1

ARI Research Note 90-90

Pedagogical Strategies for Human and Computer Tutoring

Brian J. Reiser

Princeton University

for

Contracting Officer's Representative
Judith Orasanu

Basic Research
Michael Kaplan, Director

August 1990

FIFTY YEARS OF SERVICE
1940 - 1990

United States Army
Research Institute for the Behavioral and Social Sciences

Research accomplished under contract
for the Department of the Army

Princeton University

Technical review by

George Lawton

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS --
2a. SECURITY CLASSIFICATION AUTHORITY -- 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE --	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) --	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARI Research Note 90-90
6a. NAME OF PERFORMING ORGANIZATION Princeton University 6b. OFFICE SYMBOL (If applicable) --	7a. NAME OF MONITORING ORGANIZATION U.S. Army Research Institute
6c. ADDRESS (City, State, and ZIP Code) Cognitive Science Laboratory 221 Nassau Street Princeton, NJ 08542	7b. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600
8a. NAME OF FUNDING / SPONSORING ORGANIZATION U.S. Army Research Institute for the Behavioral and Social Sciences 8b. OFFICE SYMBOL (If applicable) PERI-BR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-87-K-0652
8c. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102B; PROJECT NO. 74F; TASK NO. N/A; WORK UNIT ACCESSION NO. N/A

11. TITLE (Include Security Classification)
Pedagogical Strategies for Human and Computer Tutoring

12. PERSONAL AUTHOR(S)
Reiser, Brian J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 87/07 TO 90/06	1990, August	35

16. SUPPLEMENTARY NOTATION
Contracting Officer's Representative, Judith Orasanu

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Intelligent tutoring system; Computer; Problem solving; Instruction

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report considers the pedagogical strategies of human tutors and examines the implications of this work for research in intelligent tutoring systems. It briefly describes GIL, an intelligent tutor for simple LISP programming, and considers its effectiveness from the perspective of human tutoring strategies. Finally, it discusses the implications of research on human tutoring for the design of intelligent tutoring systems for problem solving.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL George Lawton	22b. TELEPHONE (Include Area Code) (202) 274-8722 22c. OFFICE SYMBOL PERI-BR

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

PEDAGOGICAL STRATEGIES FOR HUMAN AND COMPUTER TUTORING

Brian J. Reiser
Princeton University

Abstract

In this paper, I consider the pedagogical strategies of human tutors and examine the implications of this work for research in intelligent tutoring systems. I will briefly describe GIL, a intelligent tutor for simple LISP programming, and consider its effectiveness from the perspective of human tutoring strategies. Finally, I will discuss the implications of research on human tutoring for the design of intelligent tutoring systems for problem solving.

1 Introduction

The promise of modern educational computing has always been the hope of capturing the pedagogical advantages of effective instruction and delivering it, via computers, to students. There are a number of potential models available into which the computer might be accommodated. Some have argued, for example, that the best use of computer in instruction is to provide a laboratory for exploration and discovery. Alternatively, the computer might be used as a collaborator for problem solving, to provide the student with a cooperative learning environment. There are important reasons to explore the pedagogical benefits of discovery learning (e.g., Bruner, 1961; Papert, 1980) and of cooperative learning (Slavin, 1983). A promising model for the interaction between the computer and student that has received much recent attention is drawn from the tutoring situation. Indeed, individualized instruction is often though to be the most effective form of instruction, particularly for problem solving domains (e.g., Bloom, 1984; Cohen, Kulik, & Kulik, 1982).

What are the pedagogical benefits of individualized tutoring? Is it possible to capture these advantages and embed them in a computer system? In this paper, I discuss the strategies employed by human tutors in problem solving domains and examine whether these techniques can be incorporated into computerized tutors. I will describe an intelligent tutoring system for programming that has been demonstrated to be effective, and consider how it achieves the pedagogical goals of human tutors teaching the same material. I will consider the problems for computer-based instruction, and the ways in which it may have advantages over human tutors. Finally, I examine several current controversies in intelligent tutoring systems research from the viewpoint of the pedagogical strategies of human tutors.

2 Pedagogical Strategies in Human Tutors: Why are human tutors effective?

A number of studies have documented the pedagogical effectiveness of human tutors (e.g., Bloom, 1984; Cohen, Kulik, & Kulik, 1982), confirming a common intuition that when a student has difficulty, the best course of action is to provide the student with one-on-one instruction. Studies of individualized tutoring in a variety of domains have demonstrated effects on both learning time and subsequent performance. Several recent studies have begun to analyze the pedagogical strategies of tutors to ascertain what underlies their effectiveness (e.g., Fox, 1988; Lepper, Aspinwall, Mumme, & Chabay, in press; Lepper & Chabay, 1988; Putnam, 1987). These studies begin to suggest why tutoring is so effective. Experienced human tutors tread a delicate balance in order to provide students with enough guidance and help to keep them from becoming frustrated or confused, yet allow the student to do as much of the work as possible and feel in control.

Learning by doing is much more effective than learning by observation (e.g., Anderson, 1983; Anzai & Simon, 1979). Yet this type of learning has its pitfalls, both cognitive and motivational. Students trying to solve problems can expend much time and effort pursuing blind allies due to errors and poor strategies. Of course, in some cases students may learn something valuable while searching for a solution. In many cases, however, such episodes can leave students confused and frustrated, and it may be difficult to return to the point in the problem solving before the error occurred. The assistance of a tutor enables a type of "guided learning by doing," in which the students reap the rewards of active problem solving while the tutors minimize the costs. In this way, tutoring has both cognitive and motivational advantages.

Lepper and his colleagues have characterized the impact of tutoring on the motivation of students (Lepper et al., in press; Lepper & Chabay; 1988). They argue that tutors manage to promote a sense of challenge, provoke curiosity, and maintain the student's feeling of control. Fox (1988) provides a detailed analysis of tutorial interactions, as evidenced by characteristics of the discourse interaction. She argues that tutors provide a "safety net" during problem solving, so that student errors are kept to a minimum. She demonstrates that tutors employ subtle techniques to notify students that a step in the solution needs to be repaired. Tutors provide frequent feedback, typically indicating very briefly their agreement with each step. A short hesitation (less than 1 second) in responding with confirmation leads the student *to assume that something is wrong* with the current step, and frequently students can then correct the mistake. If more explicit help is required, the tutor focuses the student's attention on the part of the solution that requires modification, or on information potentially useful for repairing the error. Tutors avoid telling students they are wrong, and avoid telling them precisely in what manner a step is incorrect; instead they try to lead students to discover the error themselves. The effect is a guided problem solving session, in which the student takes steps and fixes wrong paths, while the tutor helps the student stay on track. In some cases, students do request guidance in the form of goals to set, filling in of missing information, or explicit confirmation of the correctness of a step, but in most cases such requests are not necessary because the tutor provides such information through hints, leading questions, verbal agreement, and other methods.

In summary, human tutors moderate their control of the interaction to provide sufficient assistance for the student to solve the problem, while being careful to avoid leading the student too much. Tutors provide enough hints to help students plan and effect a solution, and use error feedback and hints to prevent students from reaching situations from which it would be too difficult to recover. Students discover their own errors, but rely on the tutor's continual feedback for monitoring their problem solving and to direct their focus of attention on the relevant aspects of the solution. This control is indirect, however, so that the students maintain their feeling of being in control. Indeed, the feeling of control over the domain is enhanced by the tutors, who enable the students to solve problems and learn by doing but without the dangers of getting lost and frustrated.

In this way, tutoring has many of the advantages of cognitive apprenticeship (Collins, Brown, & Newman, 1989). The tutors can model expert solutions to a problem so that students can build a conceptual model of the processes required for a task. The tutor's coaching and feedback provides a scaffolding so that the tutor's assistance supports the student's problem solving. In addition, the interaction with tutors requires students to articulate their reasoning, enabling them to focus more on the process of solving the problems, rather than on the products or solutions themselves.

3 Observations of Human Tutors in the Programming Domain

My colleagues and I have studied human tutors in a variety of contexts to provide comparisons for our computer-based instructional systems. Our studies have examined students learning to write computer programs. We have compared students learning on their own and in pairs, students learning with human tutors, and students learning with the aid of intelligent tutoring systems. In this section I will summarize some of the findings concerning the effectiveness of human tutors teaching programming. In the following sections I will then compare these findings to results with our intelligent tutoring system for programming.

In one study, currently in progress, we examined students learning to program in a variety of learning situations (Reiser, in preparation). All subjects read a textbook introducing them to LISP programming (Anderson, Corbett, & Reiser, 1986). Subjects read the first three chapters, which covered the topics of using LISP functions to manipulate numbers and lists, defining new functions, and writing programs using predicates, conditionals, and logical functions. Subjects read the text and solved the problems in each chapter, using a LISP interpreter and a simplified screen editor to modify their programs. There were 26, 12, and 13 problems in the three chapters, respectively. Subjects worked on the material at their own pace, and took 3 to 5 sessions lasting from 2 to 4 hours each to work through all the text and problems.

The study included three learning conditions: No Tutor, Tutor, and Consultant.[1] The No Tutor subjects were instructed to read the text and solve the problems on their own, and ask questions only if necessary. At the end of each chapter, these subjects presented their solutions to the experimenter, who graded each solution as correct or incorrect, whereupon subjects were given an opportunity to attempt to correct their mistakes. In contrast, the Tutor subjects worked through the same material with an experienced human tutor. In the Consultant condition, the tutors were present only a few feet away in an adjacent room in the laboratory suite, within sight of the subjects. Subjects were instructed to ask as many questions as they wished of the human tutors. When asked, the tutor would sit down next to the subject and help with whatever problem or confusion the student had encountered, until the tutor felt the topic was completed, typically 5-10 minutes later. The purpose of this condition was to compare a natural tutoring situation with one in which the tutoring interactions were initiated only by the student. We employed two tutors, one male and one female, both Princeton undergraduates. Subjects were matched with the tutor of the same gender in the Tutor and Consultant conditions.

The results demonstrate a dramatic advantage for the Tutor subjects. The subjects completed the material approximately 40% faster than the No Tutor subjects, and completed the problems with only one-third the number of solution attempts. Learning times for the Consultant subjects fell between these two groups, slower than the Tutor subjects but faster than the No Tutor subjects. Interestingly, although the subjects asked relatively few questions, (approximately 3 questions per chapter, or 1 question every 45 minutes), this help at presumably crucial points in the subject's problem solving resulted in faster learning than the No Tutor subjects.

Our examination of the tutoring protocols provides further support for the view of tutors as safety nets for learning by doing. Although the solutions to these programming problems were typically programs no longer than 5-10 lines, many were relatively difficult and could require up to 45 minutes to

[1] The study also included a fourth condition, Collaborative Learning, in which students worked in pairs. We focus in this paper on the effects of tutoring, so we will use the No Tutor subjects as the control group for comparison with the Tutor and Consultant groups, and will not consider the Collaborative Learning group here. For a full presentation of this learning experiment, see Reiser (in preparation).

solve. The sessions were highly interactive, as tutors helped the students set goals, pointed out errors, provided guidance in fixing the errors, and provided hints for steps in the solution. Consistent with Fox's (1988) observations, the students relied on continual feedback from the tutors. Tutors reacted to each step with confirmations, questions, prodding, asking for justifications, etc. In most cases, the tutor's feedback, although indirect, enabled subjects to quickly determine whether a solution path was correct or likely to succeed. The tutors, through questions and hints, were able to focus students on the part of a solution that needed elaboration or repair.

The tutors' feedback varied in its timing and content. In some cases, the tutors just prompted the student to rethink a step, whereupon the student would initiate a repair of the partial solution. In other cases, the tutors interrupted to tell the students something that was missing or incorrect. In contrast, the tutors also let some types of errors go, returning to them at the end of the solution when the student was ready to check the program by running it. Thus, the tutors appeared to modulate their responses depending upon the potential learning consequences of the error. Tutors quickly corrected errors that would be distracting and might lead to floundering, and did not comment during the solution on errors that might lead to productive learning episodes later. Instead, they made sure that the subject discovered the error at an appropriate point, and then helped to diagnose the problem and fix the solution. Lepper (1989) has also discussed this type of modulation of response based on potential learning consequences. Lepper found that tutors interrupt upon "disruptive" errors and let the students continue after "productive" errors without comment until the initial solution was completed.

Through hints, leading questions, and continual confirmatory feedback, the tutors guided the students' problem solving, and prevented the students from reaching error states from which it would be too difficult to recover. This enabled the students to master the material more quickly than students working on their own. Our current analyses are focused on characterizing these different timing and feedback situations. I will return to the strategies exhibited by these human tutors in the evaluation of the behavior of intelligent tutoring systems in Section 5.

4 What Pedagogical Benefits of Tutoring Can be Achieved in a Computer Tutor?

Human tutors are clearly doing very careful monitoring of students' problem solving, and providing very subtle feedback. Can this type of reasoning be modeled in a computer tutor? Are there perhaps other potential advantages of computer-based instruction that are not evident in human tutors? In this section, I describe and evaluate our work on intelligent tutoring systems for programming from the perspective of human tutoring studies.

4.1 GIL: An Intelligent Tutoring System for Programming That Explains Its Reasoning

Anderson and his colleagues have developed the *model tracing* methodology for intelligent tutoring systems (Anderson, Boyle, & Reiser, 1985). The methodology is designed to guide students as they learn to solve problems in the target domain. A model tracing tutor contains a problem solving model encoding the target skills in the curriculum and a diagnosis component that tracks students' reasoning. This type of tutoring system provides instruction in the context of problem solving by monitoring a student's solution and providing feedback when the student requests guidance or demonstrates a misconception. The tutor analyzes each step as it is taken to determine whether it is on the path toward a solution or indicates a misconception. The student's step is analyzed by comparing it with the rules currently considered by the tutor's problem solving model, called the "ideal student model." If the student's action is one that would be produced by executing one of the rules considered by the ideal student model at that point in the problem, the model applies that rule, thus following the student's path through the problem. Following such a correct step, the tutor is silent and permits the student to continue. Alternatively, if the student's action does not correspond to a correct step, the tutor considers its catalog of buggy rules, which represent general patterns of errors. Errors are diagnosed when the student's step matches the action of a buggy rule, whereupon the tutor interrupts with the advice associated with the rule.

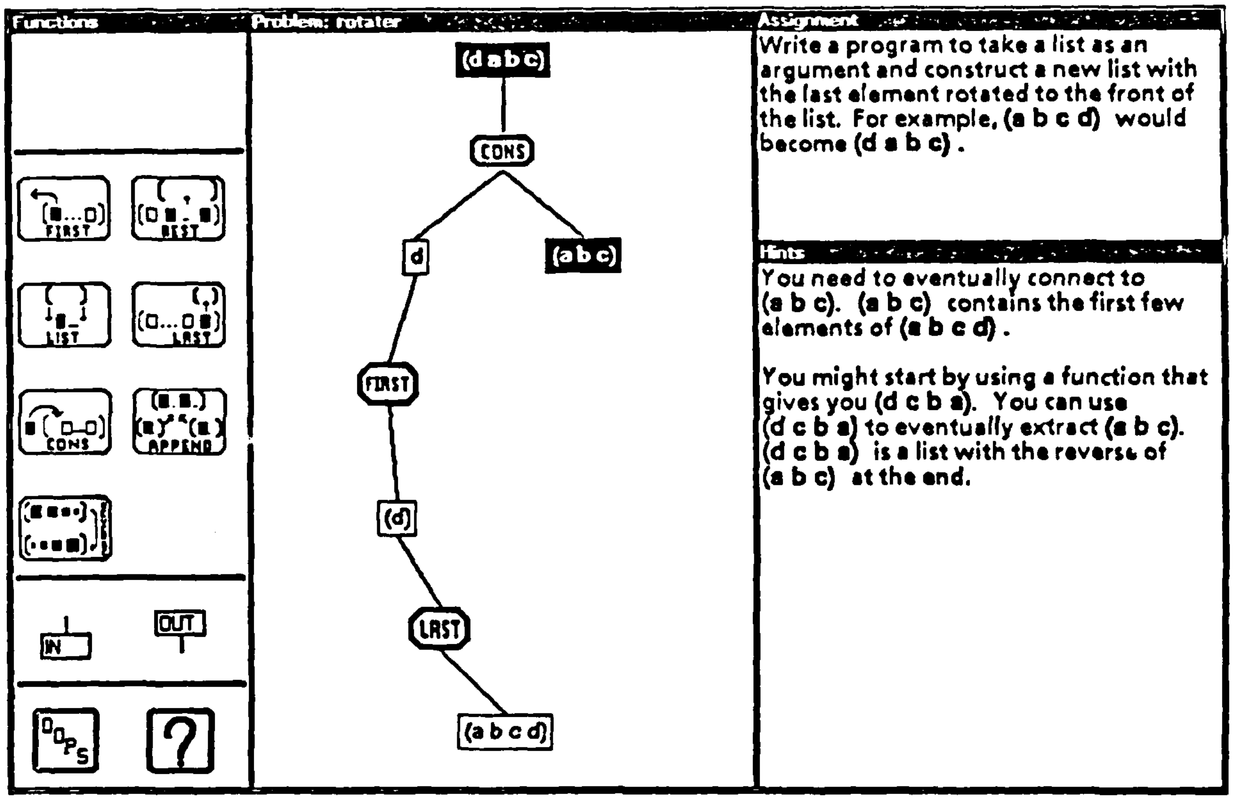
The model tracing tutor understands each step the student takes and stays in the background when the student is following a path leading to a correct solution, but upon request or an erroneous step, it provides a hint or the next step in the solution, enabling the student to continue. For example, the CMU LISP Intelligent Tutoring System helps students learn to write LISP programs by providing feedback as students compose a program to solve a problem (Anderson & Reiser, 1985; Corbett, Anderson, & Patterson, 1988; Reiser, Anderson, & Farrell, 1985). The system monitors each word a student enters in a function definition, matching the step taken by the student against the possible next steps suggested by the tutor's problem solver, providing immediate feedback when an error is diagnosed as well as hints concerning overall strategies and possible next steps when requested.

The model tracing methodology is quite effective in providing online feedback for students as they solve problems. Our current research extends the capabilities of such model tracing tutors by building a system that can construct explanations directly from its problem solving knowledge, rather than relying on canned text associated with the rules. Our research group has constructed an intelligent programming tutor called GIL, *Graphical Instruction in LISP*, that responds to student errors and requests for guidance by finding the relevant problem solving rules and plans and generating explanatory feedback to guide the student's *reasoning* (Reiser, Friedmann, Kimberg, & Ranney, 1988; Reiser, Kimberg, Lovett, & Ranney, in press). GIL contains a problem solving model designed to make explicit the causal knowledge about programming operations, and an explanation component that constructs hints and error feedback directly from the content of its problem solving knowledge. If an error in a student step is found, GIL's explainer analyzes the discrepancies between the student's step and the closest matching correct rule and offers suggestions to the student about how to improve the step. Explanations may draw upon the problem solving rule, general knowledge about the operator being used, and the higher-level plan of which the step is a part.

The GIL problem solver consists of a set of reasoning rules and abstract plans. Each rule contains a description of the properties of the intermediate products in the solution that the step creates. The tutor needs to understand how LISP transforms the data at each step so that it can guide the student in reasoning through a new plan. For example, it is not sufficient for the rules to encode the knowledge that taking the `first` of a reversed list will return the last element; the rules must also represent the reasoning in the algorithm, that is, that reversing the list results in the last element being moved into the first position, enabling the use of `first` to extract that element. To represent this knowledge, each rule describes the properties that are true of the input and output data for that step. Thus, the problem solver not only knows what step to take, but also knows how each step changes the data, and therefore why the step is effective. When the rule is applied in a solution, these properties are added to the current problem state as new inferences about the problem. These properties are then used to select further steps involving the object, and can be used to explain why a step is strategic or to explain why a student's step is in error.

The problem solving plans in GIL represent important sequences of steps to achieve a particular type of subgoal. Many of GIL's rules either initiate a plan or are applied only if a particular plan has been undertaken. Thus, GIL can use these plans to explain the larger context in which a step occurs. Recognizing which plans students are following in a problem is an important prerequisite for providing sensible feedback (Bonar & Cunningham, 1988; Johnson, 1986).

GIL contains a graphical programming interface that is structured to take advantage of the problem solving knowledge of intermediate products (Reiser, Friedmann, Gevins, Kimberg, Ranney, & Romero, 1988). GIL students build a program by connecting together objects representing program constructs into a graph, rather than by defining LISP functions in their traditional text form. Students take a step in GIL by selecting a LISP function and specifying its input and output data, making explicit assertions about the changing state of the program's data. A completed program in GIL consists of a graph specifying how a chain of LISP functions transforms the input data to achieve a particular type of output. A partially complete solution, including a requested hint from GIL about how to proceed next, is shown in Figure 1. The use of intermediate products makes the process of embedding functions to sequence operations on data more explicit, and should lead to a better understanding of functions. The interaction allows the tutor to monitor students more closely and provide more useful assistance as they learn programming plans than one in which students specify only the final surface form of the solution.



Write a program to take a list as an argument and construct a new list with the last element rotated to the front of the list. For example, (a b c d) would become (d a b c).

You need to eventually connect to (a b c). (a b c) contains the first few elements of (a b c d).

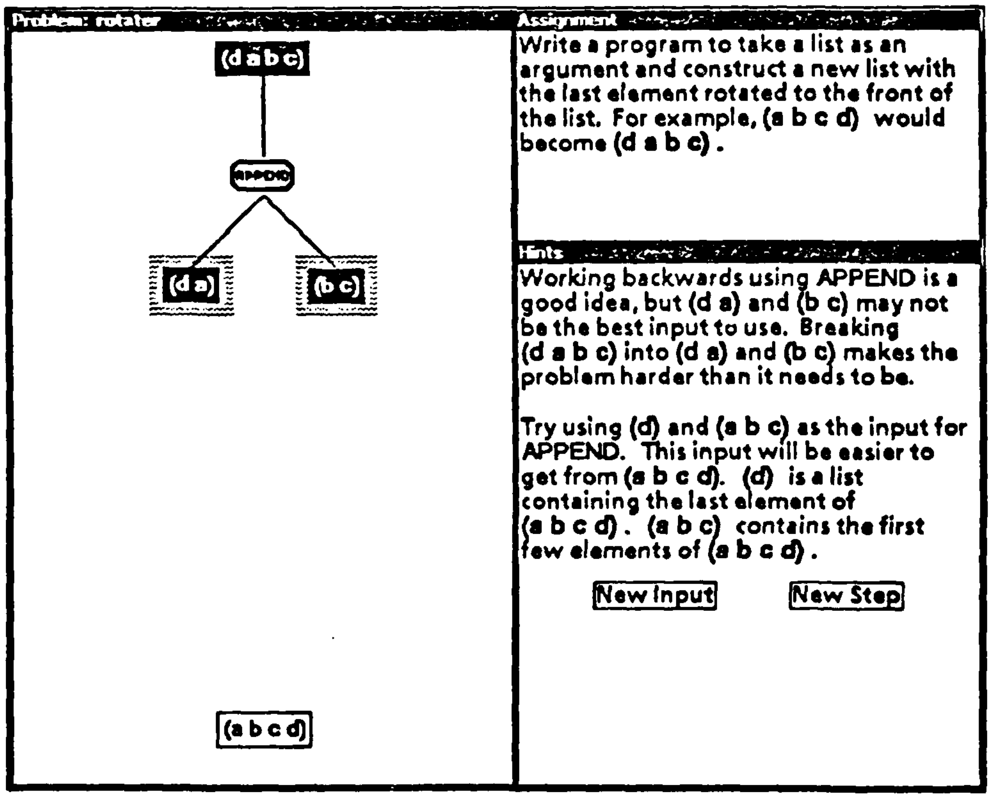
You might start by using a function that gives you (d c b a). You can use (d c b a) to eventually extract (a b c). (d c b a) is a list with the reverse of (a b c) at the end.

Figure 1. A forward working hint in the GIL intelligent tutoring system.

A difficulty in learning to solve problems in many types of domains is that the syntax of a problem's solution does not reflect the reasoning process required to construct the solution. We designed the graphical representation of a program used in GIL to be more congruent with the reasoning required in the task than the traditional text form. The structure of the solution being constructed mirrors the planning required to construct a program. Reasoning chains are represented by branches of the graph joined together to achieve the final goal. A second advantage is that the intermediate reasoning products are made explicit, so that students see the data manipulated as they construct the program. This helps students understand how particular algorithms work and, more generally, helps them learn the logic of embedding functions within other functions to construct an algorithm. Finally, the GIL interface enables students to plan in a variety of directions. The GIL problem solver contains rules for reasoning forward from the given data toward the goal, and rules for doing goal decomposition by reasoning backward from the goal toward the given data. The interface supports both types of reasoning and provides a distinct visual representation that mirrors the direction of reasoning. Students may reason in either direction, and may choose to work on any branch of the program at any time. The system supports reasoning about steps in whatever order is natural for the students, even though it may not match the order in which the components appear in the final surface form of the solution.

GIL helps students solve problems by providing feedback when students request a hint and when students make an illegal or strategically poor step. The GIL explainer draws upon the same knowledge base that the problem solver uses to construct solutions: problem solving rules, plans, and general knowledge about the LISP operators. If a hint is requested, the explainer selects a rule that best continues the problem solving exhibited by the student so far, and uses the information in the rule to construct a hint. Figure 1 displays a hint provided by GIL, suggesting a plan to work forward to achieve the current goal of getting a list containing everything but the last element. In the case of student errors, GIL finds the most similar matching rule and uses it to explain how the student's step is in error. Because the rules contain a description of the input and output, the student's step can be compared with the properties of the objects in the rule, and any discrepancies can be described. GIL handles legal errors, in which the step in the program does not correctly manipulate the selected data, and strategic errors, in which the chosen step is a legal LISP operation but is not strategically useful. GIL first describes what is good about the student's step and then points out the ways in which the step is in error or could be improved by explaining how the student's input, output, or function deviates from the properties in the correct rule.

GIL's reaction to a strategic error is displayed in Figure 2. Two levels of help are shown. The first paragraph points out the nature of the error; the second level, requested by the student, offers a specific suggestion and explains why this step would be effective. In this way, the first level of feedback provides the opportunity for students to figure out how to fix the step, or to request more directive feedback.



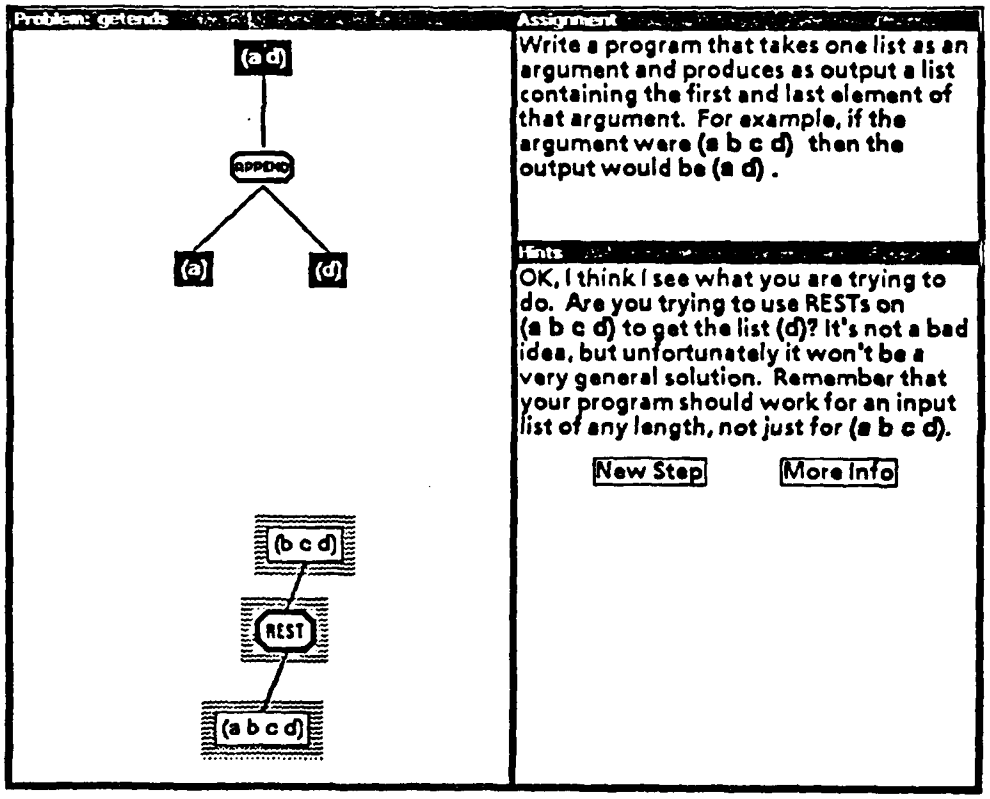
Write a program to take a list as an argument and construct a new list with the last element rotated to the front of the list. For example, (a b c d) would become (d a b c).

Working backwards using APPEND is a good idea, but (d a) and (b c) may not be the best input to use. Breaking (d a b c) into (d a) and (b c) makes the problem harder than it needs to be.

Try using (d) and (a b c) as the input for APPEND. This input will be easier to get from (a b c d). (d) is a list containing the last element of (a b c d). (a b c) contains the first few elements of (a b c d).

Figure 2. Two levels of feedback for a strategic error.

Figure 3 shows another type of strategic error. In this case, the error is made on a forward working step. GIL infers that the student has adopted a plan to extract the component (d) of the target list by using rest repeatedly until a list of the last element is obtained. The conditions for this plan match the constraints of the current problem, except that the student's plan requires knowing the position of the target element from the front of the list, and the problem description states that the target element is the last element. (It is only true for this example that the target is the fourth element; with a different input list, then the last element might be the fifth element or the sixth, etc.)



Write a program that takes one list as an argument and produces as output a list containing the first and last element of that argument. For example, if the argument were (a b c d) then the output would be (a d).

OK, I think I see what you are trying to do. Are you trying to use RESTs on (a b c d) to get the list (d)? It's not a bad idea, but unfortunately it won't be a very general solution. Remember that your program should work for an input list of any length, not just for (a b c d).

Figure 3. A strategic error corresponding to an overly specific plan.

By dynamically constructing explanations directly from its problem solving knowledge, GIL responds to student errors without any bug catalog. Instead, explanations are dynamically constructed by comparing the student's step to the relevant problem solving rules. Our work on GIL demonstrates the potential for driving the explanation process directly from problem solving rules. Such a tutor is more flexible in that it can generate feedback dynamically to new situations rather than being prepared with extensive bug catalogs and a catalog of English explanations associated with all the correct and buggy rules.

4.2 An Empirical Study of GIL

We have begun to evaluate the effectiveness of our GIL tutor. Our first study compared students learning to program with GIL with another group of students learning the same programming concepts in traditional text-based format (Reiser, Ranney, Lovett, & Kimberg, 1989). These control group subjects were also the control group for the human tutoring study mentioned earlier. The subjects read a short text describing the data structures, atoms and lists, how LISP functions work, how to combine LISP functions into a program, and how to use a chain of LISP functions to describe a general algorithm. The text was adapted from the first two chapters of the programming text described earlier. The main modification of the curriculum for the GIL students was the substitution of graphical representations for the text-based representations and the omission of a discussion of the syntax for defining functions and referring to variables.

The GIL subjects solved a sequence of 14 to 15 programming problems interspersed with the text. Subjects worked on the problems using GIL following a brief demonstration of the system by the experimenter. The experimenter demonstrated how to take forward and backward steps, how to cancel steps, and how to request hints. All subjects had little difficulty learning the procedure for specifying steps in GIL. No subject found it necessary to ask the experimenter for assistance in using the interface following the demonstration.

The simplest measure, learning time, suggests that GIL is very effective in teaching programming. Students working with GIL completed reading the text and working on the problems in less than two hours, which is roughly less than half the time that subjects typically spend working on these two chapters without tutoring. There are many reasons to expect subjects to solve problems with GIL more quickly than subjects working without tutoring. Direct comparisons with these subjects are difficult to make, however, because the curriculum is very different. In the standard LISP learning situation, students spend a good deal of time mastering the syntax of defining functions and using variables. To minimize the load of new material, students are given a full chapter of problems to familiarize them with the data structures and simple LISP functions before introducing them to function definition. In contrast, in the GIL curriculum students are immediately introduced to defining functions and learn the semantics of the simple LISP functions while defining their own functions. Comparisons of overall solution time is misleading, therefore, because the GIL subjects solve fewer problems overall (since most of the Chapter 1 problems are unnecessary for the GIL students) and are presented with simplified versions of function definitions and variables, with graphical rather than text-based structure.

A more accurate evaluation is possible by comparing performance on a selected set of problems from the end of the function definition chapter. We compared the solution times for the last five problems in this chapter. The GIL subjects had been working with LISP at this point for approximately 1 to 1.5 hours, and the control subjects had been working with LISP for about 4 or 5 hours. The control subjects, by this point, had already defined 8 functions and had mastered the syntactic form of function definitions and variables. The difficulties of the last five problems consisted in planning an algorithm to manipulate the data to produce the desired result. The GIL subjects solved these problems much more easily than the control subjects. The total solution times for the GIL subjects for these five problems was 15 minutes, while the control subjects took an average of 58 minutes. Interestingly, GIL matched the effectiveness of the human tutors for these problems. Students working with human tutors solved the problems in approximately 25 minutes.

It is important to be cautious in interpreting these results, because the GIL group produces solutions of a different sort than the Human Tutor or No Tutor control subjects, namely program graphs using concrete examples rather than the textual program with variables. However, the two groups do have a common learning criterion, in that they must successfully construct the same difficult algorithms or sequences of LISP functions to solve the problems in the final section. The results demonstrate that students can more easily master the material required to solve these problems in the GIL environment. We believe that the faster learning of the GIL students is due to two aspects of the environment: the model tracing nature of GIL and the nature of interacting with GIL's graphical programming environment.

4.3 Facilitating Reasoning with Model Tracing Feedback

The model tracing nature of GIL provides feedback for students somewhat similar to the type of feedback that human tutors provide. GIL's feedback achieves the same purpose as a human tutor's constant feedback: it keeps the student from going too far off the track and minimizes the consequences of errors. Consider the much larger number of solution attempts required for the No Tutor subjects than the Tutor subjects. The human tutors focused students' attention on which part of the solution needed to be fixed, and when necessary offered hints on how to proceed with the modifications. GIL provides feedback of the same sort. The GIL feedback points out whenever a step is in error. This feedback provides visual and verbal information to help the student locate the error by marking the portion of the student's step that needs to be modified in the graph. In addition, the feedback briefly describes how the step can be improved. The feedback is in the form of a hint, in that it describes the nature of the difference between what the student has and what is needed, but does not provide the correct answer (unless the student requests "More Info"). In addition, GIL's feedback distinguishes between a variety of errors in reasoning about LISP's behavior (legal errors) and poor strategies (strategic errors). The error feedback and hints appear to greatly reduce the students' time to construct a solution.

GIL's feedback differs somewhat from the human tutors' in that it is typically more explicit and directive. Whereas GIL says "your input to append needs to be a list," the human tutors would typically focus the student on the same error by using leading questions, such as: "Think about your input again. What type of input do we need for append?" Nevertheless, the important aspects appear to be that the feedback focuses the student's attention on the error that needs correcting and urges the student to attempt to correct it, rather than merely telling the student what to do. GIL students are typically able to fix their errors relying on the first level of help; students immediately corrected an error in approximately half of the cases, and requested more directive error feedback in only 12% of the errors.

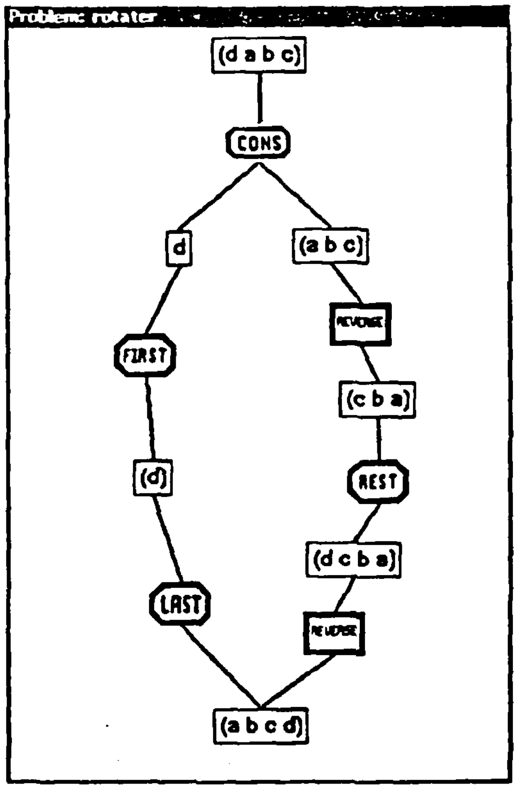
A second point about the feedback is its immediate nature. Our human tutors, like Fox's tutors, provided continual feedback. Students could always tell that they were on the right track, or that a step was not correct. Although in some cases tutors delayed responding to an error until later in the solution, an immediate response was more typical. GIL also responds immediately, as soon as the subjects have finished a complete step. If something is wrong with the step, GIL displays a hint. If the step is correct, then GIL is silent, corresponding roughly to the human tutor's quick verbal agreement. In summary, by following the students' reasoning and responding upon errors to attempt to hint the student toward a fix of the error, GIL emulates the continual feedback and leading questions provided by human tutors.

A second GIL study, conducted by John Connelly, demonstrates the importance of GIL's explanations. This experiment employed two modified feedback conditions, in addition to the standard explanatory GIL condition. In the minimal feedback case, GIL merely indicated whether each step was correct or not, and provided students the opportunity to request the correct step, without an explanation of the error or the aptness of the correct step. In the location feedback condition, GIL indicated which part of the step was incorrect and indicated how to fix it upon request, but again provided no explanations. Subjects in these conditions relied more on the second level of help to be told how to fix the error. In spite of this greater reliance on being told the right answer, these subjects exhibited longer error-fixing episodes and poorer performance on post-tests than those receiving explanations upon errors. Location feedback subjects performed better than minimal feedback subjects, but both groups fared worse on all measures than the explanatory GIL feedback subjects. The study suggests that GIL's facilitation of students' learning is not merely providing feedback on the correctness of the steps or in leading students to a solution by telling them the answer. Rather the positive effects of GIL lie in the way in which the explanations enable students to fix their errors and understand why their fixes are successful.

4.4 Facilitating Reasoning with Reasoning Congruent Representations

Another advantage of the GIL tutor that appears to account for its effectiveness is that the environment is more congruent with the reasoning students need to do to construct programs than the standard LISP environment is. Students can more easily understand how LISP works and more easily plan algorithms using GIL. The most striking evidence for this concerns their use of forward and backward reasoning. Students are free to work forward from any of the given data toward any of the goals at any time in the problem (as in the step taken in Figure 3), or backward from the goal data toward the given data (as in the step taken in Figure 2). We analyzed the number of forward and backward reasoning steps in each student's solutions (excluding final steps that linked a forward branching path with a goal). Interestingly, 95% of all steps were forward steps rather than backward steps. Backward steps were used in only 10% of the problems. When backward steps were used in a problem, only a single backward step was taken. Indeed, all uses of backward steps occurred using LISP combiner operations, which act to decompose the problem into two subgoals. These goal decomposition steps were typically the first step taken by the subject in the problem and often followed a hint from GIL to use such a goal decomposition step. In summary, subjects showed a strong tendency to work forward from the given data toward the goal. In some of the more difficult problems, a backward step might be used to decompose a complex goal into two subgoals, but this strategy was not very frequent. When a backward decomposition was employed, subjects then worked completely in a forward direction to achieve those goals, even though further backward steps of course were possible.

It is interesting to contrast the direction of steps taken by these subjects as they planned their algorithm with the surface form of the final solution in standard text-based LISP. A complete solution for the problem shown in Figure 1, along with the solution in standard LISP form, is shown in Figure 4. Note that the order of the functions in the surface form of the code corresponds to a complete top-down or backward solution, coding cons before `first` and `first` before `last`. If subjects' reliance on forward steps in GIL is a true representation of their reasoning, then novices do not appear to plan their solution in the order in which the functions appear in the solution. Forcing subjects to enter their code in the outside-in left-to-right fashion required by standard LISP interpreters and by the CMU LISP Tutor certainly forces students into a different strategy than the one in which they can work through the algorithm in the order in which it transforms the data. While it is true that a preference for forward reasoning does not guarantee that subjects are able to solve the problems more effectively when forward reasoning is allowed, the greater ease in solving the problems when given an environment that supports forward reasoning strongly suggests that in this case at least the students do know what is good for them.



```
(defun rotater (lis)
  (cons (first (last lis))
    (reverse (rest (reverse lis)))))
```

Figure 4. A completed program graph and the corresponding LISP code.

Our observations of novice programmers working with human tutors are consistent with these findings. Even though the tutors encourage students to try portions of their solution on the computer or to enter portions into the editor, students are reluctant to do this. These students appear to work through the algorithm on paper or verbally with the tutor using forward reasoning to manipulate the data of an example until a subgoal is satisfied (e.g., getting all but the last element of a list in the `rotater` problem shown in Figure 1). At that point, students are willing to enter the solution they have just planned, and can do so in the left-to-right order of the surface form of the solution.

There are several reasons why the forward reasoning available in GIL may be effective for these beginning students. In general, GIL provides an opportunity to reason in temporal order about the data transformations. Given the operators available for reasoning in this domain, domain-general techniques such as means-ends analysis would be more effective when reasoning from the given data to the goal rather than via backward reasoning. Since novices are likely to rely on this strategy, providing an environment that supports these forward steps enables them to interact with the tutor as they make each inference step, rather than being forced to plan an entire chain of reasoning before communicating it to the tutor. This reduces the memory load of keeping track of the properties of the data being manipulated. In addition, if tutorial guidance is required, it can be presented on the precise step at which it is required, rather than being delayed until the student has planned enough of the solution to be comfortable in communicating it to the tutor.

5 Comparing the Pedagogical Abilities of Computer and Human Tutors

Our preliminary analyses of our GIL system indicate that it achieves significant pedagogical improvements over a standard learning environment for programming, and approaches the effectiveness of human tutors. Strong pedagogical benefits have also been demonstrated for other tutoring systems using similar methodologies (Anderson, Boyle, & Yost, 1985; Reiser et al., 1985; Singley, Anderson, Gevins, & Hoffman, 1989). In this section, I evaluate the issues in intelligent tutoring systems design from the perspective of human tutoring.

5.1 Model Tracing and Feedback

The ability of human tutors to monitor and provide frequent feedback for students can be achieved to some extent in intelligent tutoring systems. In domains for which it is feasible to build a problem solving model, a tutor can track students' reasoning and provide guidance when necessary. This feedback may take the form of confirmation that the student is on the right track, hints to set goals when the student is not sure how to proceed next, and feedback to help students locate and repair an error in a solution. The success of model tracing tutors in these rich problem solving domains indicates that it is possible to build tutors to provide intelligent feedback to guide students' learning.

A controversial issue in the area of computer-based instruction concerns the timing of feedback. Anderson, Boyle, and Reiser (1985) argued that a tutor should respond with "immediate feedback." We argued that there are clear advantages for providing immediate feedback upon errors because the learning mechanism for adjusting a faulty rule or forming a new correct rule relies upon the problem situation being active in memory. Empirical evaluations of the effectiveness of feedback in procedural learning indicate that students learn procedures more quickly when provided with this feedback. Students can more easily utilize feedback and explanations when the reasoning that led to the error is still accessible. Furthermore, immediate feedback can prevent long episodes of counter-productive floundering. Consistent with this view, Carroll's experiments with "training wheel" interfaces demonstrate a large advantage in learning from environments that prevent typical error states (Carroll & Carrithers, 1984; Catrambone, & Carroll, 1987). Students learned computer skills more quickly because they committed less time to recovering from error states that the training interface prevented. The extra time spent by control subjects did not appear to be useful; indeed, they scored lower on performance tests than the training wheel subjects.

On the other hand, many educators argue that learning is more effective when students are given the opportunity to experiment and make errors and thereby "discover" and learn from correcting their own errors (e.g., Bruner, 1961; Papert, 1980; Schank & Farrell, 1987; Schwartz, 1989; Schwartz &

Yerushalmy; 1987). One might argue, based on this view, that the immediate feedback provided by model tracing tutors circumvents the possible benefits from exploration.

How do human tutors behave when providing feedback? As already discussed, the evidence from tutoring dialogues collected by Fox and in our laboratory demonstrate that tutors are very quick in the feedback they provide. Fox argues that tutors usually provide students an opportunity to correct their own errors by slightly delaying their feedback when the student makes an error. However, these delays are very short (1-2 seconds) and in the examples presented by Fox feedback is provided typically within the next sentence or two following the error. Furthermore, students become very attuned to the constant confirmatory feedback provided by tutors. In most cases, the lack of immediate reassurance following a step is taken by students to indicate that they have made an error. Thus, human tutors appear to provide at least minimal feedback quickly following each student step.

As discussed earlier, there is some variability in whether tutors respond immediately. The immediacy of the tutor's feedback appears to be tied to the learning potential of the error. In comparing results of different studies, it seems reasonable to propose that the learning domain may influence the behavior of tutors. For example, interruption in programming may be more likely than interruption in subtraction, because the solution times for the problems tend to be longer and hence the time delay between the context in which the error occurred and the detection and fix of the error may interfere with learning from the error. On the other hand, in interactive domains that provide feedback, such as computer programming or computer learning environments, tutors may be more willing to let errors through and let the student discover them. However, this type of error discovery is very much guided by the tutors, and thus differs from pure discovery learning. The tutors, through increasingly directive questions, help students notice that an error has occurred, help them focus on the relevant part of the solution, and may help them determine the correct repair of the error.

An important consideration concerns the content of tutors' feedback to students. Lepper and Chabay (1988) argued that tutors do not perform error *diagnosis*. This is an extremely important issue, because the diagnosis of faulty reasoning has formed the basis of many intelligent tutoring systems approaches (Anderson, Boyle, & Reiser, 1985; Burton, 1982; Clancey, 1986; Johnson, 1986; Ohlsson & Langley, 1988; Sleeman, 1982). Lepper and

Chabay argued that human tutors do not offer students their own hypotheses about the faulty reasoning that led to a particular error. Instead, they direct the student's attention to the part of the work that is in error, in an attempt to get the student to notice and fix the error. Often the tutors will ask leading questions to focus the student on the part of the step that needs to be repaired. In many cases, the tutor's feedback is sufficient for the student to rethink the step and and correct the mistake without further hints from the tutor.

There is an important distinction to be made here. It is clear that the tutors in Lepper and Chabay's examples do not offer diagnosis and descriptions of the misconceptions that would account for a student error. In this regard, they are more subtle than some diagnostic intelligent tutoring systems. Many computer tutors offer a description of the student's faculty reasoning, specifying the particular way in which a solution is incorrect, or provide specific corrective feedback explaining how the solution can be repaired (e.g., Anderson, Boyle, & Reiser, 1985; Burton, 1982; Johnson, 1986). Although the human tutors do not provide such directive feedback, they enable the student to infer this type of information. Through leading questions these tutors do focus students' attention on the parts of a step that need to be fixed. These leading questions may become progressively more directive if the student does not take the initiative and recognize the error (Lepper, 1989). In fact, their leading questions may even refer to the surface characteristics of the error, as in the following leading question from a tutor of a student learning addition: "Now look at that again. Can you put two numbers down in one column here?" (data from Putnam cited by Lepper and Chabay, 1988, p. 246). Thus, although stated as a question, such feedback essentially informs the student of the manner in which the solution is incorrect. In summary, although human tutors are more subtle in communicating their feedback than most intelligent tutoring systems, it is clear from their leading questions that the tutors do perform the extremely detailed reasoning to understand the student's solution and where it goes awry. The difference in strategy appears to be that human tutors direct students' attention through leading questions, and create a situation in which the student can do more of the work to discover the error than would be possible if the tutor would simply describe the fault in the solution.

To summarize these issues, the analyses of human tutors suggest that they exhibit certain aspects of the error diagnosis exhibited by the immediate feedback model tracing tutors. Feedback from the human tutors is quickly used by students to confirm that they are on the right track or to realize they have made an error, and very focused questions from tutors may help students isolate the particular part of their step that is in error. However, tutors will not convey any detailed hypotheses about the particular buggy rule that would lead to the exhibited error, and do not seem to engage in questioning the student in order to further elucidate the nature of their misconception (Putnam, 1987). Furthermore, human tutors are able to provide feedback in a somewhat less intrusive manner than the typical verbal explanations provided by computer tutors, when simply by delaying simple confirmation (such as "ok" or "right") they help the student realize that the step needs to be reconsidered. Human tutors also manage to direct the student's attention to the error in a more subtle manner, and may even demand more work from the student in fixing the error than the more direct and informative computer tutor's messages. On the whole, however, the careful monitoring of students' solutions by model tracing tutors appears to provide one form of the benefits of individualized human instruction.

There are also some limitations in the type of feedback possible within model tracing intelligent tutoring systems. The type of feedback it is possible to provide is limited in many respects by the relatively low bandwith of communication between the student and the computer tutor. The human tutors studied by Fox and those in our laboratory show a high degree of interactivity in their conversations — tutors and students interrupt each other, complete each other's sentences, and communicate a great deal through very short utterances or indeed even through short pauses. The communication can combine a variety of communication media, for example, talking while pointing, talking while writing, pointing at something written, and so on. The communication between students and computerized tutors is much more limited. First, despite achievements in artificial intelligence in the areas of natural language understanding and natural language generation, particularly in constrained domains (e.g., Allen, 1987; McKeown & Swartout, 1987), these capabilities are still greatly limited by comparison with human communicative abilities. Furthermore, the computer is currently confined to presentation of information on the screen. Although interesting techniques are being developed for the input and display of information (e.g., Miller 1988; Shneiderman, 1983) this is clearly a more limited medium than the variety of interaction strategies available to human tutors.

5.2 Student Control

A related issue concerns the control of the interaction by the student. One possible answer to the problem in selecting the right information to provide is to leave the selection of information up to the student. For example, a system might point out to the student that a step is in error and leave to the student selection of information about the location of the error, explanations about why the step will not work, characteristics of a correct step, possible fixes, and so on. Although letting the student control the explanation process fits well with some pedagogical philosophies, it remains an empirical question whether students will take advantage of the help provided and be able to select the information that would be most useful to them. In many situations, students may be reluctant to request assistance from a tutor, preferring instead to try to solve the problem themselves. Experiments on the content of feedback and issues of student control are needed to explore these issues.

Our initial investigations of tutoring suggests some promising potential for this possibility, as well as some caution. Our human tutoring studies show that students ask approximately twice as many questions of a tutor (our Consultant condition) than they ask questions of the experimenter when given instructions to solve problems on their own (our No Tutor condition). Interestingly, the time spent working with the Consultant comprises approximately only one-tenth of these subjects' learning time. However, the decreased learning times for the Consultant condition indicates that subjects learned the material with less difficulty than the No Tutor control group. It is encouraging that the relatively small amount of tutorial assistance, requested under student control, produced a large savings in learning time.

It is in some ways tempting to argue that these results support the view that assistance in computer-based instruction should be completely under the student's control. However, I would urge caution in extrapolating too strongly from these results to computer-based instruction controlled completely by the student. It is important to note that although the interactions were initiated by students, once initiated, the interactions followed a typical tutorial style, with much of the control in the hands of the tutor. The length of the assistance interactions varied widely, from 30 seconds to 15 minutes, presumably due at least in part to the tutor's decisions about when the student could profitably return to the problem alone. Once called in, the tutors provided help beyond answers to specific questions, often answering questions not asked, or leading students to answer the questions themselves, through hints. Thus, these tutors are providing significantly more interactive assistance than the types of hints and error feedback in current intelligent tutoring systems. It does not follow therefore that rendering the same type of feedback currently available in intelligent tutoring systems under the students' control will produce pedagogical benefits. In any case, these results are provocative, and suggest the need for empirical work on the issues involved in student control in intelligent tutoring systems.

Another important aspect of the feedback provided by human tutors is the delicate balance the tutors maintain between guiding the interaction while maintaining the students' motivation. Although the student's performance is certainly greatly assisted by the tutor's guidance, this is more apparent to outside (objective) observers than to the students themselves. Students working with human tutors derive not only cognitive benefits but also motivational benefits, and tend to develop increased feelings of competence about the domain (Lepper et al., in press). The more sophisticated feedback strategies used by human tutors may be more effective in maintaining this balance than current computer tutors. Our GIL tutor and others using the model tracing methodology are much more direct in feedback than human tutors. The control of the interaction by the tutor is much more obvious than in situations with human tutors. For example, students sometimes ask experimenters "why won't it let me do that?" There are situations in which the tutoring system will not allow a particular step that a student thinks is valid, and the relatively restricted interaction possible between student and tutor does not let students explain what they have in mind. Although it is far from true that human tutors let students take whatever steps in a problem they think best, human tutors do manage to appear less controlling. The impact of the more directed nature of computer tutors on the students' motivation has yet to be investigated. It is certainly true that intelligent tutoring systems provide students with more freedom than most computer-assisted instruction, at least of the drill and practice variety, yet it is likely to seem to the students that they are more controlling than a human tutor. Further study is required to investigate the impact of intelligent tutors on motivation, and specifically the importance of learner control. Overall, however, the guidance provided by intelligent tutors seems to have a serious positive impact on motivation, in that it prevents the extremely costly and frustrating episodes in which students working without tutoring flounder and fail to find solutions to problems. A successful tutoring system can bring a difficult domain within the student's competence, and provide a challenging task with less danger of failure. For example, Schofield and Evans-Rhodes (1989) have observed much greater student involvement and motivation among students working with Anderson, Boyle, and Yost's (1985) geometry tutor than is common for students in high school geometry classes.

5.3 Facilitating Reasoning

In some regards, computer tutors may have advantages that are difficult for human tutors to emulate. For example, computers are very effective devices for organizing the visual work space. An important part of some model tracing tutors such as our GIL tutor, Anderson, Boyle, and Yost's (1985) geometry tutor, and other systems such as Algebraland (Foss, 1987), is that the system automatically updates the screen display to represent the current problem situation. These systems also make effective use of graphical representations to represent the structure of the reasoning. Such representations are effective in helping students reason about mathematics and programming problems, even in absence of intelligent feedback. Furthermore, the explicit representation of intermediate products has been extremely effective in our GIL system. This interface seems to facilitate reasoning through algorithms more than the standard representation for solutions. Finally, the representation of the reasoning history helps students reflect upon the solution process, which can be an important component of the learning in these domains (Collins & Brown, 1988). These systems demonstrate how graphical representations, automatic updating, and history keeping characteristics of computerized visual displays can all be used to great advantage. Thus, optimal strategies for computer tutoring may diverge in some respects from those used by human tutors.

5.4 Making Reasoning Explicit

An important benefit of human tutors is that they encourage students to make their reasoning explicit. Palincsar and Brown's (1984) reciprocal teaching method and several programs for teaching "thinking skills" have stressed students' articulation of their problem solving processes as a method for building and debugging problem solving skills (see Bransford, Arbitman-

Smith, Stein, & Vye (1985) for a discussion of such programs). Thus, part of the benefit of interacting with a tutor may be the articulation of one's own reasoning. Students benefit by being paired in a collaborative problem solving situation with the tutor. This is an aspect of tutoring only partially explored by computer tutors. The model tracing systems do force students to be explicit about their reasoning, because they need to enter a solution, step by step, rather than just the final product. Furthermore, some systems teach students intermediate representations for discussing solution plans (e.g., Bonar & Cunningham, 1988). In this way, the tutor can provide a language specially devised for communicating reasoning. Nevertheless, in most tutoring systems the tutor is clearly in control. It would be interesting to explore the effects of providing a true collaborative problem solver, in which the benefits of the tutor would not be due to its explanation and guidance, but instead would be due solely to requiring students to articulate their reasoning. Another interesting possibility now being explored is the use of intelligent exploratory environments as the basis for collaborative problem solving between two students (Behrend, Singer, & Roschelle, 1988).

6 Conclusions

I have examined some of the properties of human tutors and have compared them to what has been achieved in computerized tutors. Model tracing tutors can provide the immediate feedback and guidance provided by human tutors such as hints when students are stuck, feedback to help in locating an error, and guidance in repairing an error. Currently, human tutors are more subtle, less direct, and possibly more gentle in this feedback process, but the cognitive and motivational consequences have yet to be explored. Computer tutors are currently limited by a low bandwith of communication, but many advantages of visual displays are now being explored. Much further research is required on these issues of the timing and content of feedback, student control, and learning by discovery. Intelligent tutors are now being used as excellent experimental tools with which to explore these issues. Intelligent tutors can be used to run more fine-grained studies of feedback and learning than have been previously undertaken in educational research.

References

Allen, J. (1987). *Natural language understanding.* Menlo Park, CA: Benjamin/Cummings.

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

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

Anderson, J. R., Boyle, C. F., & Yost, G. (1985). The geometry tutor. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA.

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

Anderson, J. R., & Reiser, B. J. (April, 1985). The LISP tutor. *Byte*, 10, 159-175.

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, 86, 124-140.

Behrend, S., Singer, J., & Roschelle, J. (1988). A methodology for the analysis of collaborative learning in a physics microworld. *Proceedings of ITS-88: The International Conference on Intelligent Tutoring Systems*, Montreal, pp. 48-53.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13, 4-16.

Bonar, J. G., & Cunningham, R. (1988). Bridge: Tutoring the programming process. In J. Psotka, L. D. Massey, & S. A. Mutter, Eds. *Intelligent tutoring systems: Lessons learned.* Erlbaum.

Bransford, J. D., Stein, B. S., Arbitman-Smith, R., & Vye, N. J. (1985). Improving thinking and learning skills: An analysis of three approaches. In Segal, J. W., Chipman, S. F., & Glaser, R., Eds. *Thinking and learning skills, Volume 1: Relating instruction to research.* Erlbaum.

Bruner, J. S. (1961). The act of discovery. *Harvard Educational Review*, 31, 21-32.

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

Carroll, J. M., & Carrithers, C. (1984). Blocking learner error states in a training wheels system. *Human Factors*, 26, 377-389.

Catrambone, R., & Carroll, J. M. (1987). Learning a word processing system with training wheels and guided exploration. *Proceedings of CHI+GI '87 Human Factors in Computing Systems*. New York: ACM, 169-174.

Clancey, W. J. (1986). Qualitative student models. *Annual Review of Computer Science*, 1, 381-450. Palo Alto, CA: Annual Reviews, Inc.

Cohen, P. A., Kulik, J. A., & Kulik, C.-L. C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal*, 19, 237-248.

Collins, A., & Brown, J. S. (1988). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems*. New York: Springer-Verlag.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*. Hillsdale, NJ: Erlbaum.

Corbett, A. T., Anderson, J. R., & Patterson, E. J. (1988). Problem compilation and tutoring flexibility in the LISP Tutor. *Proceedings of ITS-88: The International Conference on Intelligent Tutoring Systems*, Montreal, pp. 423-429.

Foss, C. L. (1987). Learning from errors in Algebraland. Technical Report IRL87-0003, Institute for Research on Learning, Palo Alto, CA.

Fox, B. A. (1988). Cognitive and interactional aspects of correction in tutoring. Technical Report #88-2. Institute of Cognitive Science, University of Colorado, Boulder, Colorado.

Johnson, W. L. (1986). *Intention-based diagnosis of errors in novice programs*. Palo, Alto, CA: Morgan Kaufman.

Lepper, M. R. (1989). *Goals and strategies of expert human tutors: Cognitive and affective considerations*. Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA, March 1989.

Lepper, M. R., Aspinwall, L., Mumme, D., & Chabay, R. W. (in press). Self-perception and social perception processes in tutoring: Subtle social control strategies of expert tutors. In J. Olson & M. P. Zanna, Eds. *Self inference processes: The sixth Ontario symposium in social psychology*. Hillsdale, NJ: Erlbaum.

Lepper, M. R., & Chabay, R. W. (1988). Socializing the intelligent tutor: Bringing empathy to computer tutors. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems.* New York: Springer-Verlag.

McKeown, K. R., & Swartout, W. R. (1987). Language generation and explanation. *Annual Review of Computer Science*, Vol. 2, 401-449.

Miller, J. R. (1988). The role of human-computer interaction in intelligent tutoring systems. In M. C. Polson, & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Erlbaum.

Ohlsson, S., & Langley, P. (1988). Psychological evaluation of path hypotheses in cognitive diagnosis. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems.* New York: Springer-Verlag.

Palincsar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction*, 1, 117-175.

Putnam, R. T. (1987). Structuring and adjusting content for students: A study of live and simulated tutoring of addition. *American Educational Research Journal*, 24, 13-48.

Reiser, B. J. (in preparation). Pedagogical strategies and learning outcomes of human tutoring. Manuscript in preparation, Princeton University.

Reiser, B. J., Anderson, J. R., & Farrell, R. G. (1985). Dynamic student modeling in an intelligent tutor for LISP programming. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA.

Reiser, B. J., Friedmann, P., Gevins, J., Kimberg, D., Ranney, M., & Romero, A. (1988). A graphical programming language interface for an intelligent LISP tutor. *Proceedings of CHI'88, Conference on Human Factors in Computing Systems*, ACM, New York, pp. 39-44.

Reiser, B. J., Friedmann, P., Kimberg, D., & Ranney, M. (1988). Constructing explanations from problem solving rules to guide the planning of programs. *Proceedings of ITS-88: The International Conference on Intelligent Tutoring Systems*, Montreal, pp. 222-229.

Reiser, B. J., Kimberg, D. Y., Lovett, M. C., & Ranney, M. (in press). Knowledge representation and explanation in GIL, an intelligent tutor for programming. To appear in J. Larkin, R. Chabay, & C. Scheftic (Eds.), *Computer Assisted Instruction and Intelligent Tutoring Systems: Establishing Communication and Collaboration*, Hillsdale, NJ: Erlbaum.

Reiser, B. J., Ranney, M., Lovett, M. C., & Kimberg, D. Y. (1989). Facilitating students' reasoning with causal explanations and visual representations. In D. Bierman, J. Breuker, & J. Sandberg, Eds. *Proceedings of the Fourth International Conference on Artificial Intelligence and Education.* Springfield, VA: IOS.

Schank, R. C., & Farrell, R. (1987). Creativity in education: A standard for computer-based teaching. *Machine-Mediated Learning,* 2, 175-194.

Schofield, J. W., & Evans-Rhodes, D. (1989). Artificial intelligence in the classroom: The impact of a computer-based tutor on teachers and students. In D. Bierman, J. Breuker, & J. Sandberg, Eds., *Proceedings of the Fourth International Conference on Artificial Intelligence and Education.* Springfield, VA: IOS, 238-243.

Schwartz, J. L., & Yerushalmy, M. (1987). The geometric supposer: Using microcomputers to restore invention to the learning of mathematics. In D. N. Perkins, J. Lockhead, & J. Bishop (Eds.), *Thinking: The second international conference.* Hillsdale, NJ: Erlbaum.

Schwartz, J. L. (1989). Intellectual mirrors: A step in the direction of making schools knowledge places. *Harvard Educational Review,* 59, 51-61.

Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *IEEE Computer,* 16, 57-69.

Singley, M. K., Anderson, J. R., Gevins, J. S., & Hoffman, D. (1989). The algebra word problem tutor. In D. Bierman, J. Breuker, & J. Sandberg (Eds.), *Proceedings of the Fourth International Conference on Artificial Intelligence and Education.* Springfield, VA: IOS, 267-275.

Slavin, R. E. (1983). *Cooperative learning.* NY: Longman.

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