1981). Sequential item selection: Optimal and heuristic policies. *Journal of Mathematical Psychology, 23(2),* 134-152.

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

Nitko, A. J. (in press). Designing tests that are integrated with instruction. In R. L. Linn (Ed.), *Educational measurement* (3rd ed.).

Ohlsson, S., & Langley, P. (in press). Psychological evaluation of path hypothesis in cognitive diagnosis. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring.* New York: Springer Verlag.

Ohlsson, S., & Langley, P. (1985). *Identifying solution paths in cognitive diagnosis* (Tech. Rep. CMU-RI-TR-85-2). Pittsburgh: Carnegie-Mellon University, The Robotics Institute.

Pearl, J. (1984). *Heuristics. Intelligent search strategies for computer problem solving.* Reading, MA: Addison-Wesley.

Prietula, M., & Marchak, F. (1985, August). Expert variance: Differences in solving a dynamic engineering problem. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 335-340).

Reiser, A. J., Anderson, J. R., & Farrell, R. G. (1985, August). Dynamic student modelling in an intelligent tutor for Lisp programming. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 8-14).

Resnick, L. B. (1973). Hierarchies in children's learning: A symposium. *Instructional Science, 2,* 311-362.

Resnick, L. B., & Wang, M. C. (1969, June). Approaches to the validation of learning hierarchies. *Proceedings from the Eighteenth Annual Western Regional Conference on Testing*

Problems. San Francisco.

Resnick, L. B., Wang, M. C., & Kaplan, J. (1973). Task analysis in curriculum design: A hierarchically sequenced introductory mathematics curriculum. *Journal of Applied Behavior Analysis, 6*, 679-710.

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

Sleeman, D., & Brown, J. S. (Eds.). (1982). *Intelligent tutoring systems*. London: Academic Press.

Smedslund, J. (1969). Psychological diagnostics. *Psychological Bulletin*, 71, 237-248.

Soloway, E., Rubin, E., Woolf, B., Bonar, J., & Johnson, W. L. (1982, December). *Meno-II: An AI based programming tutor*. (Research Rep. #258). New Haven, CT: Yale University, Computer Science Dept.

Steiner, H. G. (1969). Magnitudes and rational numbers: A didactic analysis. *Educational Studies in Mathematics, 2*, 371-392.

Svenson, O., & Hedenborg, M.-L. (1980). Strategies for solving simple subtractions as reflected by childrens verbal reports. *Scandinavian Journal of Educational Research*. 24, 157-172.

Thurstone, H. A. (Ed.). (1956). *The number system*. New York: Dover Publications.

VanLehn, K. (1983). *Felicity conditions for human skill acquisition: Validating an AI-based theory* (Tech. Rep. No. CIS-21). Palo Alto, CA: Xerox PARC.

VanLehn, K., & Brown, J. S. (1980). Planning nets: A representation for formalizing analogies and semantic models of procedural skills. In R. E. Snow, P.-A. Frederico, & W. E. Montague (Eds.), *Aptitude, learning and instruction: Vol 2. Cognitive process analysis and problem solving* (pp. 95-137). Hillsdale, NJ: Erlbaum.

Wang, M. C., & Walberg, H. J. (Eds.). (1985). *Adapting instruction to individual differences*. Berkeley, CA: McCutchan.

Waterman, D. A. (1986). *A guide to expert systems*. Reading, MA: Addison-Wesley.

Waterman, D. A., & Newell, A. (1972). *Preliminary results with a system for automatic protocol analysis* (CIP Rep. No. 211). Pittsburgh: Carnegie-Mellon University, Department of Psychology.

Wenger, E. (1985). *AI and the communication of knowledge: An overview of intelligent tutoring systems*. Irvine: University of California, Irvine Computational Intelligence Project, Department of Information and Computer Science.

Williams, M. D., & Hollan, J. D. (1981). The process of retrieval from very-long term memory. *Cognitive Science, 5*, 87-119.

Woolf, B., & D. D. McDonald. (1984, August). Context-dependent transitions in tutorial diaglogues. *Proceedings of the National Conference on Artificial Intelligence* (pp. 355-361). University of Texas at Austin.

Young, R. M. (1976). *Seriation in children. An artificial intelligence analysis*. Basel, West Germany: Birkhauser Verlag.