ARI Research Note 96-60

Reasoning-Congruent Learning Environments: Scaffolding Learning by Doing in New Domains

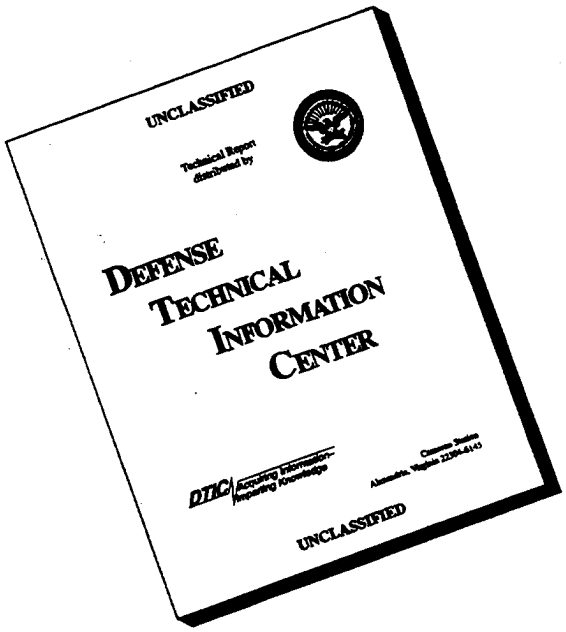
Douglass C. Merrill and Brian Reiser
Northwestern University

Research and Advanced Concepts Office
Michael Drillings, Acting Director

April 1996

United States Army
Research Institute for the Behavioral and Social Sciences

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Director

Research accomplished under contract
for the Department of the Army

Northwestern University

Technical review by

Joseph Psotka

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED (from. . . to)
1996, April	Interim	August 1992-March 1994

4. TITLE AND SUBTITLE	5a. CONTRACT OR GRANT NUMBER
Reasoning-Congruent Learning Environments: Scaffolding Learning by Doing in New Domains	MDA903-92-C-0114
	5b. PROGRAM ELEMENT NUMBER 0601102A
6. AUTHOR(S)	5c. PROJECT NUMBER B74F
Doulass C. Merrill and Brian Reiser (Northwestern University)	5d. TASK NUMBER 2901
	5e. WORK UNIT NUMBER C05

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Institute for Education and Training School of Education and Social Policy and The Institute for the Learning Sciences Northwestern University Evanston, IL 60201	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Institute for the Behavioral and Social Sciences ATTN: PERI-BR 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. MONITOR ACRONYM ARI 11. MONITOR REPORT NUMBER Research Note 96-60

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

COR: Michael Drillings

14. ABSTRACT *(Maximum 200 words)*:

One major focus of research in cognitive science and education has been the mental representation of problem solving knowledge. Novices facing problems in new domains need to reason about the causes and effects of domain operators to be able to learn from problem solving in the new domain. We argue that this causal reasoning allows novices to apply their learning to new situations. We will first highlight some difficulties facing novices in new domains, and propose a theory of learning environment design emphasizes the role of the problem solving environment as a structured note pad to support incremental planning and execution of problem solutions. We will describe three groups of actions that such an environment must lead students to perform and the outcomes of these actions for novices.

15. SUBJECT TERMS

Reasoning Learning environments

SECURITY CLASSIFICATION OF			19. LIMITATION OF ABSTRACT	20. NUMBER OF PAGES	21. RESPONSIBLE PERSON (Name and Telephone Number)
16. REPORT	17. ABSTRACT	18. THIS PAGE			
Unclassified	Unclassified	Unclassified	Unlimited	31	

Reasoning-congruent learning environments: Scaffolding learning by doing in new domains

Douglas C. Merrill
Institute for Education and Training
RAND

Brian J. Reiser
School of Education and Social Policy
and The Institute for the Learning Sciences
Northwestern University

Date: July 12, 1994

Address correspondence to:
Douglas C. Merrill
RAND
1700 Main Street
Santa Monica, CA 90407-2138
E-mail: Doug_Merrill@rand.org
Phone: (310) 393-0411
Fax: (310) 393-4818

RUNNING HEAD: REASONING-CONGRUENT LEARNING ENVIRONMENTS

Abstract

One major focus of research in cognitive science and education has been the mental representation of problem solving knowledge. Novices facing problems in new domains need to reason about the causes and effects of domain operators to be able to learn from problem solving in the new domain. We argue that this causal reasoning allows novices to apply their learning to new situations. We will first highlight some difficulties facing novices in new domains, and propose a theory of learning environment design that produces environments to minimize these difficulties. This theory of learning environment design emphasizes the role of the problem solving environment as a structured note pad to support incremental planning and execution of problem solutions. We will describe three groups of actions that such an environment must lead students to perform and the outcomes of these actions for novices.

Introduction

A major goal of the learning sciences, psychology, and cognitive science is to develop an understanding of how people attain and use new knowledge in order to support more effective learning. Much of this work focuses upon the way knowledge is represented in the mind. For example, some research has focused upon the ways people acquire conceptual understanding, and what the outcomes of this understanding might be (Carey, 1985; Chi, Feltovich, & Glaser, 1981; Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Johnson-Laird, 1983; Schank, 1986), and while others have examined the ways people learn to apply knowledge to problem solving situations (e.g., Anderson, 1983, 1990; Chase & Simon, 1974; Laird, Newell, & Rosenbloom, 1987; Larkin, 1981; Lewis, 1988; Newell, 1990; VanLehn, 1990).

This work provides a foundation of the knowledge novices need to acquire while becoming expert in a domain. In this paper, we build upon this foundation to develop a theory of learning environment design that scaffolds novices' learning in new domains by helping them render invisible behavior visible, making expert-like planning processes accessible to the students, and leading the novices to reason about situations in the world rather than about problem solving notations. Before presenting the theory, we discuss the nature of what novices must represent about a domain to achieve success.

Novices often focus upon learning facts, definitions, and examples independent of conceptual organization (diSessa, 1988; NAEP, 1988). In fact, a cottage industry has arisen to teach these facts, typified by the so-called *cultural literacy* movement (Hirsch, 1987). In contrast, many researchers argue that education should focus not upon mastery of facts in isolation, but rather on novices' attempts to integrate the new information into preexisting knowledge (diSessa, 1988; Pea, 1992).

One common method for encouraging students to be active learners is to use a problem solving approach. In such an approach, students might read a chapter and then solve a number of problems relating to that material. Curricula in a variety of different fields have taken this tack, such as mathematics (Bransford, Sherwood, Vye, & Rieser, 1986; NCTM, 1989), social sciences (Banks & Clegg, 1990), and problem-based medical education (Patel, Groen, & Norman, 1993; Williams, 1992). Problem solving is a very effective way for students to learn to apply the declarative knowledge given in the book to situations requiring it (Anderson, 1983).

However, problem solving curricula must avoid the pitfall of simply replacing a transmission model with one that values simply solving many arbitrary similar exercises, such as is the case with many traditional textbooks. This repetitive structure often leads to little more than the development of rote procedures for solving a certain type of problem, rather than engendering a rich understanding of the domain. For example, children are often given large groups of the same type of word problem to solve, one after the other. Students in this situation often choose solution methods based solely upon the occurrence of various words in the problem. For example, students would judge a problem that contains a five, a three, and the word "gives" to be a subtraction problem (Fuson & Willis, 1989). Worse yet, students after a number of subtraction problems in a row may stop even looking for the words, but instead may just find the numbers and apply the over-learned subtraction procedure (Fraivillig, 1993).

Similar rote strategies are found in many other domains. In fact, these rote strategies are so common that instructors sometimes teach them to the students (Heller & Reif, 1984). For example, the first author was taught in a literature class that the character

first mentioned in a book was the main character. Presumably this was meant as a rule of thumb, but it was encoded as a simple procedure: "To identify the main character, find the first character mentioned." This heuristic performed well for the literature read in that particular class, but fails miserably for classics such as *Moby Dick*, where Ishmael — the first character mentioned — is relatively unimportant, and *The Scarlet Letter*, in which the character first introduced is never again mentioned in the book.

A major goal of instruction is to get novices to employ newly acquired knowledge in other situations and contexts (cf., diSessa, 1988; Resnick, 1987), and also to be able to apply their knowledge to the real world, which is often very difficult (Bransford et al., 1986; McCloskey, Caramazza, & Green, 1980). As illustrated in the literature example. even a procedure that works for a few selected examples may be insufficiently general to allow the knowledge to be transferred. To execute a solution procedure, the procedure must be selected from those available in memory and must then be applied. However, the triggers that allow rote procedures to be selected may be too fragile, depending upon features of the particular set of problems used rather than upon the important structural elements of the problems (Fuson & Willis, 1988). Thus, novices in many domains need to master more than simple procedures that can be employed to attain a solution.

If rote procedures for solving problems in some domain are fragile and may not allow for transfer to problems in new but similar situations, then novices must learn more than just the procedures, they must develop an understanding of the domain. *Conceptual understanding* (Lewis, 1988; Ohlsson & Rees, 1991; Wertheimer, 1959) often refers to the students' understanding of an underlying structure of a domain. Exploiting the underlying structure of the domain, may, in turn, require reasoning about a sequence of events over

time and the ways they interact. We call such a sequence the *process in the world* (Collins, 1990; Merrill, Reiser, Beekelaar, & Hamid, 1992).

This process in the world contains information about the causal sequence leading from an initial event to a final outcome. To be able to select an appropriate operator, novices must understand what the operator does and when it can be applied. This information in encoded in the causal sequence. Thus, understanding the process in the world involves reasoning about when operators can be applied and what they do. This knowledge in turn facilitates transfer of the learned procedures to other contexts, including real world ones.

Many important problems do not exhibit single step causality, where a change directly causes the outcome of interest. Rather, in many complex domains, several intervening causes and effects may create a pathway from the initial change to the outcome of interest. Novices can use their understanding of the process in the world to generate predictions about the outcomes of a change in a system, such as the effects of tax rate increase on unemployment. To generate the prediction, a novice can use the process to simulate the incremental effects of the input, thereby generating a sequence of outcomes in turn. Students presented with this tax increase problem could simulate the tax rate change in their process, and then watch its effects over time as they go through decreased spending money to lower business revenues, and on to unemployment. The final prediction depends on the intermediate ones, which one generates with a causal model of the domain's behavior. Such a mental simulation of a causal process if often called a *mental model* (Norman, 1983).

Understanding the process in the world serves to do more than provide predictions of causal sequences. Mental models of a domain often include descriptions of when various operators can apply (e.g., Kieras & Bovair, 1984). Furthermore, causal models serve to help

people generalize operations to new but similar contexts (Lewis, 1988). Thus, attaining an understanding of the process in the world would allow novices to extend learned procedures to new areas. Since a major goal of instruction is to enable people to apply their problem solving procedures to real world situations, an emphasis on problem solving without an associated emphasis on the applicability of the operators in complex situations may not lead to sufficiently richly represented knowledge. This suggests that a curriculum that focuses not on rote procedures but rather upon understanding the process in the world will result in more generalized applicability of the knowledge than would have been achieved in more traditional curricula.

How might novices attain an understanding of the process in the world? Although solving problems is a critical component of developing expertise (Anderson, 1983, 1990; Newell, 1990), simply solving many problems may not be sufficient for the novice to develop an understanding of the process in the world. Not only does such problem solving potentially still fall prey to the development of rote procedures, there are barriers to developing an understanding of the process in the world inherent in traditional problem solving.

When solving problems, novices employ a notational system, such as predicate calculus for logic, arithmetic operators for algebra, and a certain class of diagrams for economics, and have great difficulty mapping their world knowledge into the notational requirements (Fuson & Briars, 1990). These notational systems can be very expressive, and are very powerful when understood. However, novices often have great difficulty understanding the notational systems, and this difficulty renders understanding the process in the world very difficult. As will be described in the next section, notations often fail to represent all the interactions in the domain, do not support students' expert-like reasoning, and may even

lead students to engage in poor problem solving strategies. In the remainder of this paper, we consider the difficulties facing novices and offer a theory of learning environment design that specifies how an environment can overcome these difficulties through scaffolding and subsequently fading support for profitable reasoning that connects work with a problem solving notation to reasoning about the process in the world.

Obstacles in learning new domains

We have argued that reasoning about processes in the world is an important component of learning effective problem solving procedures. Yet, novices often experience significant difficulties learning new domains. In this section, we will present some reasons why these difficulties occur, and suggest how learning environments can overcome them.

A major difficulty for learning arises when relevant information is invisible or inaccessible. For example, the process in the world often remains implicit during much of problem solving. This inaccessibility interferes with acquiring an understanding of the different causal forces that act in concert to produce a final result. For example, gravity affects the flight path of a thrown ball, but it is very difficult to tease out the impact of gravity from that of the velocity of the ball upon its release. Many real world problems have the characteristic that some of the different causes of an outcome are confounded, not easily separable. This inseparability interferes with students' acquisition of the process in the world, because various causal forces that should be separate, such as gravity and the force from the person throwing a ball, appear to be the same to the student.

One might wonder if this problem can be overcome if students could try different actions and observe the various outcomes. However, it is often impossible to vary the interacting factors independently that produce an observable physical outcome. Further, it is difficult

for students to build a model of the domain when they are unable to disentangle the various causes acting on an object. Thus, real-world experimentation alone can not overcome the problem of invisible information. Other techniques must be employed.

One major potential advantage of computer technology is to render invisible information visible and to allow control of factors that are normally fixed, such as gravity (Collins, 1990). This potential has led many researchers to investigate *simulated microworlds* (e.g., Lewis, Bishay, & McArthur, 1993; Nathan, Kintsch, & Young, 1992; White & Frederiksen, 1990). Novices in such microworlds can interact with some process as it occurs over time, changing parameters of the world to see each change's effects independently. Thus, instead of simply throwing a ball around a room physically, a student might be able to set the value of factors such as gravity and the initial velocity of the ball, and see the outcomes on a computer screen, perhaps through changes in the spacing dots marking the objects motion that indicates the object's speed (White, 1993) or through vectors representing the same information (Roschelle, 1992). Or when trying to understand how to repair airplane avionics computer equipment, a novice could use such an environment to hook devices together to investigate how the new configuration affects performance and to watch the effects propagate throughout the system (Lesgold, Lajoie, Bunzo, & Eggan, 1992). Students could also use a simulation to help identify intermediate factors that affect the object's behavior. For example, a simulation of the economic world described in the tax increase example earlier might help novices discover the path of intervening causes and effects, such as the effect on personal income, between the tax increase and resulting unemployment increase.

A computational environment can allow novices to perform controlled experiments and

to access invisible intervening states. Thus, simulations, unlike real-world processes, offer students the potential to manipulate the relevant factors involved in some event and see the effects of the change.

Despite the clear promise of simulations, they are not necessarily a successful way to facilitate students' mastery of the process in the world by themselves. Recognizing an effect in a simulation may require reasoning about volumes of data to select that which is relevant. However, novices often have difficulty recognizing which features are relevant to a problem (cf., Chi et al., 1981). Without this recognition, students can not select relevant data, and therefore may become overwhelmed, making learning very difficult. In addition, recall that students need to reason about the causal connections between states. Although simulations do present the states themselves, discovering the underlying causality is often very difficult, and simulations typically provide no assistance for this analysis. Such causal inferences may require the identification of patterns of data leading to some outcome (Pennington & Hastie, 1986). Students often have great difficulty recognizing such patterns, even in simple domains. This is due in part to students' poor hypotheses (Klahr, Dunbar, & Fay, 1990; Wason, 1983) and to poor experimentation strategies to explore these guesses (Schauble, Glaser, Raghavan, & Reiner, 1991).

Learning is not solely determined by the simulation's design. A critical determinant in whether a student learns from a simulation is the student's behavior. People are not uniform in the manners in which they approach learning. Successful students often engage in cognitive behaviors that are qualitatively different than and more successful than those of other students, such as elaborating upon material presented in examples (Chi et al., 1989). Novices who attempt to reason about the physical situation embodied by an algebra

problem make fewer errors than others who attempt to solve the problems without reference to it (Nathan et al., 1992). Furthermore, teaching novices more profitable ways to classify problems into different possible types reduces students' application of less correct keyword driven solution methods (Fuson & Willis, 1989).

We refer to the strategies students undertake when working on problems as *solution processes* (Merrill et al., 1992). These solution processes can offer the leverage for students to acquire an understanding of the process in the world. These strategies include profitable solution strategies used spontaneously by successful novices, such as the elaborative reasoning (Chi et al., 1989), or found to be useful to novices when taught to them, such as a method for classifying arithmetic problems (Fuson & Willis, 1989). Notice that these strategies are not necessarily processes that experts would perform when solving the same problems, who may be using more compiled knowledge (Anderson, 1983). Rather, these are processes that tend to result in the development of expertise with greater gains. The important shared feature of these different techniques is that all of them lead novices to elaborate simple solution procedures into more complex structures including information about when a procedure can be applied, what its constraints are, and what it does. These elaborated structures form are the basic elements of an understanding of the process in the world, which in turn offers the flexibility to solve other types of problems and to generate predictions of effects.

Students also face significant difficulties in the acquisition of solution processes. There are multiple causes of these difficulties. One primary cause is an unfortunate focus in education upon the products of problem solving rather than upon the process by which a solution is attained (Heller & Reif, 1984).

Some instruction does in fact try to lead students to develop such a process-oriented understanding, often by presenting a variety of worked-out examples. Learning how to plan a solution by studying worked-out examples is a serious inductive challenge because reasoning that led to the use of some domain operator is usually not specified, leaving the student with the challenge of figuring out, for example, why the author of the book chose to apply the Side-Angle-Side theorem in an example instead of some other one. Successful students overcome this challenge by trying to predict and explain each step in the example to highlight gaps in their understanding of a problem (Chi et al., 1989; VanLehn, Jones, & Chi, 1992). This sort of reasoning facilitates students' understanding of when an operator can or can not be applied, and why one particular operator was used rather than another, which helps them know when to apply an operator in a different situation.

In addition to this step-by-step reasoning, global problem solving strategies are also a crucial to developing domain expertise. For example, one useful global strategy is breaking a problem down into subproblems each of which is then attacked. This sort of information is also not captured in instruction (Heller & Reif, 1984), but is an important component of student knowledge, serving to guide students' problem solving.

Further difficulties for the acquisition of solution processes arise when there are significant discrepancies between the way solutions must be recorded and the way in which solutions are most easily constructed. For example, Anderson, Boyle, and Reiser (1985) argued that the two column proof format used for geometry proofs misleads students into thinking that proofs should be constructed as linear forward chains, ignoring the likelihood of dead ends and the importance of backward reasoning. Anderson, Boyle, and Yost (1986) designed their Geometry Tutor to overcome this difficulty by using a graphical representa-

tion, called a *proof tree* that helps students realize that geometry problem solving involves search, forward and backward reasoning, and, potentially, dead ends.

In summary, the goal of instruction is for students to be able to apply their knowledge to real situations. Yet, instruction often leads students to focus on procedures sufficient only for textbook style exercises. We have considered the difficulties students experience when learning new domains, focusing on the importance of rendering invisible behavior visible, making expert reasoning processes accessible to the novices, and allowing students to record their inferences as they are actually performed instead of in an arbitrary manner. Overcoming these three difficulties facilitates the acquisition of solution processes which in turn helps students attain mastery of the process in the world. An understanding of the process in the world provides the additional information beyond simple procedures required in many domains, such as the conditions of applicability of operators and the ways operators intervene between a starting action and its final outcome. This additional knowledge serves to support transfer of the skills to problems beyond that which could be achieved through rote procedures.

In the next sections, we will present design principles for creating learning environments that help students attain an understanding of the process in the world by scaffolding key solution processes. As shall be seen, this scaffolding takes the form of changing the tasks students must perform while solving problems.

Design principles for reasoning-congruent learning environments

In the previous section, we described the difficulties novices encounter when learning new domains, focusing on the barriers to achieving an understanding of the process in

the world and solution processes for the domain. In this section, we describe how learning environments can circumvent the difficulties. A reasoning-congruent learning environmentis a learning environment that is congruent with the way successful students reason about the domain and that leads students who might not use these more profitable solution strategies to do so. This is done by scaffolding the development of appropriate solution processes and by leading students to engage in cognitive activities that support the acquisition of the process in the world. We shall focus particular attention upon two interactive learning environments developed in our laboratory.

Insert Figure 1 about here

GIL (*Graphical Instruction in LISP*) is a reasoning-congruent learning environment that supports programming novices as they learn to construct and debug LISP programs (Merrill et al., 1992; Reiser, Kimberg, Lovett, & Ranney, 1992; Reiser, Ranney, Lovett, & Kimberg, 1989). Before starting the in-depth discussion of each principle in turn, we present an example of problem solving in GIL that will serve to demonstrate many of the reasoning-congruent learning environment design principles.

Figure 1 displays an early problem in the GIL curriculum. In this part of the curriculum, GIL students create programs by applying various LISP functions to concrete data examples. Figure 1 shows a partial solution to a difficult problem from this curriculum, called *rotater*. The value at the bottom of the screen, the list *(i am sam)*, is the input provided for the student's function, and the expected output is the list *(sam i am)*. Students select functions from a menu and connect them to data on the screen, which could be the original input, the output, or some data in the middle of the program that has not yet been connected to the input or output. For example, this student applied the function *LAST* to

the input *(i am sam)*. Figure 1 displays some of the important features that will be discussed in detail later in this section. Briefly, the explicit data nodes attached to functions, such as the *(sam)* attached to the *LAST*, are provided by students while they build their solutions. These nodes serve to encode normally hidden information in the solution itself. Furthermore, the structure of the solution is more congruent with the way students plan solutions — pending goals are explicitly noted, reasoning chains are represented as separate branches in the graph, and so on. These and other advantages will be discussed later in this section.

We will present a set of design principles for *reasoning-congruent learning environments* that specify how a learning environment can support this type of problem solving. Reasoning-congruent learning environments that support effective problem solving strategies can be designed according to the principles that are summarized in Table 1. These principles specify activities and information students need to learn new mathematics and science domains effectively. These seven principles cluster into three closely related groups, which address each of the difficulties presented in the previous section. The remainder of this section is organized around these interrelated principles. First we deal with two principles designed to foster the acquisition of the process in the world through rendering behavior visible. Then we turn to three principles that focus on students' ability to plan solutions incrementally and record the associated inferences in the environment, thereby using the environment as a note pad that helps students structure their reasoning. Finally, we turn to techniques for requiring students to engage in more effective problem solving strategies. For each principle, we will describe the pedagogical motivation, the ways the pedagogical goal can be achieved in a learning environment, the effectiveness of the approach, and a

summary of the claimed learning consequences.

Insert Table 1 about here

Principle 1: Make students' reasoning explicit

Reasoning-congruent learning environments encourage students to make their reasoning explicit. Recall that developing an understanding of the process in the world requires recognizing the relationships between states — developing a model of why one state follows another. Chi et al. (1989) found that students who predicted the next state and then explained any incorrect predictions developed a better understanding of the domain than those who did not. These more successful students' representations of the domain rendered all intervening states explicit and contained detailed explanations of the progression of states, both of which help students locate and repair errors. These types of causal connections are also essential for analogical use of an example for later problem solving (Faries & Reiser, 1988; Gentner, 1983). Furthermore, this articulation facilitates students' reflection on their own reasoning, including the reasoning that previously would have been tacit (Collins, 1991). Reflection on one's own problem solving makes it possible for students to identify areas of the solution that may be correct but not optimal (Collins & Brown, 1988), and thus to be able to generate superior solutions to similar later problems. The inclusion of tacit reasoning in the final solution product facilitates students' understanding of the behavior of the solution, thus enabling learning from the solution procedure (Anderson, 1983, 1990; Anderson & Corbett, 1993).

Recall the example problem shown in Figure 1. The student had used the function *LAST* in the solution. GIL then required the student to state what *LAST* would return

in this case; the student correctly answered that the list *(sam)* would be output from *LAST*. By requiring students to provide these intermediate results, GIL leads students to make explicit their beliefs about the expected behavior of each function, rather than simply assembling a sequence of functions to try to achieve some goal. This facilitates students' learning of the preconditions and outcomes of each individual operator. Indeed, these intermediate products have strong pedagogical benefits. Trafton and Reiser (1993) demonstrated that students asked to articulate these intermediate products created more accurate initial solution attempts and to exhibit superior debugging of these programs than other students who were not asked to do so. Thus, leading students to make explicit predictions of intermediate results leads to superior performance.

Principle 2: Render invisible behavior of a complex operation visible

Mastering the process in the world that generates an answer given some input often requires reasoning about the relationships between invisible states intervening between start and end. Natural solutions often have many such invisible states. Figure 2 contrasts a GIL solution to a traditional text solution which embodies a great deal of invisible behavior. Intermediate products in GIL serve not only to lead students to make their reasoning explicit, but also to render normally invisible behavior both visible and accessible.

Insert Figure 2 about here

In contrast, in a traditional LISP environment, there would be no simple way to examine the intermediate states of the definition of the function *rotater*. LISP does provide a tool called *trace* to let programmers see the data that was passed to a function and what that function returned. However, typically students use the tool sporadically and it is difficult to

follow an entire path through from start to finish because understanding the trace output is often difficult.

Insert Figure 3 about here

It is important to consider students' mastery of subskills when desiging this type of facility. For example, later in the curriculum, students use variables to solve these problems, which involve conditional expressions and logical functions. When writing programs with variables rather than concrete data examples, students to not need to predict the output of each function at each point. Instead, GIL provides a testing facility, shown in Figure 3, that allows students to see what their solution would do with some input value that they themselves provide. During this test, GIL flashes the functions being evaluated and darkens links between functions to provide a concrete trace of the the path the program takes as it is executed. GIL also provides the most critical intermediate values — the output of all tests and actions evaluated — on the title bar of the test or action box itself, as shown in Figure 3.

These major outputs are not sufficient to help students explore the behavior of their solutions. Students may also need to be able to inspect the outcomes of intermediate operators when analyzing the source of faults in a system (du Boulay, 1986). A reasoning-congruent learning environment can provide virtual probles to allow access to internal states. For example, in GIL, students can click on any function or link that was evaluated to see the value carried on the link or output by the function, such as is shown in the middle test box of Figure 3. In VSE (*Visual Structured Editor*), students can click on any function to see the data it produced, as shown in Figure 4.

Insert Figure 4 about here

In Sherlock, students have an electronic tool much like an electrician's multimeter that allows them to measure currents, voltages, and resistances across any element (Lesgold et al., 1992). Similarly, in ThinkerTools (White, 1993), motion of objects at any given instant is shown by a *data cross* that displays the object's velocity in the horizontal and vertical dimension. The movement of the object over time is displayed in a *wake*, a series of dots that shows the position of the object at fixed time intervals. Thus, students have access to the instaneous velocity of an object and a persistent measure of the object's behavior. All of these allow students to probe values that otherwise might be invisible, such as the actual path taken by an object over time, or difficult to inspect, such as the resistence across a circuit encased inside a large computer.

Since probes do render invisible behavior visible, we would predict that students who use a probe to gain access to intermediate states, to verify their predictions of behavior, and to develop explanations of errors will develop a better understanding of when operators can and should be applied. This knowledge can later serve to guide similar problem solving.

Indeed, probes do scaffold student problem solving, particularly solution debugging. This point can be illustrated with an example from data from a recent study (Merrill & Reiser, 1993a). Figure 3 shows an erroneous partial solution to a difficult conditional problem. In testing this solution, the student noticed that the graph was not returning the correct value, *end*, and used the meter to examine the results of the computations in the middle test box. Figure 3 shows the student examining the output of *LAST* in the middle *Test* box of the conditional, which is the erroneous component. The student apparently

recognized the error at this point, because the student's next step in the solution was to remove the last element from its list, which resulted in a correct solution. Thus, allowing students to inspect invisible states can guide their error recovery and planning.

Summary of rendering invisible behavior visible principles

Rendering invisible behavior visible should improve students' ability to learn from their solution (Anderson, 1983, 1990) and to use their solutions analogically in later problems (Faries & Reiser, 1988; Gentner, 1983). It is very difficult for students to acquire an understanding of the process in the world when much of the behavior is invisible, because students can not disentangle the causal contributions of various factors. We suggested two different ways of rendering invisible behavior visible: predictions and virtual probes.

Students who are led to predict the behaviors of operators while creating a solution perform better on learning tasks (Trafton & Reiser, 1993). We argue that this increased performance is due to students' focus on the preconditions and actions of each component of the solution path while creating and debugging a solution. This focus leads students to learn more than simple procedures for solving problems. Rather than just stringing together sequences of partially understood operators to generate solutions, students develop an elaborated understanding of when operators are applicable and how they behave. A virtual probe allows students to inspect the different causal factors. These inspections serve to help students acquire an understanding of the process in the world.

The next group of principles covers the ways students plan and construct solutions to problems, and the ways a notation can help their abilities to do so.

Principle 3: Minimize translation between solution plan and notation

When students solve problems, they must keep a great deal of information available at all times, such as their active subgoal, pending subgoals, and those that have been completed. This creates the potential need to mentally rehearse a great deal of information. Often the working memory overload that can result leads them to forget crucial goals or drop other information about what operators do or when they can be applied (Anderson, 1987). This overload is a serious obstacle to learning, since students do not have sufficient spare cognitive resources to maintain problem solving information needed to learn from a situation (Sweller, 1988).

One source of this load is that traditional representations for solving problems require that students translate their internal solution plan into an external representation that does not match the students' way of thinking about the problem. This translation often leads students into severe difficulties, such as situations in which students know what action to perform but can not express it correctly in the external representation, or situations in which the external representation itself leads students to use poor strategies.

Reasoning-congruent learning environments attempt to minimize this translation load by providing a notation better suited as a means of expressing their solution plans. The notation of a reasoning-congruent learning environment can make it simpler to express solutions, minimizing the need to construct a solution and mentally rehearse it while reasoning about how to express the operations in the notation. For example, the graphical notation in GIL is designed to better correspond to the structure of a solution. Two reasoning chains are represented by two branches in the graph (see Figure 1). The test-action windows in a GIL conditional solution makes the organization of the program elements into those higher

order structures easily observable, as seen in Figure 3.

Similarly, in the ANIMATE system of Nathan et al. (1992) designed to teach algebra word problem solving, a graphical notation for algebraic expressions makes the structure of the relationships in the problem more transparent. Although experts can easily discern the structure of an expression in standard text-based notations like programming languages or algebra, by such means as indentation or parenthesis matching, perceiving this structure is an acquired skill that is difficult for novices. A reasoning-congruent learning environment circumvents that difficulty by providing structures that are more congruent with students' planning. Another issue in translation concerns the order in which students construct components of their solutions.

Trafton and Reiser (1991) argued that constructing solutions working from a program's initial input toward the output best matches students' planning as they reason chronologically about each function's output being input to the next function. So, for example, in Figure 1 students will, in general, start by applying functions to the *(i am sam)*. However, LISP is a functional language, so text solutions are written in the opposite order – the leftmost function, the *CONS* in Figure 2, is typed first but is in fact the last operation to be performed on the data. Although it is in principle possible to construct solutions in any order using a text editor, novices are strongly biased to write out solutions consistent with the form in which they appear in the final solution. If students indeed can plan solutions more easily reasoning from the initial input, they may be led to develop an entire solution plan internally, and then reverse it to express it in the text notation. This introduces serious danger of translation errors and places a load on working memory to rehearse this plan while communicating it.

GIL allows students to work in any direction they desire at any time to enable them to express solutions in the order in which they were planned (Merrill et al., 1992; Ranney & Reiser, 1989; Reiser et al., 1992). Allowing students to reason in any direction they desire has significant pedagogical outcomes. In studies contrasting novices required to construct their programs in a forward directions (from givens to goals) or in a backward direction (from the goals to the givens), or in either order desired, Trafton and Reiser (1991) found that students forced to reason backwards required more initial planning time before beginning a solution, made more errors, and spent more time debugging these errors than students allowed to reason forwards. Furthermore, students allowed to choose their own direction of problem solving chose to work almost entirely forward.

Trafton and Reiser (1991) argued that forward reasoning maps best onto students' naive model of functions, in which they reason about th chronological sequence of input feeding into a function and producing a result. Translation between their model and its expression in the notation, as occurs when novices are forced to reason forward from the goal, creates additional difficulty. Second, the search is more constrained in LISP when reasoning forward, because a range of input data and functions can produce a desired output, whereas a particular function applied to certain input data yields a unique result. Third, students experience significant difficulty when they are forced to commit to a given solution method before they are ready (Green, Bellamy, & Parker, 1987). Backward reasoning in LISP, selecting functions to combine portions of a list, requires them to commit to a solution method before they know the data that will be produced from each subgoal, thus causing novices to commit to a solution plan that may be more complex than one selected if the decision could be made later.

Principle 4: Structure reminds students of solution progress

Classical views of problem solving posit sequential stages of understanding a problem, constructing a plan, implementing the plan, and evaluating the result (Polya, 1983). For example, schema based models of expertise propose that experts categorize a problem, select an appropriate solution schemata, and instantiate the plan (Chi et al., 1981; Larkin, 1985). When problem solving becomes complex, however, problem solving is better described as an iterative process of planning a portion of the solution, implementing the portion, and evaluating the progress so far (Bauer & Reiser, 1990). Problem solving impasses may cause students to reconsider their current solution plans or the additional load imposed by the reasoning about the local impasse may drive the rehearsed plan out of working memory. This could cause the student to forget what has been done so far in the solution and interfere with needed reasoning about why the error occurred (Chi et al., 1989; Schank, 1986). When students lose their understanding of what has been accomplished in the solution, and what remains to be done, they must reconstruct their solution plan.

Maintaining a close consistency between students' mental solution plans and their expression in the solution notation is essential for successful problem solving. Errors can be difficult to detect when they arise due to inconsistencies between different parts of the solution, each of which appears correct in its local context (cf., Merrill, Reiser, Merrill, & Landes, 1993). Indeed, errors often arise due to difficulties in merging subplans, each of which successfully accomplishes its own subgoal (Soloway, Spohrer, & Littman, 1988)

One way to recreate a solution plan is to rebuild it from the portions of the solution already completed. This may not be as easy as it seems, however. If the notation in which the plan is expressed is significantly different from the way novices would spontaneously

generate it. they would have to translate their previous work from the external notation into their own understanding. This translation, like that required in solution creation, causes additional working memory load and can lead to errors.

If a problem solving environment could facilitate the process of monitoring a plan, serving as a concrete representation of each subplan in a solution, students could more easily assemble their plans and reconstruct them when needed. Collins and Brown (1988) have called such use of an external environment to scaffold planning by making plans explicit in a notation *reification*, and have argued it to be a critical type of scaffolding that a learning environment can provide (Brown, 1985). An environment could reify the students' problem solving by presenting the previous problem solving steps in a structure that preserves the students' reasoning about the relationship of each solution component to the others and to the plan as a whole. Students could reconstruct their solution plans in such an environment by reading the previous steps off in order, since the structure of what the students thought about a problem is present in the solution to serve as an external memory. External memory has been shown to aid problem solving, even in tasks with relatively small goal structures and few possible states (Kotovsky, Hayes, & Simon, 1985).

Reasoning-congruent learning environments reify plan structures this because the external representation of a solution matches the way novices spontaneously produce the solution plan. Thus, a reasoning-congruent learning environment solution can remind students of their position more easily than a solution which requires significant translation from external to internal representation. The reasoning-congruent learning environment solution can help students assemble plans more easily, help them monitor their progress, and enable them to reconstruct and revise their goal structures, more easily allowing problem solving

to continue after impasses with little translation.

GIL solutions, in contrast to text LISP, convey program structure through the arrangement of boxes representing functions and lines connecting them. The spatial arrangement of the icons conveys the relationships between the operators in a more discernible manner than relying on indentation and embedding of text. Connections between icons convey the explicit path of data. Students may use spatial organization to group related parts of the program before all the connections have been fleshed out and can leave space for pending parts of the program. In the early part of the GIL curriculum, intermediate products can be used to mark pending goals, as in Figure 1, and to eliminate the need to rederive the current data since the student indicated the data values at any given state during program construction.

In a pilot study comparing students solving problems in traditional LISP to students solving the same problems in GIL, we found that, not surprisingly, the text students spent more time and required more changes to make the syntactic elements of the solutions correct. More importantly, the text LISP students required more changes to the structure of their solutions to achieve a correct solution than the GIL students did (Reiser, Beekelaar, Tyle, & Merrill, 1991). Thus, not only did GIL scaffold difficulty with syntactic elements, its representation did a better job of conveying the structure of what was there to students, which focused their structural debugging. In data recently gathered in our lab (Merrill & Reiser, 1993a), we found that GIL students required less time and fewer edits to achieve correct solutions than students using a structured editor, a text-based editor that prevented certain syntactic errors.

Principle 5: Allow students to focus upon subproblems in any order

Another issue relating to the match between the structure of a solution and students' plans is the ability to break larger problems into meaningful subproblems and then attack each in turn. Problem decomposition strategies are an important part of the knowledge specific to each domain that novices must master for effective problem solving (e.g., Chi et al., 1981; Larkin, McDermott, Simon, & Simon, 1980; Newell & Simon, 1972). A reasoning-congruent learning environment can support problem decomposition by enabling students to set explicit subgoals for themselves to achieve, rather than requiring students to develop solution plans to achieve the entire problem at once. For example, the BRIDGE tutor (Bonar & Cunningham, 1988) allows students to articulate the major subgoals of the problem in English and then provides a graphical notation for each type of plan.

In GIL, the intermediate products allow students to specify major subgoals. In Figure 1 the student realized that the list *(i am)* will be required input for the *CONS* function and put down an intermediate product to note this inference. The next few steps in the student's solution will be to try to achieve this new subgoal. The process of explicitly noting goals to be achieved is sometimes called *goal posting* (Singley, 1990). Singley (1990) has argued that making the current goal explicit via goal posting leads students to focus their problem solving more effectively and makes it easier for them to recognize when they have achieved a goal.

The notation used in a reasoning-congruent learning environment can support decomposition and posting goals to achieve. The notation itself may change depending upon the domain, and may even change over time. In GIL, for example, particular LISP objects themselves serve a posted goals. For example, in the early curriculum, students are given

an explicit value which they try to achieve and can also post intermediate goals by putting other data values on the screen. In the later GIL curriculum, students can put functions to be used later on the screen to remind themselves of where they are going, and can put out as many conditional cases as they think they will need. Thus, GIL uses aspects of its problem solving notation to support goal posting. The ability to construct and run each major subcomponent such as separate cases of a conditional or items in a looping construct supports decomposing problems into subgoals and focusing upon each subgoal independently.

The calculus tutor developed by Singley (1990) has students type English descriptions of calculus goals into a problem tree to indicate what they are working on and what they plan to do next. Thus, the calculus tutor adds another notation on top of the problem solving notation to support goal posting. Singley (1990) argued that these goals led students to acquire a better understanding of the goal structures for calculus problems, which aided their later problem solving.

Summary of minimize translation principles

These three design principles all serve to aid problem solving by decreasing the cognitive load associated with translation from the manner students would spontaneously create a plan to the manner it must be expressed. We propose three closely related pedagogical principles of reasoning-congruent learning environments to do this. First, reasoning-congruent learning environments should allow novices to express their solution in the manner they normally would. Translation between the solution plans that novices would express spontaneously and the ways that a notation requires them to do so introduces significant cognitive load for the students. Students must plan more of a solution before expressing it, must

rehearse more of their partial solution as they reason about other parts of the problems, and may be forced to commit prematurely to particular solution plans. Reasoning-congruent learning environments provide notations so that this translation is minimized, with problem solving activities structured around notations based upon the ways that students do in fact prefer to express their solution plans, thus enabling students to plan their solutions more easily as they express them. Second, reasoning-congruent learning environments should help students iteratively plan portions of the solutions, implement them, and evaluate the portions. Furthermore, reasoning-congruent learning environments scaffold students' recreation of solution plans, to allow the students to focus on the operators and their relations, rather than being forced to expend energy trying to understand why they did the previous work in the solution. Finally, reasoning-congruent learning environments should provide a means for students to note future goals to achieve. Leading students to post goals further facilitates students' incremental solution planning by allowing students to plan a portion of the solution, write it out, and then rely upon the environment to indicate the goals they have achieved and those that remain to be attained.

The next group of principles covers important cognitive activities that enable students to acquire understanding of the process in the world.

Principle 6: Lead students to predict the behavior of objects and construct explanations when the predictions are incorrect

These final principles of reasoning-congruent learning environments concern activities for students — solution processes — to facilitate their acquisition of the process in the world. rather than behaviors of the learning environment. Solution processes were defined earlier as a set of tools for profitable problem solving, such as generating predictions, explaining

faulty outcomes, or testing all relevant cases of the behavior of the behavior of some object.

One major goal of reasoning-congruent learning environments is to lead students to engage in more effective problem solving strategies, and, thereby, to lead to a mastery of the process in the world. Prior to this section we focused on providing students opportunities to perform useful cognitive activities, such as to use a virtual meter to render hidden behavior visible. In contrast, this section will describe ways of not just providing opportunities, but requiring students to engage in superior problem solving practices.

An important component of problem solving is to make predictions, explaining what led to erroneous predictions (cf., Chi et al., 1989; VanLehn et al., 1992). GIL's intermediate products require students to make function by function predictions. Later, in the variable portion of the curriculum, GIL requires students to predict the outcome of each test of a solution.

These predictions are behaviors students might not otherwise perform, since novices are strongly biased to focus their reasoning on producing what the notation requires as the final form of a solution, and text LISP does not provide a mechanism for even seeing internal states, much less predicting their value. Students, like most rational beings, often try to do no more work than they are forced to do, so if an environment does not lead students to make predictions, they might not do so, which would lead to worse learning performance. Thus, requiring students to make predictions adds to their problem solving load in some ways, but may reduce it in others, for example by rendering visible invisible behavior, and results in superior overall solutions with fewer planning errors that take less time to construct and debug (Trafton & Reiser, 1993). Similarly, the goal posting supported in Bridge (Bonar & Cunningham, 1988) and the Calculus Tutor (Singley, 1990) encourage

students to decompose problems into appropriate subgoals.

Émile is a learning environment for creating documents using text, video, pictures, and sound to explain and model some physical behavior, such as what happens to a parachuted object after it leaves the plane (Guzdial, 1993). Guzdial (1993) argued that there is an optimal path to follow when trying to understand such phenomena sufficiently well to explain them to others, so Émile requires students follow this set path. The path roughly corresponds to forming hypotheses about the behavior to be explained, developing a model of the behavior, comparing the model's behavior to that of the real world, and then debugging it as needed. Left to themselves, students often do not progress through simulation tasks in as orderly a progression as might be desired (cf., Klahr et al., 1990). Émile's requirement that students follow a set order is another example of improving students' learning through leading them to engage in behaviors they otherwise might not.

Principle 7: Lead students to tie the notation to the real world meanings

Another component of solution processes is connecting the notational concerns to the situation embodied by a problem. The real world can constrain the available options for students in problem solving situations and can make additional information available to guide students' solution creation.

ANIMATE (Nathan et al., 1992) provides a graphical simulation of the situations embodied in algebraic problem solving. Students who are not adept at understanding the constraints on algebraic manipulations, such as any number added to the left side must be added to the right as well, are able to understand the constraints of everyday life, where the equivalent constraint would be that one cannot create objects afresh that are not specified

in the problem while solving it.

The original notation in algebra does not serve to provide the appropriate constraints on student problem solving. Furthermore, notations may even lead students to engage in poor strategies. Fuson and Willis (1989) argued that students learning to solve word problems tend to learn a fragile strategy for classifying problems based solely upon the surface features of the problem. Students rapidly notice that some words like "takes from" indicate a certain type of problem, such as subtraction. Further, students will often assume that subsequent problems are the same type as previous ones, and will simply extract the numbers without even looking for the keywords (Fraivillig, 1993). However, classifying a problem without regard to the conceptual relationship between objects in the problem leads students into difficulty when even subtle changes are made in the problems. Thus, the strategy of categorizing problems based solely upon the use of keywords rather than upon an understanding of the situation represented by the problem is very fragile (cf., Chi et al., 1981). Fuson and Willis (1989) taught children to classify problems using a set of diagrams that capture the various conceptual relationships possible, such as *change-get-more* or *compare* that refer to the event in the world references by the problem, such as a child receiving additional items (a change-get-more), rather than by using the presence or absence or keywords in the problem. This new strategy led to better problem success on the most difficult problems, and created a useful language for teachers and students to use when discussing problems. Thus, these diagrams proactively helped students to internalize appropriate solution processes for arithmetical word problems.

In a similar vein, Merrill and Reiser (1993b) argued that students solving microeconomics problems using supply and demand graphs are able to solve problems without

referring to the situation at all, but these solutions are often inconsistent with their real world knowledge. In situations where these students did fall back on reasoning about the situation, they experienced more success overcoming impasses encountered in the problem.

Summary of proactive guidance principles

The goal of reasoning-congruent learning environments in general is to get students to attain an understanding of the process in the world through attaining solution processes. Some profitable solution processes are not behaviors that students would know to do without explicit instruction. Students that do not engage in these practices will have more difficulties acquiring an understanding of the process in the world that generates the behavior due to the inductive difficulties associated with invisible behavior and solution translation. Reasoning-congruent learning environments scaffold the acquisition of the process in the world in a variety of manners, including by leading students to engage in cognitively useful behaviors that they otherwise might not.

How can scaffolding be faded?

We have considered how a learning environments can provide scaffolding for student problem solving. Learning environments must also face the thorny issue of allowing students more control over their problem solving. This process is often called *fading* (Collins, Brown, & Newman, 1989). Students must eventually become practitioners of the learned skills. At some point, they must be able to solve problems that do not involve the learning environment. Thus, there is a pragmatic issue of helping students achieve self-sufficiency in the domain so that they can perform the problem solving on their own, without the support of the learning environment.

Student dependence on the environment can be overcome through fading the scaffolding of appropriate aspects of the task as the student gains expertise. However, the means for choosing the aspects to fade is not at all clear. Remove too much support and the student will become lost in the domain. Fade too little and the student may become dependent on the system or may become bored. To decide what scaffolding to remove and how quickly to do so requires a psychological theory of the skill acquisition process in the domain. This aspect of instruction is an area of future investigation, current research has not sufficiently addressed it.

Systems, such as Émile that requires students solve problems in a particular manner, could relax the constraints on the process of problem solving. This is one form of fading that has been investigated. However, the important issue is to know when fading can be performed and also what to fade.

The theory underlying reasoning-congruent learning environments provides a hint of what to fade. We have proposed two classes of knowledge to be mastered: the process in the world and solution processes. The two processes offer one way of deciding what to fade. Much of the scaffolding described so far has explicitly attempted to help students acquire solution processes so that they can understand the process in the world. In fact, a student who has mastered appropriate solution processes may even need less scaffolding for the process in the world, since the solution processes will contain a set of skills for the student to use to provide the intermediate states that the system previously was revealing. Thus, if a student has mastered solution processes, then not only could an environment reduce support for them, but also could fade some of the scaffolding of the process in the world.

We have attempted to accomplish fading in GIL in several ways, based upon an as-

sumption that students have mastered appropriate solution processes. We expected that students in the later curriculum have mastered the skills of basic list manipulation and the solution processes involved in making explicit predictions of intermediate results, so the scaffolding of this set of solution processes is faded. To accomplish this fading, students work with variables instead of explicit data examples and are not required to make predictions at each step. The students themselves must maintain an understanding of the state of the data and the operation of each individual function. Students can get access to these intermediate states through testing, as shown in the middle test box of Figure 3, but the system does not require the students to do so.

However, it is crucial that the system continue to support students' prediction in some manner. But what sort of predictions should students make? Since students in the later curriculum are focused more upon algorithms than upon the simple list manipulation functions, the predictions should now be at the algorithmic level, unlike previous intermediate products. Thus, GIL asks students to provide the final output of any test they perform of their program, and then highlights any difference between the actual outcome and the predicted outcome. Figure 3 shows an incorrect prediction in the conditionals portion of the curriculum. This facility leads students to continue making predictions but fades the scaffolding of prediction at each function to an algorithmic level, requiring a great deal more responsibility from them.

This approach to fading, letting students choose when to perform cognitive actions, could lead to unfortunate pedagogical outcomes. Students may not know that they should continue to do actions that are no longer scaffolded. If the activities are still important, then such a response will not lead students to learn the process in the world as completely.

Future research goals for interactive learning environments must begin to explore how to select which scaffolding to fade and how to reinstate the scaffolding when required.

Summary of principles

These principles, taken together, define a theory of learning environments that guide students' problem solving, helping the students acquire an understanding of the process in the world by scaffolding the use of appropriate solution processes. Reasoning-congruent learning environments attempt to achieve all these principles to help students construct a understanding of more than just procedures for solving problems in a domain, by enabling students to propagate causes and effects through an entire solution and explain each event as it occurs. Making one's own reasoning explicit leads to an elaborated model of the process in the world about which students are learning. Furthermore, rendering normally hidden behavior visible helps students focus on the behavior of each subcomponent of a solution.

Furthermore, we argued that reasoning-congruent learning environment solutions match the structure of how students express plans spontaneously. This match gives rise to several learning advantages. First, it minimizes difficulties in translation from solution plan to external representation. This translation process is difficult and error prone, leading to increased working memory load that interferes with learning. Second, the external representation can remind students of their current position in their solution plans, since reasoning-congruent learning environment solutions are designed to be understandable with little effort. This reminding helps students remember where they are in their solution path, which is a very important component of problem solving and learning. Finally, breaking a problem up into subcomponents via goal posting and testing meaningful subcomponents

makes it easier for students to recognize when goals have been achieved and to focus their problem solving search.

Reasoning-congruent learning environments focus very heavily on getting students to make use of appropriate solution processes for the domain, often by creating a new framework within which problems are solved. They may do so by requiring students to follow a certain sequence of steps before problem solving commences, as in Fuson and Willis (1989), or by forcing students to engage in a certain sort of process during the problem solving itself (Guzdial, 1993), or even by removing an artificial barrier that leads students to select a less efficient problem solving strategy (Anderson et al., 1986; Merrill et al., 1992). These seven interrelated principles together define the information and activities that reasoning-congruent learning environments scaffold and describe why each should have pedagogical consequences.

Conclusion

Novices can solve problems using rote procedures and still be at least moderately successful in school. However, if we wish students to be able to apply their knowledge to new situations, they must master more than simply procedures. Novices must acquire an understanding of the causal interconnections in the domain — the process in the world that generates the behavior of interest. Mastering the process in the world is facilitated by students' employing profitable means for solving problems — solution processes. The design principles for reasoning-congruent learning environments describe how to construct learning environments based upon the ways students learn.

We have discussed how students face significant hurdles when learning new domains, and have attempted to show how reasoning-congruent learning environments overcome these

hurdles. These difficulties included the prevalence of invisible behavior, required translation from internal plan structure to external plan structure, and the lack of access to expert reasoning. We described a set of principles that addressed each of these in turn. These seven principles together define the theory of reasoning-congruent learning environments. We reviewed results from several domains including programming, mathematics and science. These results show that reasoning-congruent learning environments are in fact superior to traditional problem solving environments on a variety of cognitive and motivational measures. These benefits derive from scaffolding students' acquisition of the process in the world that determines the behavior in the domain. This scaffolding takes the form of supporting students' use of solution processes that lead to additional problem solving success and by capitalizing on what the students already know about real world behavior.

References

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

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192–210.

Anderson, J. R. (1990). *The adaptive character of thought.* Lawrence Erlbaum Associates, Hillsdale, NJ.

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. (1986). The geometry tutor. *Journal of Mathematical Behavior, 5*, 5–19.

Anderson, J. R., & Corbett, A. T. (1993). Tutoring of cognitive skill. In Anderson, J. R. (Ed.), *Rules of the mind*, pp. 235–255. Erlbaum, Hillsdale, NJ.

Banks, J. A., & Clegg, A. A. (1990). *Teaching strategies for the social studies: Inquiry, Valuing, and decision-making.* Longman, New York.

Bauer, M. I., & Reiser, B. J. (1990). Incremental envisioning: The flexible use of multiple representations in complex problem solving. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pp. 317–324 Cambridge, MA.

Bonar, J. G., & Cunningham, R. (1988). Bridge: Tutoring the programming process. In Psotka, J., Massey, L. D., & Mutter, S. A. (Eds.), *Intelligent tutoring systems: Lessons learned*, pp. 409–434. Erlbaum, Hillsdale, NJ.

Bransford, J. D., Sherwood, R., Vye, N., & Rieser, J. (1986). Teaching thinking and problem solving. *American Psychologist, 41*, 1078–1089.

Brown, J. S. (1985). Process versus product: A perspective on tools for communal and informal electronic learning. *Journal of Educational Computing Research, 1*, 179–202.

Carey, S. (1985). Are children fundamentally different thinkers and learners than adults?. In Chipman, S. F., Segal, J. W., & Glaser, R. (Eds.), *Thinking and learning skills*, Vol. 2. Erlbaum, Hillsdale, NJ.

Chase, W. G., & Simon, H. A. (1974). Perception in chess. *Cognitive Psychology, 4*, 55–81.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13*, 145–182.

Chi, M. T. H., Feltovich, P., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5*, 121–152.

Collins, A. (1990). Cognitive apprenticeship and instructional technology. In Jones, B. F., & Idol, L. (Eds.), *Dimensions of thinking and cognitive instruction*, pp. 121–138. Erlbaum, Hillsdale, NJ.

Collins, A. (1991). The role of computer technology in restructuring schools. *Phi Delta Kappan, 73*(1), 28–36.

Collins, A., & Brown, J. S. (1988). The computer as a tool for learning through reflection. In Mandl, H., & Lesgold, A. (Eds.), *Learning issues for intelligent tutoring systems*, pp. 1–18. Springer-Verlag, New York.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In Resnick, L. B. (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, pp. 453–494. Erlbaum, Hillsdale, NJ.

diSessa, A. (1988). Knowledge in pieces. In Forman, G., & Pufall, P. B. (Eds.), *Constructivism in the computer age*. Lawrence Erlbaum Associates, Hillsdale, NJ.

du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research, 2*, 57–73.

Faries, J. M., & Reiser, B. J. (1988). Access and use of previous solutions in a problem solving situation. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pp. 433–439 Montreal.

Fraivillig, J. (1993). Personal communication..

Fuson, K. C., & Briars, D. J. (1990). Using a base-ten blocks learning/teaching approach for first- and second-grade place-value and multidigit addition and subtraction. *Journal for Research in Mathematics Education, 21*, 180–206.

Fuson, K. C., & Willis, G. B. (1988). Teaching representational schemes for solving addition and subtraction word problems. *Journal of Educational Psychology, 80*, 192–201.

Fuson, K. C., & Willis, G. B. (1989). Second graders' use of schematic drawings in solving addition and subtraction word problems. *Journal of Educational Psychology, 81*, 514–520.

Gentner, D. (1983). Structure mapping: A theoretical framework for analogy. *Cognitive Science, 7*, 155–170.

Green, T. R. G., Bellamy, R. K. E., & Parker, J. M. (1987). Parsing and Gnisrap: A model of device use. In Olson, G. M., Sheppard, S., & Soloway, E. (Eds.), *Empirical studies of programmers: Second workshop*, pp. 132–146. Ablex, Norwood, NJ.

Guzdial, M. (1993). Technological support for science learners programming in multiple media. Paper presented at the American Educational Research Association annual meeting in Atlanta, April, 1993.

Heller, J. I., & Reif, F. (1984). Prescribing effective human problem-solving processes: Problem description in physics. *Cognition and Instruction, 1*, 177–216.

Hirsch, E. D. (1987). *Cultural literacy: What every American needs to know.* Houghton Mifflin, Boston.

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness.* Harvard Press, Cambridge, MA.

Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science, 8*, 255–273.

Klahr, D., Dunbar, K., & Fay, A. L. (1990). Designing good experiments to test bad hypotheses. In Shrager, J., & Langley, P. (Eds.), *Computational models of scientific discovery and theory formation*, pp. 355–402. Morgan Kaufmann Publishers, Inc., Palo Alto, CA.

Kotovsky, K., Hayes, J. R., & Simon, H. A. (1985). Why are some problems hard?: Evidence from the tower of hanoi. *Cognitive Psychology, 17*, 248–294.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence, 33*, 1–64.

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In Anderson, J. R. (Ed.), *Cognitive skills and their acquisition.* Erlbaum, Hillsdale, NJ.

Larkin, J. H. (1985). Understanding, problem representation, and skill in physics. In Chipman, S. F., Segal, J. W., & Glaser, R. (Eds.), *Thinking and learning skills, Volume 2: Research and open questions.* Erlbaum, Hillsdale, NJ.

Larkin, J. H., McDermott, J., Simon, D., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science, 208*, 1335–1342.

Lesgold, A., Lajoie, S., Bunzo, M., & Eggan, G. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In Larkin, J. H., & Chabay, R. W. (Eds.), *Computer assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches*, pp. 201 – 238. Erlbaum, Hillsdale, NJ.

Lewis, C. (1988). Why and how to learn why: Analysis-based generalization of procedures. *Cognitive Science, 12*, 211–256.

Lewis, M., Bishay, M., & McArthur, D. (1993). The macrostructure and microstructure of inquiry activities: Evidence from students using a microworld for mathematical discovery. In Brna, P., Ohlsson, S., & Pain, H. (Eds.), *Proceedings of AI-ED '93: World*

Conference on artificial intelligence in education, pp. 169–176 Edinburgh, Scotland. Association for the Advancement of Computing in Education.

McCloskey, M., Caramazza, A., & Green, B. (1980). Curvilinear motion in the absence of external forces: Naive beliefs about the motion of objects. *Science, 210*, 1139–1141.

Merrill, D. C., & Reiser, B. J. (1993a). Pedagogical outcomes of reasoning-congruent learning environments. Manuscript in preparation, Northwestern University.

Merrill, D. C., & Reiser, B. J. (1993b). Reasoning-congruent learning environments in real world domains: Scaffolding learning by doing in everyday problem solving. To be presented at the 1994 Annual meeting of the American Educational Research Association, New Orleans.

Merrill, D. C., Reiser, B. J., Beekelaar, R., & Hamid, A. (1992). Making processes visible: Scaffolding learning with reasoning-congruent representations. In Frasson, C., Gauthier, G., & McCalla, G. I. (Eds.), *Intelligent Tutoring Systems: Second International Conference, ITS '92*, pp. 103–110 New York. Springer-Verlag.

Merrill, D. C., Reiser, B. J., Merrill, S. K., & Landes, S. (1993). Tutoring: Guided learning by doing. Manuscript submitted for publication.

NAEP (1988). *The mathematics report card: Are we measuring up?* Educational Testing Service, Princeton, NJ.

Nathan, M. J., Kintsch, W., & Young, E. (1992). A theory of algebra-word-problem comprehension and its implications for the design of learning environments. *Cognition and Instruction, 9*, 329–389.

NCTM (1989). *Curriculum and Evaluation Standards for School Mathematics.* National Council of Teachers of Mathematics, Reston, VA.

Newell, A. (1990). *Unified theories of cognition.* Harvard University Press, Cambridge, MA.

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

Norman, D. A. (1983). Some observations on mental models. In Gentner, D., & Stevens, A. L. (Eds.), *Mental Models*, pp. 7–14. Lawrence Erlbaum Associates, Hillsdale, NJ.

Ohlsson, S., & Rees, E. (1991). The function of conceptual understanding in the learning of arithmetic procedures. *Cognition and Instruction, 8*, 103–179.

Patel, V. L., Groen, G. J., & Norman, G. R. (1993). Reasoning and instruction in medical curricula. *Cognition and Instruction, 10*, 335–378.

Pea, R. D. (1992). Practices of distributed intelligence and designs for education. In Salomon, G. (Ed.), *Distributed cognition.* Cambridge University Press, New York.

Pennington, N., & Hastie, R. (1986). Evidence evaluation in complex decision making. *Journal of Personality and Social Psychology, 51*, 242–248.

Polya, G. (1983). *How to solve it.* Princeton University Press, Princeton, NJ.

Ranney, M., & Reiser, B. J. (1989). Reasoning and explanation in an intelligent tutoring system for programming. In Salvendy, G., & Smith, M. J. (Eds.), *Designing and using human-computer interfaces and knowledge based systems: Proceedings of the Third International Conference on Human-Computer Interaction*, pp. 88–95. Elsevier Science Publishers, New York.

Reiser, B. J., Beekelaar, R., Tyle, A., & Merrill, D. C. (1991). GIL: Scaffolding learning to program with reasoning-congruent representations. In *The International Conference of the Learning Sciences: Proceedings of the 1991 conference*, pp. 382–388 Evanston, IL. Association for the Advancement of Computing in Education.

Reiser, B. J., Kimberg, D. Y., Lovett, M. C., & Ranney, M. (1992). Knowledge representation and explanation in GIL, an intelligent tutor for programming. In Larkin, J. H., & Chabay, R. W. (Eds.), *Computer-assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches*, pp. 111–149. Erlbaum, Hillsdale, NJ.

Reiser, B. J., Ranney, M., Lovett, M. C., & Kimberg, D. Y. (1989). Facilitating students' reasoning with causal explanations and visual representations. In Bierman, D., Breuker, J., & Sandberg, J. (Eds.), *Proceedings of the Fourth International Conference on Artificial Intelligence and Education*, pp. 228–235 Springfield, VA. IOS.

Resnick, L. (1987). Learning in school and out. *Educational Researcher, 16*(December), 13–20.

Roschelle, J. (1992). Learning by collaboration: Convergent conceptual change. *The Journal of the Learning Sciences, 2*(3), 235–276.

Schank, R. C. (1986). *Explanation patterns: Understanding mechanically and creatively.* Erlbaum, Hillsdale, NJ.

Schauble, L., Glaser, R., Raghavan, K., & Reiner, M. (1991). Causal models and experimentation strategies in scientific reasoning. *The Journal of the Learning Sciences, 1*, 201–238.

Singley, M. K. (1990). The reification of goal structures in a calculus tutor: Effects on problem solving performance. *Interactive Learning Environments, 1*, 102–123.

Soloway, E., Spohrer, J., & Littman, D. (1988). E unum pluribus: Generating alternative designs. In Mayer, R. E. (Ed.), *Teaching and learning computer programming.* Erlbaum, Hillsdale, NJ.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*, 257–285.

Trafton, J. G., & Reiser, B. J. (1991). Providing natural representations to facilitate novices' understanding in a new domain: Forward and backward reasoning in programming. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pp. 923–927 Chicago, IL.

Trafton, J. G., & Reiser, B. J. (1993). Novices' use of forward and backward reasoning in a new problem solving domain. Manuscript in preparation, Northwestern University.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions.* MIT Press, Cambridge, MA.

VanLehn, K., Jones, R., & Chi, M. T. H. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences, 2*, 1–59.

Wason, P. C. (1983). Realism and rationality in the selection task. In St. B. T Evans, J. (Ed.), *Thinking and reasoning*, pp. 44–75. Routledge and Kegan Paul, London.

Wertheimer, M. (1959). *Productive thinking.* Harper and Row, New York. Originally published in 1945.

White, B. Y. (1993). Thinkertools: Causal models, conceptual change, and science education. *Cognition and Instruction, 10*, 1–100.

White, B. Y., & Frederiksen, J. R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence, 42*, 99–157.

Williams, S. M. (1992). Putting case-based instruction into context: examples from legal and medical education. *The Journal of the Learning Sciences, 2*, 367–427.

Author Notes:

We gratefully acknowledge Shannon K. Merrill, Michael Ranney, and other members of the GIL tutoring group for helpful conversations and suggestions. The systems described in this paper, GIL and VSE, were constructed with the assistance of many other people, both at Princeton and Northwestern Universities, including Ron Beekelaar, Chrys Wurmser, Adnan Hamid, Eliot Handelman, Jonathon Meeks, Kevin Neel, Greg Trafton, and Alka Tyle. The work reported here was funded by contracts MDA903-92-C-0114 to Northwestern University and MDA903-87-K-0652 and MDA903-90-C-0123 to Princeton University from the Army Research Institute and a Spencer Foundation Dissertation Year Fellowship to the first author. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, expressed or implied, of these institutions. Address correspondence or requests for reprints to Douglas C. Merrill, RAND, 1700 Main Street, Santa Monica, CA, 90407-2138.

Table 1: The design principles embodied in reasoning-congruent learning environments

Render invisible behavior visible.

1. Make students' own reasoning explicit by having them make predictions of behavior.
2. Render behavior visible by allowing student to access normally invisible states.

Support incremental planning and use of environment as a representation of intermediate inferences.

3. Minimize the translation process from the students' internal plans to the external representation of the solution.
4. Have the structure of a partial solution remind the students of where they are in their solution plan and the search space of the domain.
5. Allow students to focus on subproblems on the way to solving the entire problem, thereby avoiding premature commitment and exploiting independence of subgoals.

Lead students to engage in more effective strategies.

6. Lead students to predict the behavior of objects and construct explanations when the predictions are incorrect.
7. Encourage students to tie inferences in the solution notation to the behavior of the referent objects in the domain.

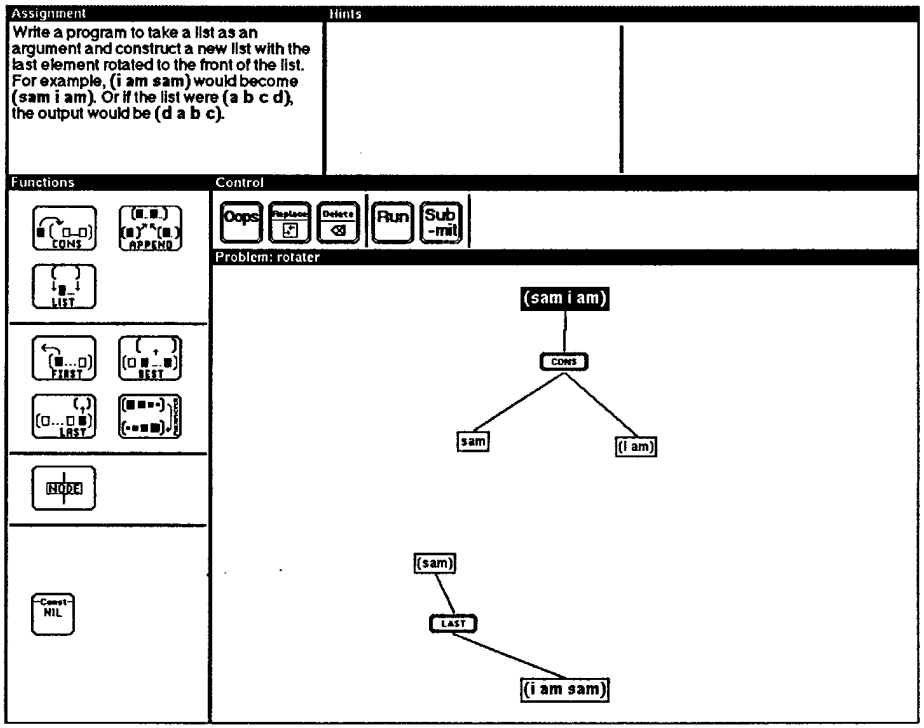
Figure Captions

Figure 1. A partially completed solution to a list manipulation problem in GIL.

Figure 2. A comparison of (a) a completed solution in GIL, (b) the corresponding text form, and (c) a test of the text form. Notice that in a standard programming environment, students observe the behavior of a solution after they have constructed it and tested it with sample data, and then see only the final output.

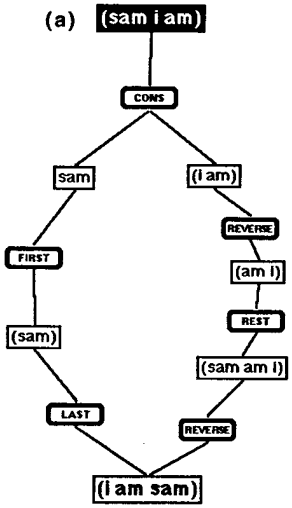
Figure 3. A student discovering an error in a problem solution using pop-up values.

Figure 4. A Visual SE run. Notice that the functions evaluated are blackened, the student's incorrect prediction is displayed over the *DEFUN*, and the student has used VSE's probe to see the output of the *CONS* function.



F16 1

(a)



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

(c)
```
=> (rotater '(i am sam))
   (sam i am)
```

FIG 2

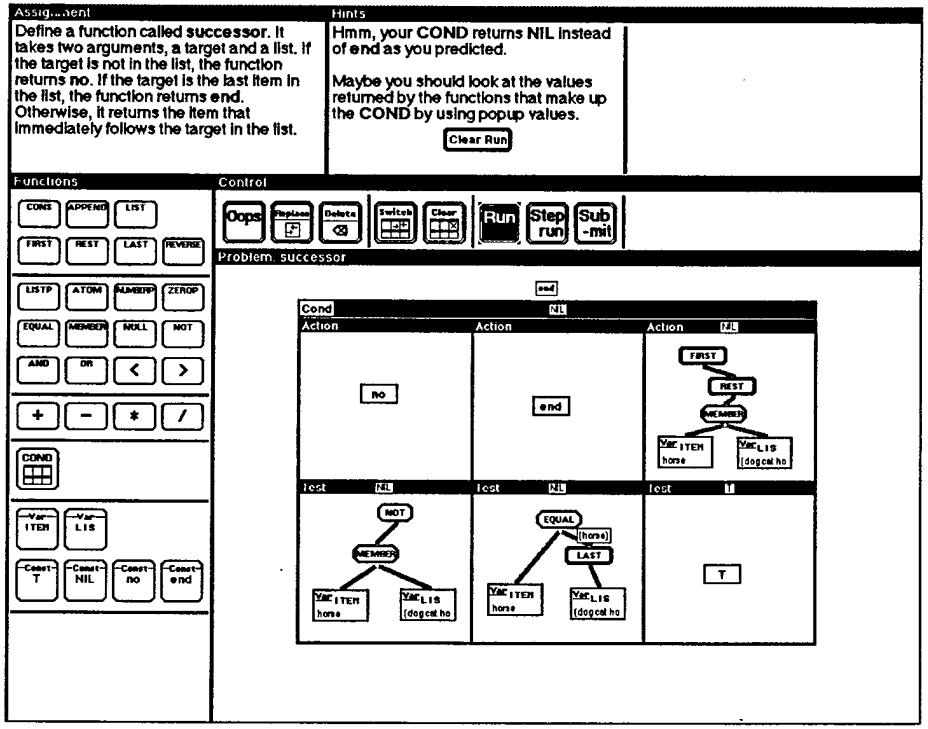
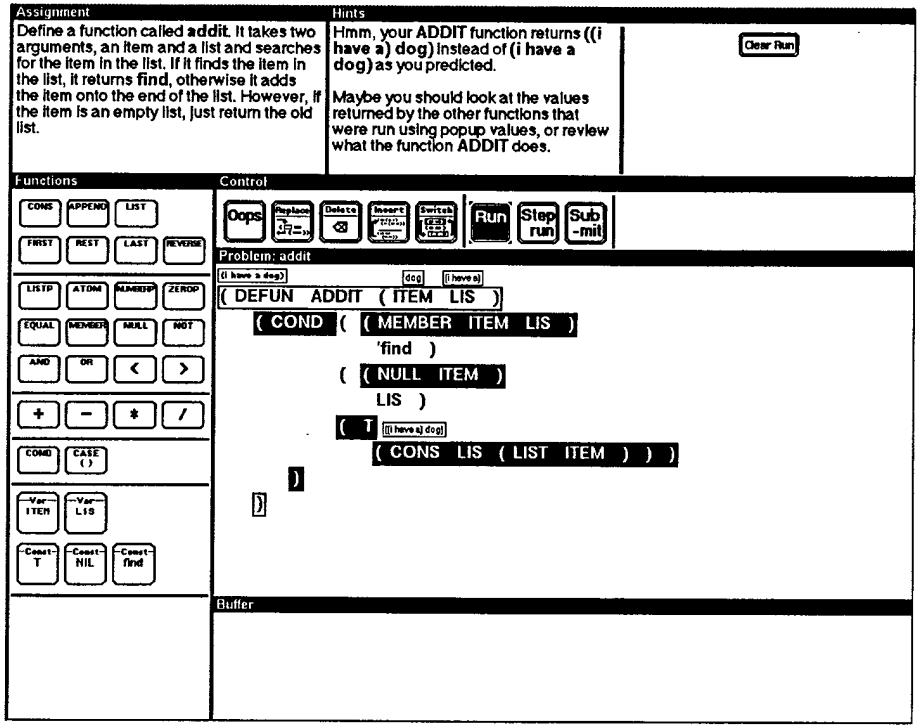


FIG 3


Assignment
Define a function called addit. It takes two arguments, an item and a list and searches for the item in the list. If it finds the item in the list, it returns find, otherwise it adds the item onto the end of the list. However, if the item is an empty list, just return the old list.
Hints
Hmm, your ADDIT function returns ((i have a) dog) instead of (i have a dog) as you predicted.
Maybe you should look at the values returned by the other functions that were run using popup values, or review what the function ADDIT does.
Clear Run
Functions
CONS APPEND LIST
FIRST REST LAST REVERSE
LISTP ATOM NUMBERP ZEROP
EQUAL MEMBER NULL NOT
AND OR < >
+ - * /
COND CASE ( )
Var ITEM
Var LIS
Const T
Const NIL
Const find
Control
Oops
Replace
Delete
Insert
Switch
Run
Step run
Sub -mit
Problem: addit
(DEFUN ADDIT (ITEM LIS)
(COND ((MEMBER ITEM LIS)
'find)
((NULL ITEM)
LIS)
(T
(CONS LIS (LIST ITEM)))
)
)
Buffer

F164