An Intelligent Tutor for Basic Algebra

David McArthur, Cathleen Stasz

RAND

The work described in this report was supported by the National Science Foundation under Grant MDR-8751515.

ISBN: 0-8330-1063-8

The RAND Publication Series: The Report is the principal publication documenting and transmitting RAND's major research findings and final research results. The RAND Note reports other outputs of sponsored research for general distribution. Publications of The RAND Corporation do not necessarily reflect the opinions or policies of the sponsors of RAND research.

Published by The RAND Corporation
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90406-2138

R-3811-NSF

An Intelligent Tutor for Basic Algebra

David McArthur, Cathleen Stasz

August 1990

Supported by the
National Science Foundation

RAND

PREFACE

This report describes research completed under National Science Foundation (NSF) Grant MDR-8470342 (Science and Engineering Directorate, Applications of Advanced Technologies Program) from September 1985 to February 1988. The goal of this research was to design, implement, and test an intelligent tutor for basic algebra. This report is the culmination of all the research accomplished under the original grant; segments of that research and follow-on work have been documented in previous RAND publications: D. McArthur, C. M. Stasz, and J. Y. Hotta, *Learning Problem-Solving Skills in Algebra*, N-2595-NSF, May 1987; D. McArthur, C. D. Burdorf, T. H. Ormseth, A. Robyn, and C. M. Stasz, *Multiple Representations of Mathematical Reasoning*, N-2758-NSF/RC, May 1988; D. McArthur, C. M. Stasz, J. Y. Hotta, O. J. Peter, and C. D. Burdorf, *Skill-Oriented Task Sequencing in an Intelligent Tutor for Basic Algebra*, N-2966-NSF, June 1989; and D. McArthur, M. W. Lewis, T. H. Ormseth, A. Robyn, C. M. Stasz, and D. A. Voreck, *Algebraic Thinking Tools: Supports for Modeling Situations and Solving Problems in Kids' Worlds*, N-2974-NSF, July 1989.

In addition to describing and explaining the specific software the authors developed, the report touches on broad issues of pedagogy, types of learning, curriculum concerns, and evaluative tools. Thus, this report's findings should interest not only developers and users of computer-based educational tools but also the educational community at large and mathematics instructors in particular.

Although the authors are reporting the final results of a now-completed grant, they are also discussing work in progress, for much research still needs to be done in this rich and relatively unexplored field. This new work is being funded by a follow-on NSF grant (MDR-8751515) and by a related NSF grant (MDR-8751104).

SUMMARY

This report describes research to develop an intelligent tutor for basic algebra. Our initial objective was to develop a one-to-one computer tutor that would help students understand algebra; we completed developing such a tutor for solving linear equations in the early stages of this study. However, our observations while developing that version led us to realize that any one tutor was too limited, both for our purposes and for those of the students. Therefore, we shifted our emphasis and focused on developing a generic tutor from which many tutor versions could be constructed. This report documents the rationale for redirecting the research effort, the development of the generic tutor, the development of tutor versions from that generic model, and our efforts to evaluate the tutor versions.

LIMITATIONS OF A SINGLE TUTOR

In developing and testing our initial computer-based tutor, we became aware of two major limitations of a single tutor. First, we discovered that a single tutor must focus on particular mathematical skills, and that different skills might best be acquired using different tutoring tools. Our initial tutor, for instance, focused on goal-directed reasoning skills and self-diagnosis skills. We concluded that a single tutor might be inadequate for effectively teaching the range of skills essential for true mathematical competence.

Second, a single tutor must incorporate a general set of guidelines the tutor uses in a tutoring session; that is, the designer must select a pedagogical policy about whether, how, and when the tutor will intrude into the student's problem-solving to correct mistakes, provide directions, and so forth. In the case of our initial tutor, we employed a "passive" policy, meaning that the tutor did not intervene unless the student requested help. The problem was that numerous pedagogical policies existed, but no data in the research literature demonstrated that one was superior to another. Because no one proven pedagogical way to tutor algebra existed, we decided to shift our focus from a single tutor tied to one pedagogical policy to a range of approaches to tutoring. Central to our study, therefore, was a decision to separate policy from mechanism.

DEVELOPING A CORE TUTOR

Our options were to develop one tutor and continue to develop software for that tutor over the years or to develop a core tutor from which other, substantially different versions could be created. For the reasons above, we chose to develop a core tutor that would permit various teaching approaches.

The core tutor is simply a collection of modules and tools—a generic framework—that can be used to construct tutor versions. One component of the core tutor embeds algebra knowledge and skills to help the student at each step in problem solving. A second part embeds intertask skills for proceeding from one problem to another in a session. The algebra expert component becomes part of each tutor version; it is simply plugged in. The version designer selects other capabilities. No single pedagogical module is built into the core; instead, the designer defines rules for several knowledge sources that collectively define pedagogical skill. From the core, then, a designer can select the capabilities he or she wishes for teaching a specific skill in a specific way.

The core tutor resolves the two problems we encountered in a single tutor. First, one or more tutor versions can be created specifically to emphasize the teaching of lower-level skills, if those are a curriculum's priority; others can be created to emphasize higher-level mathematical reasoning. By employing specialized tools, students can be taught multiple skills. Second, each tutor version constructed from the core tutor can have a different pedagogical policy. One can be highly intrusive (teacher controlled), another can be highly passive (student controlled), and another can combine the control (mixed initiative). In short, the core allows great variability and flexibility. By developing a wide range of versions from the core tutor, we are then able to test those versions to determine how well they achieve prescribed goals.

TUTOR VERSIONS

Although computer-based tutors can be designed to focus on mastering lower-level mathematical skills such as axioms and procedural rules, the tutor versions we created from the core tutor focused on higher-level mathematical reasoning. This procedure was in keeping with our belief that students could master lower-level skills and yet achieve only a limited understanding of other, perhaps more essential, skills. We wanted to create tutoring tools that would help students acquire such cognitive skills as heuristics, debugging, strategic planning, and problem solving.

All three of our tutor versions emphasized these higher-level skills, but we used a different pedagogical policy in each version. The first was the passive version, which we developed before the core tutor. It required students to perform all algebra symbol manipulation. The second was a multiple-representations version, which used two distinct representational systems to help students understand algebraic reasoning (traditional algebraic symbols and primitive objects, for example). This version intermixed symbol-manipulation reasoning and reasoning in an alternate representation. The third version was a goal-commands tutor that permitted students to solve algebraic problems using a set of commands. By automating lower-level manipulations, the tutor freed students from the details of symbol manipulation and helped them learn higher-level skills. In each version, the tutor appeared as a collection of menus and windows for the student.

From these versions, we derived nine basic principles we advocate embodying in the design of intelligent computer-assisted instruction (ICAI) systems:

1. Tutoring systems should be designed around expert systems that can be inspected.
2. Computer-based learning tools should be designed to support the specific cognitive processes involved in learning those skills through practice.
3. The student's reasoning should be externalized as much as possible.
4. Representations of mathematics within tutors do not have to be the traditional notation; other representations can sometimes facilitate certain skills more easily than can the traditional representation.
5. When multiple representations are used in a tutor, they should be connected dynamically—that is, so that a change in one representation results in a change in another.
6. Students should be allowed to focus on selected aspects of reasoning with multiple representations.
7. Teaching environments should permit novel sequencing of skills in mathematical curricula.

8. Students should be permitted to learn different knowledge levels of complex cognitive skills independently.
9. Environments for constructive learning must be provided.

EVALUATION

To date, our empirical evaluations of the versions have been only preliminary. Although one purpose for conducting these early tests was to determine whether the software was educationally effective, the main purpose was to guide us in redesigning the versions of the tutor. The evaluation indicated both success and failure: Student learning was low, but the evaluation informed the redesign of subsequent versions and resulted in our developing important data-gathering instruments and procedures.

One problem we encountered when trying to evaluate the passive tutor's effectiveness was that current standardized tests did not measure acquisition of the skills we were emphasizing. We therefore relied very little on standardized tests of mathematics achievement to assess our software. We have devoted considerable effort to designing new assessment tools that will evaluate the utility of software with multiple and ill-defined benefits. Our strategy has been to develop several kinds of measurement tools (achievement tests, attitude questionnaires, classroom observations, and student scripts), each focusing on a different skill, and we use the *combined* results for evaluation. We consider the development of the tools as important as the test results.

To test the passive tutor, we used it in five first-year algebra classes at Santa Monica High School in California. Testing intelligent tutors in the classroom instead of (as usually happens) developing them in the void of a laboratory is essential. Collecting and reporting student data, in fact, is one important asset of the study.

Achievement results based on students' pretest and posttest scores indicated that students showed a statistically significant change in achievement, but the change was of little *practical* significance. The evaluation further showed that students who did well on the pretest and considered algebra an easy discipline did well on the posttest, but students whose algebra skills were initially low showed little improvement. Perhaps the biggest disappointment was that the tutor failed to improve the students' model of math.

Student evaluations revealed several limitations of the tutor as a learning tool, including insufficient explanations of why the student had made an error, awkward wording of hints and explanations, and the tutor's passivity (that is, the student had to request assistance from the tutor). Keeping in mind these observations, we have implemented a variety of changes and additions to the software, including simplifying the interface, providing more detailed feedback, and making the tutor more intrusive when students commit errors. Nevertheless, these changes do not help the deficiencies in students' models of math.

Despite the unimpressive results in students' learning, the study was informative and constructive and will allow us to improve the capabilities of subsequent tutors and to measure their effectiveness more accurately. Much research remains to be done in the field before significant gains will accrue; therefore, to expect immediate gains from research is to judge prematurely the potential ICAI offers.

FUTURE DIRECTIONS

The ultimate goal of ICAI research is to develop computer programs that embed much of the subject matter, student diagnosis, and pedagogical knowledge of human tutors, permitting construction of powerful one-to-one computer learning environments. The research outlined here is only one step toward that ultimate goal. Our future work will attempt to enhance our lesson control module and add to the core's repertoire of techniques for aiding students at particular steps in a task.

To reach the ultimate ICAI goal, research must be undertaken on how to improve the pedagogical expertise of ICAI systems, whether new forms of learning might be more appropriate for computer-assisted instruction, what skills are important to acquire, whether the mathematics curriculum should be altered, and how to evaluate ICAI systems. These issues will require the combined efforts of various professionals, including cognitive psychologists, mathematics educators, professional mathematicians, and researchers in ICAI.

ACKNOWLEDGMENTS

The work reported here would not have been possible without the cooperation and enthusiasm of the students, teachers, and administrators at the Santa Monica High School. In particular, we thank Gretchen Davis and Edward Winkenhower of the Mathematics Department for sharing their teaching expertise and allowing us to work in their classrooms.

Andrew Molnar, our project monitor at the National Science Foundation, has been steadfast in his support of this work and other innovative research on advanced technologies for education. We thank him for his continual encouragement and for expanding our research vision.

Several current and former RAND colleagues made important contributions to this research effort. James Hodges provided statistical consulting and Tor Ormseth assisted with data analysis. Abby Robyn, Matthew Lewis, and Tor Ormseth, project staff members, contributed to the ideas and work reported here. John Hotta and Orli Peter, former RAND staff, provided research assistance. D. Robert Worley, John D. Winkler, and Daniel Koretz provided thorough and thoughtful technical reviews, which much improved this report. Joyce Gray assisted in preparing the manuscript for publication. Robert M. Brown ably applied his communication and writing skills in helping us produce this final document.

CONTENTS

PREFACE iii

SUMMARY v

ACKNOWLEDGMENTS ix

FIGURES AND TABLES xiii

Section

I. INTRODUCTION 1
- Research Goals 1
- Organization of the Report 3

II. THE CORE TUTOR 4
- Rationale for the Core Tutor 4
- The Core Tutor's Components 5

III. TUTOR VERSIONS 10
- Physical Specifications 10
- The Passive Tutor 10
- The Multiple-Representations Tutor 16
- The Goal-Commands Tutor 19

IV. EVALUATION AND ANALYSIS OF THE PASSIVE TUTOR AND ITS USES 24
- Multiple Methods for Assessment 25
- Preliminary Analysis of Student Outcomes 27
- Student Evaluation of the Tutor 34
- The Tutor's Limitations 35
- Changes to the Current Tutor 37
- Additions to the Tutor 38

V. CONCLUSIONS AND FUTURE DIRECTIONS 39
- Overcoming Competence Limitations 39
- Investigating New Forms of Learning 39
- Focus Changes 40
- Integration and Evaluation 41

Appendix: BACKGROUND QUESTIONNAIRE AND ACHIEVEMENT TEST 43

REFERENCES 57

FIGURES

1. The core tutor's components 6
2. The algebra tutor interface 11
3. Student using the *Go Back* option 12
4. Tutor explains a step for the student 14
5. Student receives a hint from the tutor 15
6. The multiple-representations tutor 17
7. Response to an incorrect step in the multiple-representations tutor 18
8. The goal-commands tutor using lower-level commands 20
9. The goal-commands tutor using higher-level commands 21

TABLES

1. Types of knowledge comprising algebra competence 3
2. Sample problems from the algebra achievement test 26
3. Mean achievement test scores for all groups 29
4. Item statistics for the achievement test 30
5. Selected correlations between student characteristics (at posttest) and posttest and change scores 31
6. Regression results for the posttest score 33
7. Regression results for the change score 34
8. Student evaluations of the tutor help options 35

I. INTRODUCTION

RESEARCH GOALS

It has long been recognized that most computer-assisted instruction (CAI) programs are generally less effective than human teachers (Bloom, 1984). The goal of our research has been to enhance the performance of CAI by applying techniques from artificial intelligence and cognitive science to the field of CAI. The aim of intelligent CAI (ICAI) programs is to help bring the skill level of educational software closer to that of human tutors, thereby potentially triggering a revolution in education. We envision a future in which each student has his or her own intelligent tutoring system, permitting students to learn subjects at a pace that suits their skills and in a style that adapts to their learning preferences. At present, such an environment is widely available only through one-to-one tutoring by humans. Although this one-to-one tutoring has shown impressive educational benefits (for example, Bloom, 1984), it is prohibitively expensive and usually available for only a few hours a week. The aim of ICAI systems is to make such learning environments inexpensive and unlimited.

Our discussion of research goals falls into three categories: first, the kind of approach to learning we wanted to implement in our software; second, the topics or subject matter we wanted students to learn using the software; third, the kinds of mathematical reasoning skills we hoped to support.

Target Learning Environment

At the most general level, our goal throughout the project has been to develop a computer-based learning environment for algebra that closely approximates a one-to-one learning experience with a human tutor. At this time, one-on-one tutoring appears to be the only teaching technique that reliably yields "two-sigma" improvements in student performance.

We can easily find reasons why one-to-one tutoring is so successful. Human tutors possess several kinds of expertise that allow them to provide detailed explanations of concepts and to personalize the learning experience to the student's needs:

- *Knowledge of the subject.* Tutors are expert problem solvers in their field.
- *Knowledge of the student.* Tutors understand which concepts are most difficult for students and are able to detect and help remedy student misconceptions.
- *Knowledge of teaching.* Tutors know pedagogical techniques that make difficult concepts easier for the student to understand. These techniques also determine what information to provide and when, thus expediting learning.

Our aim has been to use techniques from artificial intelligence and cognitive science to incorporate these kinds of expertise into an intelligent computer-based tutor for basic algebra.

Target Topics in Algebra

We initially chose to explore the learning of symbol manipulation skills required to solve linear equations and inequalities. We began with these topics for several reasons. First, solving linear equations is an enduring part of most curricula in basic algebra. Although many recent suggestions for curriculum reform (for example, Fey, 1984) propose eliminating various

familiar topics (factoring, for instance), linear equation solving remains a mainstay. Second, solving linear equations is a tractable topic, in addition to being an important one. Implementing the expertise for well-defined skills (such as symbol manipulation) is within artificial intelligence's state of the art; implementing the knowledge involved in less well-defined subjects is not. Third, we wanted to focus (at least for the present) more on issues of intelligent tutoring than on issues of curriculum reform. Some colleagues have wondered whether developing software for novel topics in mathematics, rather than for familiar aspects of symbol manipulation, might not be more educationally valuable. Many educators would agree that hand-held calculators have made arithmetic less important for students to learn. By analogy, because symbol manipulation can be performed by machines, isn't it also a devalued skill?

Clearly reforming the algebra curriculum is an important task. However, developing computational principles of intelligent tutoring systems is important in its own right. Because the competence of current ICAI systems falls far below that of human tutors, implementing a new curriculum using state-of-the-art ICAI techniques would not produce revolutionary educational software. Both the goals of curriculum reform and of developing a better computational theory of tutoring must be accomplished before we can expect to see substantial improvements in educational software.

Our approach has been to use basic symbolic algebra as a realistic test subject for developing generalizable tools and ideas concerning ICAI. The lessons we have learned in this area will likely extend to other areas of mathematics—including whatever new topics appear in the curriculum over the next few years. In addition, our study of tutoring in basic algebra has led us to a better appreciation of which computer-based tools might be part of a new high school curriculum for mathematics. We are now beginning to pursue these ideas.

Target Levels of Mathematical Reasoning

Basic algebra can offer a surprisingly rich set of skills for the student to learn. Although most traditional algebra curricula focus on teaching facts and procedural rules, these comprise only a subset of the different levels of skill or knowledge involved in algebra competence. Table 1 shows a more complete list of the kinds of skills that can be acquired, even in the context of learning to solve simple linear equations (see also Schoenfeld, 1985; Collins and Brown, 1987). Our own classroom observations, along with the analyses of others (for example, Schoenfeld, 1985), indicate that a student who has mastered the lower-level skills in Table 1 (for instance, algebra axioms) may have little understanding of the other skills essential to algebra competence (broadly defined). In this project, we have focused on developing computer-based tools that help students learn the higher-level mathematical reasoning skills of Table 1 and have focused less on the lower-level ones.

In our view, curriculum reform for mathematics has at least two dimensions. On the one hand, the curriculum *content* should change in response to the changing nature of mathematics, its real-world applications, and the newly available computer tools for automating mathematics. On the other hand, along with others (for example, Schoenfeld, 1987; Pea, 1987), we believe that a change in *emphasis* is also necessary. Many schools emphasize a "cookbook" approach to mathematics education, which stresses the learning of rote algorithms and largely ignores the heuristic- and strategic-level reasoning that are crucial aspects of mathematical thinking. Our tutor versions have focused on this dimension of curriculum reform.

Table 1

TYPES OF KNOWLEDGE COMPRISING ALGEBRA COMPETENCE

Knowledge Type	Description and Example
Facts or axioms	The axiomatic truths of algebra—for example, the distributive rule, usually stated like $a(b + c) = ab + ac$.
Procedural rules	Algorithms, usually comprising a fixed sequence of steps, that are guaranteed to produce the desired result when applied in the correct context—for example, the FOIL method of expanding binomials (multiply the first elements of each pair in the binomial, the outside elements of each pair, the inside elements, then finally multiply the last elements; add all the multiplications).
Local heuristics	Rules that indicate the conditions under which a given goal or transformation might be reasonable. Local heuristics are not algorithms, because they usually suggest operations that don't completely solve a problem, and the operations, although locally reasonable, might not lead to an overall optimal solution. An example is the informal rule, taught to students, to "always remove parentheses first when solving an equation."
Metacognitive skills	General skills concerning the control of inferencing when problem solving—for example, self-monitoring skills, debugging ability, knowledge of techniques such as decomposing complex goals into more manageable ones.
Problem-finding skills	General skills concerning the ability both to find situations in the world in which known mathematical tools might be useful and to develop formal mathematical models of situations using those tools—for example, seeing that two properties of objects (such as size and age) are related and then developing equations that formalize the relationships between the two variables.

ORGANIZATION OF THE REPORT

The report is organized from the development of the software to its evaluation. Section II describes our "core" tutor, comprising both a generic framework for viewing ICAI systems and a set of tools and components we used to construct our specific tutor versions. Section III discusses several tutor versions we created using the core components and tools. In Sec. IV, we discuss our empirical evaluation of the "passive" version of the tutor and some limitations of our current tutoring tools for algebra. In addition to presenting the results themselves, we also describe the methods we developed to obtain our data. Finally, Sec. V discusses our plans for future research. We attempt here to generalize beyond our own software, describing several ways in which ICAI systems might be improved.

Because this report attempts to survey all facets of the research we have accomplished, it necessarily treats some topics briefly. The reader who wishes more detail may consult the papers cited in various sections.

II. THE CORE TUTOR

Our original plan was to implement a single tutoring system for all of basic algebra; proceeding on this assumption, we developed an initial tutor (see McArthur, Stasz, and Hotta, 1987). However, we subsequently changed our basic approach for developing ICAI programs; instead of developing a single tutor that would undergo incremental changes, we decided to develop a core that could be used to create several substantially different tutor versions.

RATIONALE FOR THE CORE TUTOR

Several considerations caused us to redirect the focus of our research. First, we quickly realized that no one "right" way to tutor algebra existed. Hence, rather than examining one approach in detail, we felt that developing software that permitted the implementation and testing of various pedagogical approaches would be more appropriate. For example, we could consider highly intrusive tutors (teacher controlled), ones where the student made most of the decisions (student controlled), and ones that combined both (mixed initiative). Second, we believed that different skills would best be served by different kinds of tutoring tools. Table 1 shows how learning even basic algebra demands that the students learn a variety of skills. Instead of attempting to provide a single computer-based (or even non-computer-based) environment that attempts to tutor all such skills, we decided that implementing specialized tools, each designed to help the student learn one or two component skills, would be more profitable. Third, we came to believe that *multiple views* or representations of concepts and topics were crucial to learning. Consequently, we developed several tutoring versions that presented varied views of algebra and algebraic reasoning to the student.

To facilitate the construction of multiple versions of the tutor, we constructed the core tutor, though it is not actually a tutor version at all. Rather, it is a collection of modules and tools to help us build new tutoring versions. In addition to affording the benefits listed above, the core tutor also saves time in creating tutor versions. For instance, the first version (which was created before the core tutor) took approximately a year to build, but subsequent versions using the core tools have taken from two days to two months.

Although specific tutors can also be constructed from "tutoring shells" (see, for example, Sleeman, 1987), the core tutor is substantially different from tutoring shells in the way it facilitates the construction of tutor versions. Generally, tutoring shells make constructing a tutor easy by drastically reducing the number of options available to the designer. This typically limits tutorial policies to ones that equate tutoring with local interventions triggered by local problems—a kind of fire-fighting model of pedagogy.

The modules and tools that comprise the core tutor impose fewer restrictions on the designer. For example, the programmer is free to specify a tutorial strategy that involves more than just how to deal with student errors. On the other hand, the core tutor also imposes more responsibilities on the programmer. This trade-off strikes the right balance given the state of the art of ICAI. To exaggerate only slightly, many tutoring shells give the impression that the main research problems of ICAI concerned with understanding human didactics and pedagogical skills have been solved. The primary remaining problem is technical: to provide a means of communicating the domain-specific performance and pedagogical rules to the machine as

efficiently as possible. However, many fundamental questions regarding the cognitive skills of teaching remain unanswered (in Sec. V, we discuss some of these issues in greater detail). Accordingly, our core tutoring tools provide relatively weak constraints on the representation and content of tutors.

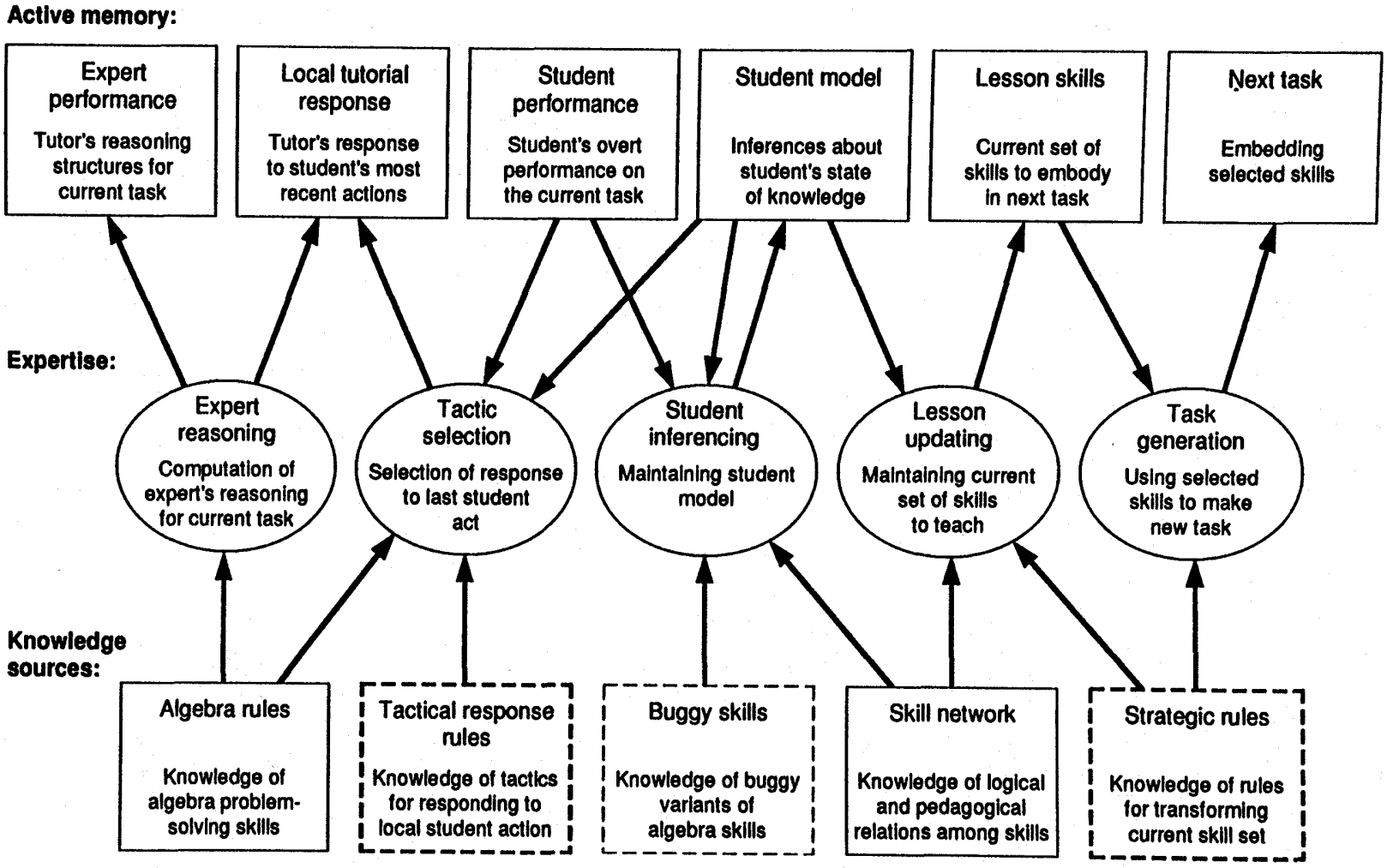
THE CORE TUTOR'S COMPONENTS

The core tutor consists of modules and tools to help build each of the main parts of a tutor version. To convey the core tutor's components, Fig. 1 provides an overall model of our view of a generic intelligent tutoring system. The model includes three kinds of components. *Active memory* components encode data structures that the tutor computes during a tutoring session. Often this information is transitory; for example, the current student performance changes from task to task. However, we emphasize that this information is not necessarily short term in nature; aspects of the student model may endure across several tutoring sessions. The components of our model labeled *expertise* perform the computations and inferences that result in active memory structures. In order to make intelligent computations, expertise exploits various kinds of knowledge sources. *Knowledge sources* encode general kinds of information about students, teaching, and subject matter. This information is distinct from data represented in active memory about specific students, teaching situations, and tasks. Knowledge sources are often referred to as long-term memory because their contents change little over time.

Although the model in Fig. 1 appears complex, its organizing principles are actually quite simple. Broadly, the tutor reasons at two levels. At the local or *tactical* level, the tutor decides how to respond to the student's most recent responses (for example, algebraic transformation or a request for help), and at the *strategic* level the tutor plans the tasks or problems to give to the student throughout the session.

Memory, expertise, and knowledge that contribute to local or tactical decisions appear in the left half of Fig. 1; strategic modules appear on the right. The tutor's ability to reason about the current task (generating an "ideal" solution) allows it to provide several types of rich feedback on students' solutions. Tactical response rules are used to decide which type of local feedback to choose at any time. To tailor the local feedback appropriately, knowledge of the student's performance and inferences about the student's state of knowledge (the student model) are also important. In addition, the student model is essential for updating the set of skills that are the global (strategic) topics of tutoring during a session. The skill network encodes all the skills or concepts that might become topics of tutoring, as well as the strategic rules that dictate how to move between topic skills. Lesson updating uses both this knowledge and knowledge of the student's current conceptual strengths and weaknesses (in the lesson skills and student model) to reason about new topics for tutoring. The final link in the tutor's strategic reasoning involves generating new tasks that will elicit the skills that are the current topics for tutoring. The following subsections document in more detail the roles of various components involved in tactical and strategic reasoning.

When constructing a version of the tutor, the programmer needs to specify the "rules" (or other representations of knowledge) that comprise the various knowledge sources in the generic system. The reasoning processes comprising the expertise modules are given, and the working memory modules are constructed by student and tutor behavior during a session. The core tutor facilitates the definition of knowledge sources in process in several ways. First, some knowledge sources come built in. (In Fig. 1, these sources have solid borders.) For these, the



NOTE: Knowledge sources with a solid border are provided by the tutor as a built-in part. A heavy dashed line indicates that the tutor provides tools that simplify the definition of rules in that knowledge source. A lightly dashed line indicates a built-in knowledge source that is currently only in skeletal form.

Fig. 1—The core tutor's components

programmer of a new version need do nothing. Second, for other knowledge sources, we provide tools to facilitate the definition of the appropriate rules. (In Fig. 1, these sources have heavy dashed borders.) In the following sections, we discuss the various built-in parts and tools.

The Algebra Expert

The algebra expert is an artificial intelligence–based program that encodes the algebra knowledge common to all versions of the tutor. It is an algebra "expert system," comprising a knowledge base of rules (*algebra rules*) and an inference engine (*expert reasoning*) that applies the rules to a given algebra problem, generating reasoning structures (*expert performance*). The reasoning generated by the algebra expert has several features. First, the reasoning paths are natural in the sense that the sequence of reasoning steps generated by the system in solving a problem is always comprehensible to students and in many cases is similar to their steps. Second, the system generates multiple reasoning paths in the sense that it usually produces several different plausible solutions to a given problem. McArthur (1987) describes the algebra expert in more detail.

Programs whose reasoning and final answers are comprehensible to humans are often termed *inspectable* (McArthur, Stasz, and Hotta, 1987). From an educational standpoint, inspectable systems have several advantages over *black-box* systems, whose reasoning processes are generally incomprehensible. The uses to which we have put our algebra expert amply demonstrate these advantages:

- *It models expert reasoning.* It provides the student with ideal reasoning paths for solving a problem, annotated with a discussion of the goals and operations that justify each step in the algebraic symbol manipulation for the problem (see Sec. III for examples).
- *It explains expert reasoning.* It provides natural language explanations of the goals underlying reasoning steps in a solution.
- *It articulates different levels of knowledge.* The algebra expert's knowledge is organized by different knowledge levels (Newell, 1982). The knowledge levels used to solve linear equations, for example, include *plans*, which encode the high-level policies about how to approach a problem; *strategic rules*, which decide what kind of algebraic transformation to apply at any point in problem solving; and *axioms*, which encode the algebra-specific rules for how to implement a transformation. In short, the algebra expert can reason separately about higher-level problem-solving decisions and lower-level algebra-specific decisions. In general, this permits us to tutor these different levels of mathematical thinking in isolation, as subsequent sections on specific tutor versions will demonstrate.
- *It provides hints and remedial assistance.* When the student needs help or makes a reasoning error in solving a problem, the algebra expert can access the appropriate levels of knowledge to describe what kind of transformation to choose next and why to select it.

Pedagogical Tools

Unlike the algebra expert component, which is simply "plugged in" and becomes part of each tutor, the core tutor's pedagogical capabilities are not built in. No single pedagogical component exists that the version designer can simply plug in to make decisions about *when* to

show ideal solution lines, explanations, hints, or remedial help. We have not defined a pedagogical module because *there is no single good policy for tutoring, or even a small set of good policies*. Instead, we provide tools that permit the designer to define rules for several knowledge sources; these tools collectively define pedagogical skill in our overall model. In essence, we have separated the pedagogical policy from the mechanism in order to allow a designer to define whatever policy he or she prefers in a particular tutor version. The tools included in the core tutor are divided into those that help define rules for local tactical responses—structuring tutoring sessions at an *intratask* level—and those that help define strategic rules—organizing sessions at the *intertask* level.

Intratask Pedagogical Tools. Intratask tutoring skills are responsible for controlling the dialogue with the student as he or she solves a single problem or a single step within a problem. The overall goal of intratask tutoring expertise is to use a relatively rich set of current conditions (*student performance* information and diagnostic inferences in the *student model*) as a basis for computing a specific tactic (*tactic selection*) to instantiate. When instantiated, a tactic generates a tutor response—for example, a hint, error description, or explanation (*local tutorial response*). These decisions are tactical in the sense that they are computed opportunistically, as a function of changing conditions, rather than as part of an enduring tutorial plan.

The current core tutor provides a simple language for creating *tactical response rules* that define an intratask tutorial policy. The core tutor has no "default" policy. The designer is responsible for writing a set of rules that covers all student responses and stipulates a tutorial action for each type. The following two rules are translations of ones that implement two distinct tutorial policies.

1. If the student has just finished typing in a new step, display the step in the display window.
2. If the student has just finished typing in a new step, determine the validity of the step. If the student's step is invalid, say it's incorrect and describe the kind of error.

Although these rules are simple, they show how radically different intratask policies can be. The first rule was part of a completely passive policy in which students were never given unsolicited advice. The second rule was part of a completely autocratic policy in which the tutor, not the student, controlled the initiative.

Intertask Pedagogical Tools. Intertask tutoring skills decide how tasks or problems should be sequenced. In general, although ICAI programs possess considerable expertise in dealing with local student difficulties within a problem, they have little ability to organize a lesson or a sequence of problems (see Ohlsson, 1986). Elsewhere (McArthur et al., 1988) we review in detail the planning limitations of ICAI systems. The main challenge in implementing intelligent intertask tutorial control involves reconciling the need for lesson plans, which impart structure to a session, with the need to respond opportunistically to each student's difficulties as they arise. This question needs extensive research; our work in this area should be regarded only as a first step.

Our main effort concerning intertask tutoring has been to develop a lesson control module for algebraic equalities and inequalities. Lesson control is the ability to generate an intelligent sequence of tasks for the student. The overall flow of our model of lesson control appears in the modules on the right side of Fig. 1. We assume tutors begin with a set of component skills necessary to solve problems in the chosen domain (the skill network module of Fig. 1). At any given time, some subset of all skills is assumed to be the current topic of tutoring (lesson skills). Tasks are generated that will tap those skills (task generation). Depending

on the student's performance on the task and on inferences about student performance maintained in a student model (student performance and student model components), the tutor will first update the target set of component skills (lesson updating) and possibly make new inferences about the student's state of knowledge (student inferencing). The cycle repeats as the tutor generates a new task. If the current skill set is appropriate for the student, lesson tasks will not be too difficult or too easy. Given that questions are generated using an incrementally modified skill set, the task sequence should be coherent as a whole while still being responsive to student performance.

Our current implementation of lesson control includes representations of a simple student inferencing component and complete lesson updating and task generation capabilities for basic algebra. The skill network is relatively complete (for basic linear algebra) and can be extended in a simple fashion to new concepts outside of linear equation solving. (In one day, for example, we added a network for inequalities using a simple language for representing new skills.) However, the strategic rules knowledge source exists largely in skeletal form. Instead of supplying a complete set of rules, we have built a simple language in which designers can readily construct differing lesson control policies in this knowledge source.

We have implemented several preliminary policies to test the lesson control language. The strategic rules for the current lesson control structure can each be thought of as functions that map current skill-state information and student performance information onto a new skill-set state. Two skill-set constructs are referenced: the most recent skill set used to generate a question (Sk) and the lesson stack (LS). The lesson stack stores skill sets and permits the tutor to suspend one tutorial goal (for example, a skill set), interpolate another goal (for example, focus on a new skill set), and then resume the original goal.

The following are rough English translations of some rules used in the current control structure:

- If the student's score on the past six questions totals < 350 and they are all from the same skill set, push Sk on the lesson stack and set Sk to be a simpler skill set.
- If the student's score on the past four questions totals 370 and they are all from the same skill set and the LS isn't empty, pop the LS and set it as Sk.
- If the student's score on the past four questions totals 370 and they are all from the same skill set and the LS is empty, select a more complex skill set.

The rules are doubtless too simple to yield intelligent lesson control in all cases. For example, all student modeling information has been collapsed into numerical scores. However, although many current rules are questionable, they have an important value as *tools for constructing alternate pedagogical policies*. Different rule sets can easily be constructed, encoding a variety of different theoretical approaches. Because little research describing the important features of intelligent sequencing has occurred, our framework can provide a useful way of empirically testing alternate ideas about sequencing.

III. TUTOR VERSIONS

In the following subsections, we describe several tutoring versions or tools we have implemented over the past two years. In addition to describing how each version operates, we attempt to illuminate the underlying justification, or abstract tutoring principles, that have guided the version's design. In many ways, these principles are as important as the tutor versions themselves. The particular tutors we have developed will be superseded, or radically changed, in the coming years. We hope their underlying principles will endure.

PHYSICAL SPECIFICATIONS

The initial version of our tutor was implemented in a computer language called Zetalisp on a Symbolics 3600. Preliminary "lab" tests of the software were conducted on this version at The RAND Corporation using students from Santa Monica High School in Santa Monica, California. Subsequent versions of the tutor have been implemented in Franz Lisp (another dialect of the Lisp language) on Sun Microsystems workstations. These run in both a lab and school setting. The lab setting continues to use paid student volunteers from the local high school. In February 1987, six Sun 3/50 workstations were installed in the local high school. Except for summer vacation, the machines have been in continuous use in several different first-year algebra classes. Over the next several years, we plan to continue developing and testing the software, both in the lab and school settings.

THE PASSIVE TUTOR

The first tutor version we created was passive, meaning that all initiative was under the student's control. In this section, we discuss the rationale for this version and then its implementation. We go into considerable detail because subsequent versions share many features of this initial version. (Additional information on the passive tutor version appears in McArthur, Stasz, and Hotta, 1987.)

The Passive Tutor's Tutorial Goals

We had several goals in developing our first version of the tutor. First, we wanted to complete an initial implementation of each component of an ICAI system (see Fig. 1). Second, we decided to construct a tactically passive tutor, largely because we found no principled reason to prefer a different kind of local tutorial policy.[1] Third, we wanted to structure the tutoring environment to permit students to focus on select higher-level problem-solving skills (see Table 1). In this version, we focused in particular on two kinds of problem-solving skills not found in typical curricula: debugging and self-diagnosis skills, and goal-directed reasoning skills.

[1]Most ICAI systems embed a more active local tutoring policy in the sense that the tutor takes the initiative in deciding when to give the student advice. No principled reason exists for adopting either kind of policy because few empirical data exist on the kinds of techniques used by good human tutors. We have recently begun to develop a cognitive model of human tutoring skills (McArthur and Stasz, 1987; Shavelson et al., 1989; and McArthur, Stasz, and Zmuidzinas, in press).

Debugging and self-diagnosis skills refer to the abilities of good students to learn from their errors—to track down the error's location and determine how to find the knowledge that will remedy the error. Our own research shows that students' debugging skills are almost nonexistent in algebra. The problem-solving skills involved in debugging are critical: They can make the difference between students who merely practice when doing homework questions and those who *learn* from practice.

Goal-directed reasoning skills enable the student to break down problems into subproblems recursively until a piece of axiom-level knowledge can be selected to satisfy a particular goal. These contrast with cookbook procedures commonly taught in the classroom. Such procedures give the impression that algebra problem solving always involves a fixed set of steps. The goal-directed reasoning skills we refer to are true problem-solving techniques that selectively search for patterns in questions that suggest certain goals or approaches.

Implementation of the Passive Tutor

The student sees the tutor as a collection of windows and menus, as Fig. 2 shows. The large window in the upper right is the "display" window, where the student's reasoning is recorded and queried. Problem solving is represented here as a *reasoning tree*. Each branch in the tree represents an alternative solution or line of attack on the problem. A tree representation allows easy comparison of different solutions, both the student's and the tutor's. The boxed equation represents the student's current focus of attention in problem solving. To the left of the display window are several menus, comprising a set of options that permit users to manage their reasoning and allow the tutor to assist students in reasoning in various ways.

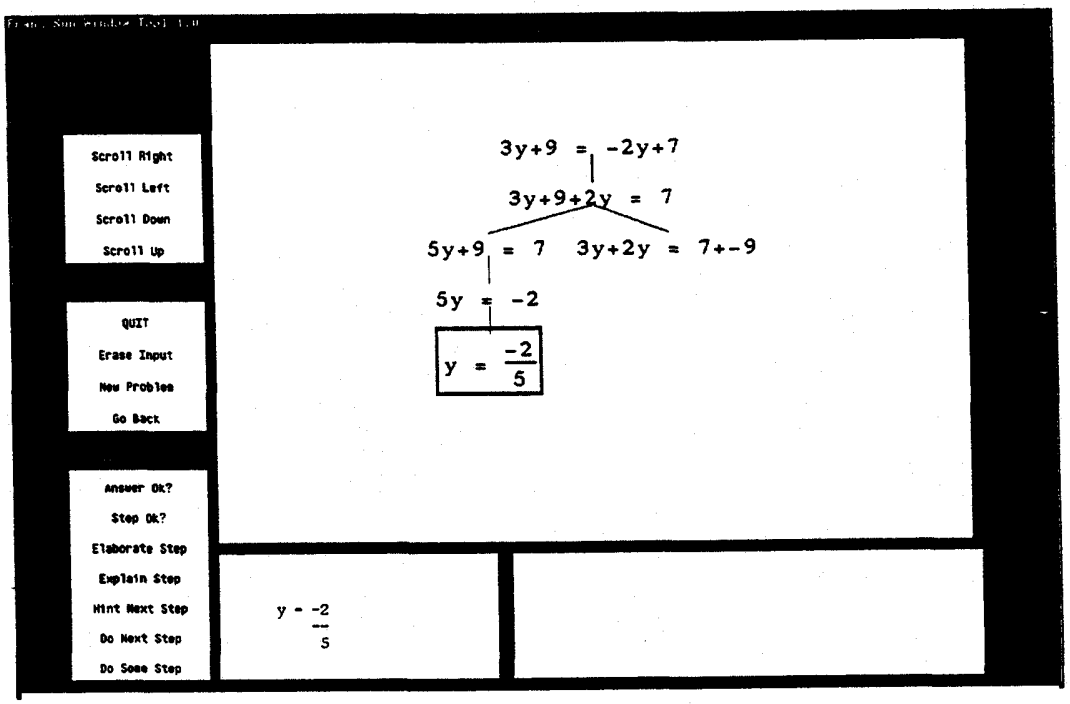


Fig. 2—The algebra tutor interface

Below the display window, on the right, is the "comment" window, where the tutor sends textual feedback to the student. To the left is the "work" window, where the student creates each new line in his or her solution. New lines or reasoning steps can be created either by typing equations using the keyboard or by writing them on an electronic tablet that recognizes the strokes as characters and passes them to the tutor, where they are interpreted as mathematical expressions.[2] When the student has finished inputting a line, the tutor places the new expression in the display window.

Support for Learning Debugging Skills. The first way the tutor facilitates the learning of debugging skills is by encouraging mistakes and not penalizing them. If the student determines that the current solution line is not correct, or just wants to investigate a new line of attack on the problem, he or she can easily try alternatives. To determine whether the answer is correct, the student uses *Answer Ok?* to find out if the solution is acceptable. Next, the student can select the menu item *Go Back*. The tutor will then ask which solution expression in the display window the student now wants to be the problem focus. The student responds by using a mouse to point to any expression in the reasoning tree and selects it by clicking one mouse button. For example, in Fig. 3, having been told that the answer ($y = 32/3$) is wrong, the student has just clicked on *Go Back* and then selected the equation $-6y + -15 = -17 + -9y$. The tutor then moves the box that indicates the current focus of attention to this equation or node.

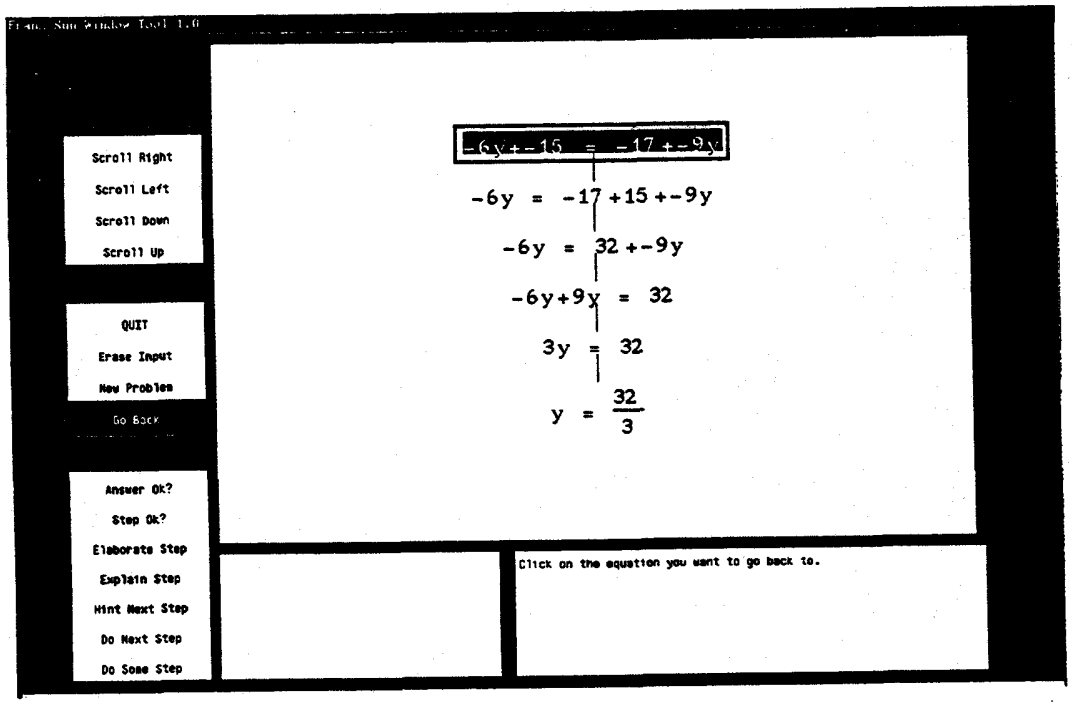


Fig. 3—Student using the *Go Back* option

[2]We are using a Penpad, a product of Pencept, Inc., for input of handwritten characters, and initial results look encouraging. The Penpad recognizes approximately 95 percent of characters and places only a few constraints on students' penmanship.

The *Go Back* option provides a way for the student to try out problem-solving strategies efficiently. The traditional pencil-and-paper medium does not encourage this exploration and learning because it exacts a high cost for trying multiple solution lines. To try a new line, students must erase the old one, which they are reluctant to do because the process is slow and because they may forget the old line if they wish to return to it.

Debugging is a multifaceted skill; merely encouraging students to make and explore errors will not make them skilled at debugging. In addition, students need to learn how to isolate a bug and fix it. Isolating an error may be problematic because the tutor's problem-solving environment encourages students to try out many reasoning steps. To learn from their mistakes, students may have to wade through these steps to find the one step that hides a misconception. In the case of Fig. 3, for example, how does the student know which step is the culprit? Several menu item options will help simplify the search. First, the *Step Ok?* item allows the student to ask the tutor if any step in the reasoning tree constitutes an appropriate mathematical transformation in the current context. The student selects *Step Ok?* and then points to any step in the reasoning tree.

When the student has selected a reasoning step, the algebra expert system then says whether the step from the previous node to the selected one is acceptable. In critiquing the student's step, the tutor makes a distinction between steps that are mathematically *invalid* and steps that are *inappropriate*. Inappropriate steps are logically valid but would not be chosen by the algebra expert because they do not move the student closer to a solution. Thus, using *Step Ok?* the student not only succeeds in isolating a faulty reasoning step but also obtains a characterization of the misconception underlying the error. Invalid steps imply errors in knowledge of algebra transformation rules; inappropriate steps imply errors in goal-directed reasoning methods. Having successfully used *Step Ok?* to isolate an error, the student is now in a position to execute *Go Back*, as in Fig. 3.

Support for Learning Goal-Directed Reasoning Skills. The supports we provide for helping the student understand that solving algebra problems can be a goal-directed reasoning process, as opposed to a cookbook procedure, are divided into two classes. First, we enable the student to observe the tutor's own exemplary reasoning processes in considerable detail. Second, we provide aids for the student's problem solving in the form of hints that reveal the goal structure of algebraic problem solving. We describe these tools in turn.

Several menu items allow students to "open up" the tutor's reasoning process to a depth and detail that students tailor to their cognitive needs. If, in viewing the expert's reasoning, the student does not understand how the expert went from one expression to the next, he or she can use *Elaborate Step* to see more detailed intermediate steps that may illuminate the expert's reasoning. Although the *Elaborate Step* option allows the student to view selectively more of the reasoning substeps that comprised a larger step, the option *Explain Step* provides more textual justifications of how and why those steps were taken.[3] The tutor's explanation capability is particularly powerful because it can exploit its hierarchical reasoning structures to create multileveled explanations.

At the start of Fig. 4, for example, the student selects the *Explain Step* menu item and then points to the indicated step. The tutor begins by bringing up an explanation window (on the right) and, below it, an explanation menu. In the explanation window, the tutor prints the basic explanation describing the generation of the step. Each explanation placed in the win-

[3]The original explanation facility in the tutor was programmed by Mike Pazzani, a graduate student at UCLA.

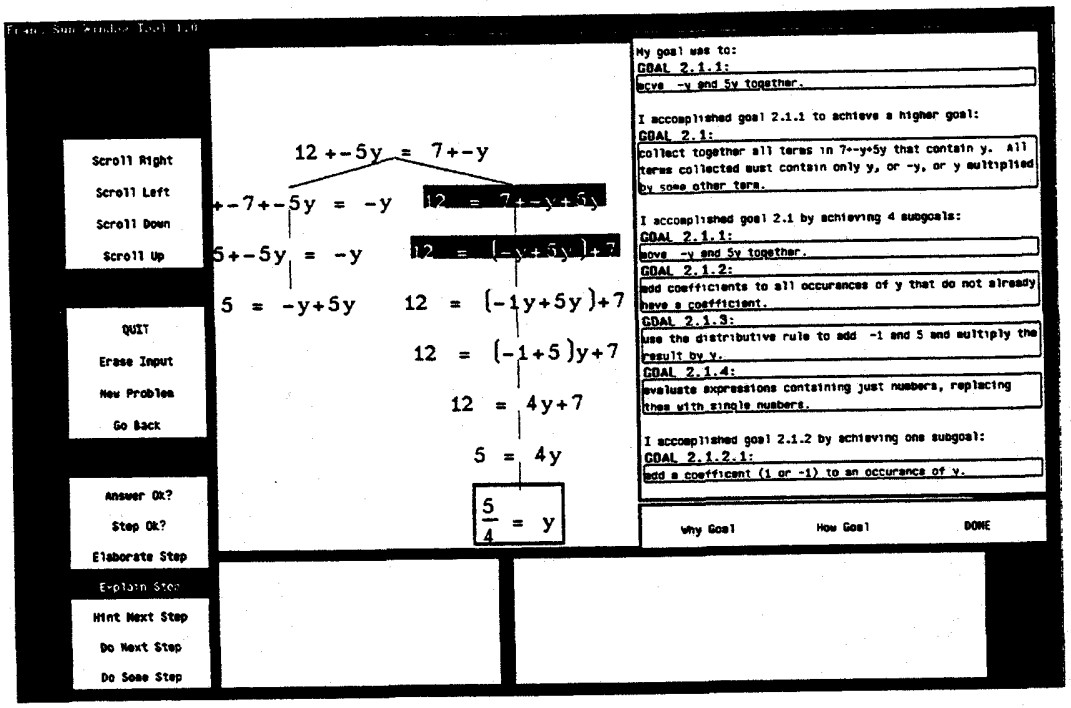


Fig. 4—Tutor explains a step for the student

dow is boxed and prefixed by a goal number. Goal numbers explicitly show how goals are connected in a hierarchy of subgoals and supergoals. Once a basic explanation is displayed, the student has an opportunity to deepen the explanation and tailor it to his or her own needs using the menu items *How Goal* and *Why Goal.* The student first selects one of these items and then clicks on a goal in the explanation window by placing the mouse in the box describing a goal. A new goal description is then placed in the explanation window; its goal number reflects its subgoal/supergoal relationship with the goal just selected.

The reasoning structures so useful in generating explanations for the student have various other important tutorial uses. One of the most important uses concerns hints. When students are stuck at a given point in solving a problem, we generate hints to try to get them back on track. Using the reasoning structures that the tutor produces when answering a question, we can generate hints at many levels. For example, Fig. 5 shows the hints we can now generate for the simple equation $5w + 8 = 9$, from the most general to the most specific. The student controls the specificity of hints generated by selecting *More Detailed Hint* as needed.

Principles Derived from the Passive Tutor

By way of summary, we list below some important principles we feel are embodied in the passive tutor. These principles are sufficiently general to inform the design of other ICAI systems (including other versions of our tutor), as well as of non-computer-based tutoring tools.

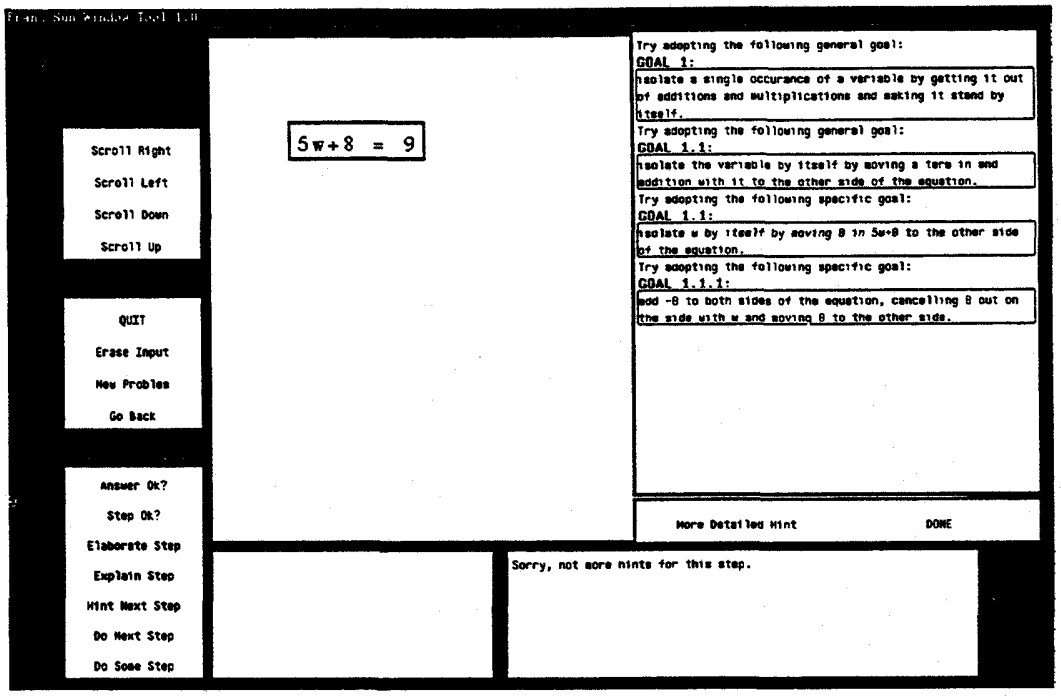


Fig. 5—Student receives a hint from the tutor

1. *Design intelligent tutoring systems around inspectable expert systems.* This lesson is not new with us (see, for example, Brown, Burton, and deKleer, 1982), but our experience underscores it. In reviewing other ICAI systems, we have found *no* exception to the rule that most of the potential educational value of a system derives from its ability to generate reasoning paths that are comprehensible to students. This ability provides the basis for several tutoring skills of general value, including *modeling* (showing the "ideal" reasoning paths), *explaining* (interactively justifying reasoning paths), and *coaching* (providing hints or descriptions of the kinds of reasoning the student should consider).
2. *Design computer-based learning tools that support the specific cognitive processes involved in learning chosen skills through practice.* Any learning tool will be more effective if it is based on an analysis of the specific cognitive skills students must execute in accomplishing the desired goal and if facets of the tool are specifically designed to support the component skills. For example, debugging has several components, including the execution of (faulty) reasoning steps, review of the steps, localization of erroneous steps, and fixing of the error. Consequently, to assist students in learning debugging skills, our tutor includes facets that both delimit and facilitate each of these activities.
3. *Reify or externalize reasoning.* In several ways, the tutor attempts to externalize the student's reasoning as much as possible. For example, several different student and tutor reasoning paths can coexist, showing different approaches to a problem. The

tutor also externalizes its reasoning by providing hierarchical explanations of its goal structure. Such reification reduces student memory load, permitting the student to concentrate on problem solving rather than on retention and rehearsal; it also emphasizes the process of problem solving as opposed to the product (Brown, 1984). Students can learn more by studying the process of faulty reasoning than by studying just its results, because the cognitive process of reasoning is what must be "repaired" if one is to learn from practice. Tools that permit reasoning to be recorded and "played back" (Collins and Brown, 1987) can greatly facilitate the process of learning from errors.

THE MULTIPLE-REPRESENTATIONS TUTOR

Using the passive version of the tutor, students are confined to solving problems by manipulating algebraic symbols in the traditional notation.[4] In this subsection, we describe a version of the tutor that permits students to solve problems in two distinct representational systems and to see the connection between the two.

Tutorial Goals of the Multiple-Representations Tutor

Several goals guided us in developing a tutor version that used multiple representations of algebra and algebraic reasoning. First, we wanted to test the tools of our core tutor. Would they facilitate the construction of a new version of the tutor? Second, we wanted to explore the possible leverage of combining an intelligent tutor with the idea of multiple representations of algebraic reasoning. Several researchers (for example, Dienes, 1960; Kaput et al., 1986; Kaput, 1986; and Lesh, Behr, and Post, 1987) have described alternate representations of mathematics and conjectured about their educational benefits. However, no one has yet implemented an environment in which it is possible for *both* the system and tutor to reason in different representations, not just the student.

Implementation of the Multiple-Representations Tutor

Figure 6 shows the interface for the multiple-representations version of the tutor. In this version, students are given a problem to solve—here, $4(x + 1) + 2(x + 1) = -6$—but instead of solving it by typing in successive algebraic transformations, they manipulate an analogous boxes-and-weights representation. The boxes-and-weights representation is shown in the left window, and corresponding equational representations are shown in the right window.

The primitive objects in the concrete model are weights, boxes, and a balance scale. A positive unit weight is a mass of single unit size (according to some arbitrary linear scale). All unit weights are identical. Each positive unit weight is designated by +. A positive unit weight denotes the numeral 1 or +1. A negative unit weight is a mass of single unit size whose weight is the inverse of a positive unit weight on our arbitrary linear scale. Intuitively, a negative weight is "lighter than air"; when combined with a positive unit weight, it would yield a compound with exactly 0 weight. Each negative unit weight is designated by −. A negative unit weight denotes the numeral −1.

[4]We use the phrases "the traditional notation" or "the traditional representation" to refer to the way mathematics is expressed in mathematics textbooks and, more generally, to describe the idea that doing mathematics is equivalent to manipulating expressions stated in this syntax.

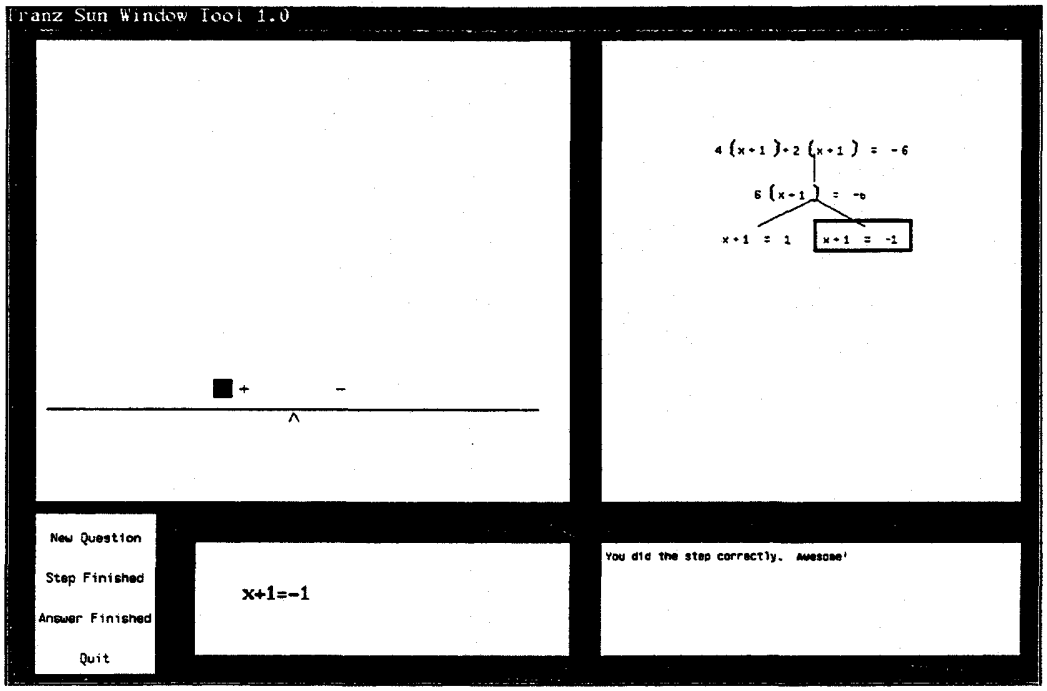


Fig. 6—The multiple-representations tutor

A black box is a weightless container with a fixed number of weights, *K*. For any given problem, a black box either contains all positive weights or all negative weights. We refer to this as the black box's valence. A white box is a weightless container that must contain the same number of unit weights as a black box. For any given problem, a white box must have a valence opposite that of a black box. If a black box contains only positive unit weights, a white box must contain negative unit weights. A black box denotes a variable (for example, *x*).

When a student is given a problem, the initial configuration of boxes and weights is "balanced," indicating a true algebraic statement. The student's goal is to deduce the number of unit weights in a black box—that is, a black box's valence. Students must arrive at their answers by transforming the given problem configuration into a final configuration in which one black box is on one side of the balance and only unit weights are on the other side.

Configurations are transformed using a mouse and menu that enable students to add and delete boxes or weights from either side. When students think they have made a balance-preserving transformation, they click the menu item *Step Finished*. The tutor underlying the interface will then translate the configuration of boxes and weights to an equational representation and display the equation on the window on the right, determine whether the manipulation represents a valid deduction, give textual feedback about the step's validity in the lower right window, and graphically unbalance the configuration if the step is invalid. Figure 7 shows the tutor's response to an invalid student transformation. (A more detailed account of the multiple-representations version of the tutor appears in McArthur et al., 1988.)

Students can play several different roles in this learning environment. In the version shown in the figures, the student manipulates the concrete boxes-and-weights representation

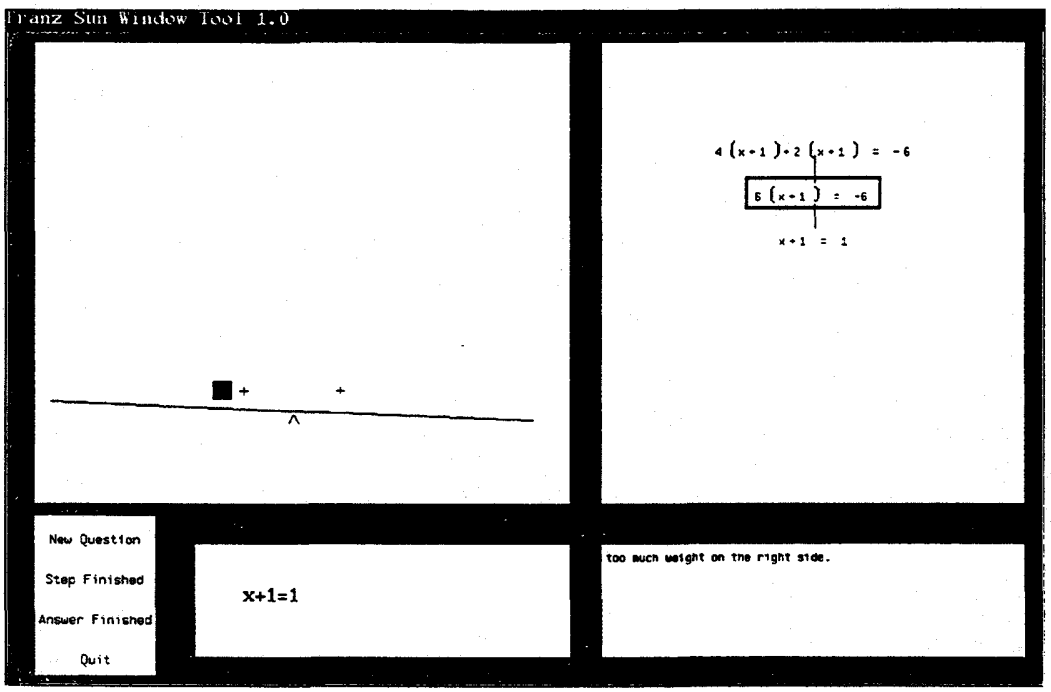


Fig. 7—Response to an incorrect step in the multiple-representations tutor

and observes the effect on the corresponding equations. In another version, the student effects the equation transformations, and boxes-and-weights manipulations occur automatically. In a third version, the student merely watches as the tutor solves the problem in both representations.

Principles Derived from the Multiple-Representations Tutor

The boxes-and-weights notation has certain limitations in its expressive capacity. It is applicable only to a well-defined subset of basic algebra equation solving and therefore cannot be thought of as a possible replacement for the traditional representation for algebra. However, viewed as one of several representational systems in a heterogeneous package, it has several potential pedagogical advantages. Most important, it is a concrete representation. Symbols in the language, and combinations of symbols, have natural interpretations in terms of physical objects and forces with which the student is already familiar. The students can use their existing intuitions about weights and balances to guide their interpretations of given configurations. In addition, the same physical intuitions can guide students in the process of problem solution. To assess the validity of a step, students can appeal to their understanding of how balance changes with the addition, deletion, or rearranging of weights. For example, it should be obvious to the student that any manipulation that simply rearranges the objects on a single side of the balance will not cause it to become unbalanced. Corresponding transformations of equations (for example, the distributive law of multiplication over addition) are not at all obvious to many students.

To summarize this version, we list below several points that are general principles concerning the use of multiple representations in tutorial environments:

1. *Use representations with diverse properties.* The traditional notation of mathematics is so dominant that we often regard it as synonymous with mathematics. But just as many possible computer languages for representing algorithms exist, many representations of mathematics can exist. How do we decide which alternative representations to explore? Our principle has been to look for representations that complement the traditional one. Of interest will be notations that are strong where the traditional notation is weak, from the student's point of view. The alternative notations need not be able to accomplish easily some things the traditional representation facilitates. From this point of view, the boxes-and-weights notation is strong in its concreteness and connection to physical intuition. Elsewhere (McArthur et al., 1988) we describe these properties in more detail and from a more theoretical perspective.
2. *Connect multiple representations dynamically.* We believe that using multiple representations to facilitate the process of mathematical reasoning, not just the comprehension of mathematical expressions, is important. In the sense of Lesh, Behr, and Post (1987) and Dienes (1960), the representation should be dynamic. It should not be merely another way of stating static algebraic equations; it must be an alternative way of reasoning mathematically. Students should be able not only to translate structures of one representation into structures in the other one, but also to transform structures within either representation to arrive at solutions—and possibly to help them understand reasoning in the other representation. In addition, the tutoring environment should be able to perform both translation and transformation, as well as to facilitate these operations by the student. For example, the multiple-representations version can both reason within the boxes-and-weights notation and translate to and from either notation.
3. *Allow the student to focus on selected aspects of reasoning with multiple representations.* Even in an environment as simple as our multiple-representations tutor, the student can engage in many different cognitive activities. The student can effect transformations in either notation, translate one notation to another, or watch the tutor translate or transform. Providing all these activities and structuring students' experience so that they are able to focus on each of these activities separately is important. The different variants of the multiple-representations tutor accomplish this goal.

THE GOAL-COMMANDS TUTOR

The passive version of the tutor required students to effect all algebra symbol manipulation. In the multiple-representations version, symbol-manipulation reasoning can be intermixed with reasoning in an alternative representation. In contrast, the goal-commands version of the tutor permits students to solve algebra problems using a set of commands, freeing them from the details of symbol manipulation.

Tutorial Goals of the Goal-Commands Tutor

We constructed the goal-commands tutor for two reasons. First, as with the passive tutor, we wanted to provide a tool that would help students learn higher-level problem-solving

skills (see Table 1). In contrast to our technique with the passive tutor, our technique here was to focus on higher-level skills by providing an environment in which all lower-level manipulations were automated. Second, we were interested in whether students, freed of the "clerical" duties of symbol manipulation, could speed through the learning of linear equations faster than they usually could in the classroom, or if they would progress further than the first-year algebra curriculum typically permitted.

Implementation of the Goal-Commands Tutor

Figure 8 shows one variant of the goal-commands tutor. In this version, students solve typical problems, but instead of manipulating equations or boxes, they issue commands that the tutor uses as instructions to effect the actual algebraic symbol manipulation. For example, in Fig. 8, the student has issued commands to add $+2x$ to both sides, simplify, subtract 8 from both sides, and divide both sides by 5. Commands are recorded in the command window at the right. Commands done by the tutor (for example, in response to a succession of *Help Next Step* requests) are shown in inverse video. As in other versions of the tutor, the student is free

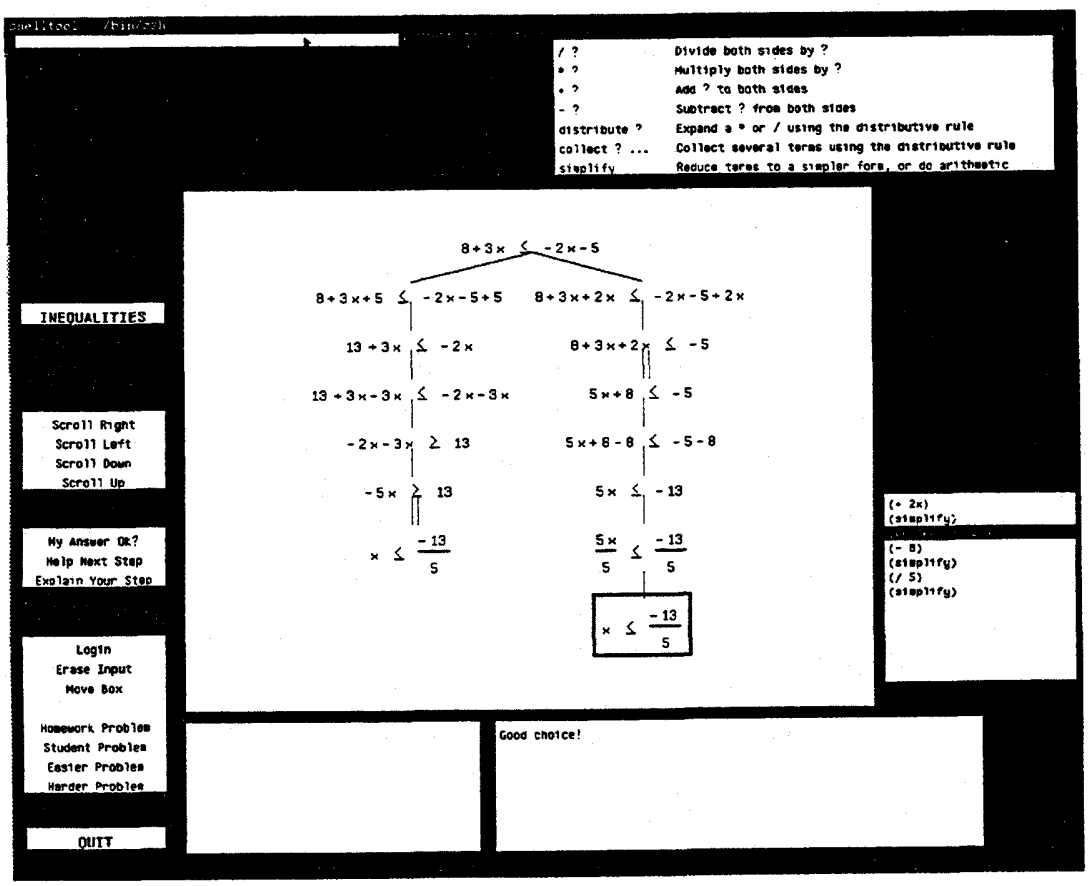


Fig. 8—The goal-commands tutor using lower-level commands

to explore alternative lines of reasoning. When the student moves from one line to another, the tutor simply restores the commands that generated the new line to the command window. In Fig. 8, for example, the commands in the command window are those that generated the current (right) path in the display window. If the student used *Move Box* to change the current focus back to the left path, the command window would be updated to show the commands that generated that line.

We have implemented several variants of the commands tool, each offering the students different levels of commands or imposing different tasks on them. Figure 9, for example, shows a variant with a different command set. Here students choose from a much higher-level set of options. The commands in this version do not describe the desired symbolic transformation of the current expression, but rather mention the *strategic goal* the student wants achieved by the next step. The tutor takes the responsibility of interpreting this goal in terms of a lower-level symbolic transformation. Permitting students to focus on very high-level decision-making, while ignoring the details of symbol manipulation, results in some exciting learning possibilities for students. For example, we have seen some students speed through the whole first-year algebra curriculum for linear equations in a matter of hours, rapidly acquiring a general feeling for the kinds of skills they need, while leaving the practice of detailed symbol manipulation skills until later.

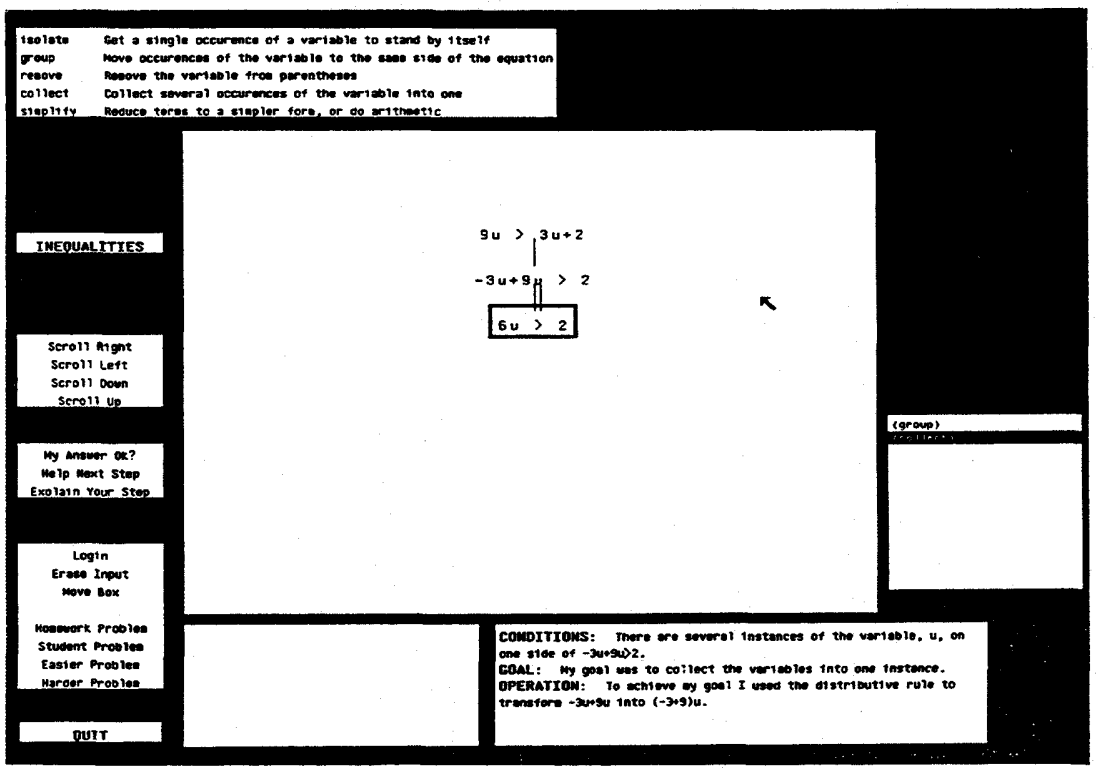


Fig. 9—The goal-commands tutor using higher-level commands

We are currently expanding the goal-commands tutor in several ways. First, we will permit students to name sequences of commands, put them into their library of commands, and reuse them at a later date. This will be useful when the student observes that a set of goals usually follows one after another. By defining new commands, students essentially construct their own highly simplified algebra problem-solving programs. Once the programs are constructed, students can test and refine such routines by executing the new command on different problems to observe its effects. Moreover, because different programs are permanently stored, students will be able to share their results with other students.

Second, in addition to saving sequences of generic commands as algebra programs, we propose permitting students to save the sequences of instantiated commands they issue in solving a particular problem. These problem-solving traces may have many uses. For example, the student (or another student) may invoke them at a later time to see an *abstracted replay* (Collins, Brown, and Newman, 1989) of how the problem was solved. Such abstracted replays may be much more tutorially effective than simply seeing the concrete symbol manipulations used in solving a problem.

Third, we propose varying the environment's tutorial intent by manipulating the availability of the math commands. For example, higher-level commands may be suppressed if we want the student to develop his own explicit command set. As another example, the student and tutor could reverse roles, with the tutor specifying the high-level commands while the student does the implied symbol manipulation. A variety of different policies for sharing the problem solving between student and computer will be available.

Principles Derived from the Goal-Commands Tutor

The goal-commands tutor and its extensions exemplify several important tutoring principles:

1. *Design teaching environments that permit novel sequencing of skills in mathematics curricula.* Algebra is typically taught bottom-up. Students are usually first taught basic properties (axioms), such as the distributive rule (or multiplication over addition), then simple procedures for applying such laws, then (possibly) local heuristics for problem solving. A much more top-down approach can be considered using the goal-commands tool, because the tutor shares problem solving with the student. For example, students can first acquire an overall understanding of equation solving by watching the tutor solve problems and by solving equations themselves using only very high-level commands. The potential advantages of a more top-down approach to teaching cognitive skills are apparent in many contexts. Fey (1984) has suggested that a similar reordering might be possible in a curriculum organized around the notion of a function. Collins, Brown, and Newman (1989) note that sequencing more general tasks before specific ones might be a general principle of "cognitive apprenticeship."
2. *Permit different knowledge levels of complex cognitive skills to be learned independently.* Because the goal-commands tool can allow students to play different roles, students can learn different levels of knowledge—understanding algebra facts concerning symbol manipulation and heuristics for applying them—independently of one another. Although students eventually need to practice both skills in combination, for the student to practice each component skill separately when *initially* learning algebra might be advantageous. This technique can be implemented in a particularly

powerful way in ICAI systems. In general, although learning skills in isolation can be *locally* effective, doing so may lead to global incompetence—the student may not be able to put together the well-learned pieces to solve complex, multistep problems. Intelligent tutoring systems can be configured to provide all the virtues of learning in isolation while avoiding the vices. As the goal-commands tutor illustrates, although the student focuses on practicing one knowledge level at a time, he or she does so *situated in a problem-solving context* (Collins, Brown, and Newman, 1989). When a student makes a decision at a given level, the tutor adopts all additional decisionmaking roles necessary to implement a solution. Thus, the student both focuses on a specific knowledge level in isolation and sees how it fits together with other levels.

3. *Provide environments for constructive learning.* Many ICAI programs use their intelligence to take much initiative away from the student, presumably on the assumption that an intelligent tutor is in a better position than the student to know what to do and when to do it. Balancing the development of computer-dominant tutors with the investigation of tools that promote student-controlled constructive learning is important. The extensions of the goal-commands tutor give simple examples of such tools. Ideally, we envision an environment in which the student can begin with *no* previous instruction on the solution of linear equations and build his or her own commands that are adequate to solving a wide range of problems. A key challenge in developing such tools will be designing aids that constrain the student's constructive exploration to a manageable scale. In our tutor, two components provide this assistance. First, the base commands with which the student would begin exploration (for example, doing the same operation to both sides of an equation, distributing) constrain the students to consider a subspace of reasonable, mathematically valid operations. Second, lesson control (see Fig. 1) can sequence questions from simpler to more difficult, so the student is essentially exploring a space whose size and complexity increase as his or her ability to explore it does. These are just simple examples of an idea we believe is of general importance in the development of novel ICAI systems: providing computer-based *guides* for managing students' explorations of learning spaces.

IV. EVALUATION AND ANALYSIS OF THE PASSIVE TUTOR AND ITS USES

In this section, we discuss our overall approach to evaluating the algebra tutor versions and our findings about the passive tutor's effectiveness. Specifically, we discuss the assessment instruments and procedures we have devised to date, present some preliminary outcomes from our first attempt to use the tutor in a school setting, describe students' evaluations of the tutor, and assess the passive tutor's limitations (which we later modified).

The evaluator seeking to determine student outcomes from ICAI systems has a wide range of alternative designs from which to choose (for example, Cook and Campbell, 1979). Which choice is best depends on several factors, such as the types of decisions (and decision-makers) on which the evaluation focuses, and the feasibility of implementing the design (Shavelson et al., 1986).

Generally speaking, two questions are raised in most evaluations: How much do students gain in knowledge, skills, and attitudes from the instructional intervention? Is what they learn the same as what students learn in traditional courses that are intended to teach the same subject matter (Shavelson et al., 1986)? As developers of an ICAI system, we weigh the question of knowledge gain as being more important than the question of equivalence of outcomes. This focus seems appropriate for this early stage of our research. Because we develop our software incrementally in a design-and-test cycle, our preliminary testing's first goal is to inform the (re)design of versions of the tutor.

Second, this preliminary evaluation is aimed at developing student outcome measures to be used in later phases of the project. As we debug or fix problems in each version of the tutor, we eventually reach a steady state—a version that operates as it was intended, that is comprehensible to students, and that embodies the intended instructional goals. At that point, we would also be interested in the second evaluation question. Measures developed and, in a sense, piloted in this early phase of the project can be tried and revised in parallel with system development. This process is also important because of the nature of our instructional software. Our emphasis is on teaching higher-level problem-solving skills rather than on rote symbol manipulation. Because standardized mathematics tests in wide use in schools focus on measuring the latter, we need to develop our own tests of higher-order skills.

In addition to questions about student outcomes, we are interested in the issue of implementation: What does it take to implement ICAI in schools? Our own and others' research on implementing instructional programs or new technologies indicates that the implementation process can strongly influence a program's success or failure (see, for example, Shavelson et al., 1986; Newman, 1989). Thus, we seek to understand both the implementation opportunities and the barriers present in the school setting.

In sum, the evaluation for the first implementation of the algebra tutor is more formative than summative. The aim is to pilot test one prototype system and our data-gathering instruments and procedures. The data from this study are useful for planning subsequent evaluations, making needed changes in the tutoring system, and learning more about how an intelligent tutoring system can best be used in typical mathematics classrooms.

MULTIPLE METHODS FOR ASSESSMENT

We developed various evaluation procedures and methods. First, we designed algebra achievement tests and attitude and background questionnaires to be administered before and after students used the tutor in the classroom. In addition, we conducted systematic observations of the tutor as it was used in the classroom. Finally, we gathered student-tutor interaction data and are developing tools for analyzing these data to assess student outcomes further. We discuss each of these measures and methods below. Like the software, all these measures are in early stages of development and will likely change as the project continues.

Achievement Tests

We designed our achievement tests to tap several kinds of skills, from "traditional" problems—for example, solve for x: $7(x + 9x + 4) = 12$—to higher-order problem-solving skills. The latter include mathematical skills identified as important in the literature (for example, flexibility and reversibility; see Rachlin, Matsumoto, and Wasa, 1985) and two kinds of problem-solving skills supported by the passive tutor environment: debugging and self-diagnosis skills, and goal-directed reasoning skills. Such higher-order skills are not typically included in most basic algebra curricula. To develop an item pool, we examined several standardized mathematics tests (for example, the Orleans-Hanna test), teacher-made tests, and tests developed by other researchers studying mathematics (for example, Rachlin, Matsumoto, and Wasa, 1985). In consultation with two algebra teachers working with the project, we constructed a 19-item test that included traditional and higher-order questions and also represented linear equations, inequalities, and simultaneous equations across these types. Interestingly, the teachers were very pessimistic about their students' ability to answer any nontraditional questions; more generally, they had low expectations for student achievement. Table 2 shows examples of problems for tapping higher-order skills; the complete test appears in the appendix.

Attitude Questionnaire

The attitude and background questionnaire (see appendix) included items on basic student characteristics (for example, age, sex, grade in school), previous experience with algebra, expected grade, interest in learning algebra, and assessments about how difficult the student thought learning algebra would be. Six questions assessed the students' previous experience using computers. Previous experience is important for at least two reasons. First, if students have little experience, their performance may be confounded by the initial demands of learning to use the tutor or by the fears that may arise from inexperience with computers. Both can adversely affect learning. Second, if students have little experience, outcomes may be biased by a novelty effect (Clark, 1985). In this case, novelty itself can enhance learning, but this positive effect can diminish over time.

In addition to learning about students' experience, we are interested in students' opinions about learning with computers. We asked six questions, adapted from an earlier study of telecourse instruction (Shavelson et al., 1986), to assess initial student attitudes and any changes after using the algebra tutor. Finally, the postcourse questionnaire included items to assess students' reaction to the tutor in general, as well as their reactions to specific features of the software. These responses might suggest important areas for software redesign.

Table 2

SAMPLE PROBLEMS FROM THE ALGEBRA ACHIEVEMENT TEST

Type	Problem
Debugging	Hector made an error when he solved this problem for x. See if you can find the error and put a circle around it. Then try to solve the problem correctly. $x + 4 + 4 = 10 - 4$ $x + 8 = 6$ $x = 2$
Reversibility	We will give you the answer to a problem and ask you to write an equation. Write an equation in which the answer is $x = 2$. The equation should have at least one set of parentheses.
Flexibility	Try to solve each problem below. Then, if you can, try to solve the problem in a different way and mark the method that you like the best. Solve for x: $7x - 2 = 5x - 8$

Classroom Observation

To gather data about tutor implementation and to aid students working with the tutor, at least one project member was present in the classroom when the tutor was in use. Project members filed daily reports on their classroom observations, dividing observations into several categories: software bugs (an outright tutor error), software weaknesses (the software did not "break," but did not appear to support student learning when it had an opportunity to do so), software strengths (tutor appeared to support student learning successfully), and student use patterns ("bugs" in students' knowledge of algebra, nonoptimal sequences of operations, and so on). These observations help us determine whether the software was operating as we intended and assess the tutor's value in promoting specific problem-solving skills.

In addition, our presence allowed us to observe how the teachers conducted their lessons, managed their classes, and integrated the use of the tutor into everyday instructional activities. Our previous work indicated that even well-designed computer-based instruction can fail if teachers do not integrate it with their regular, ongoing instructional plans. Integration proves difficult even for exemplary teachers experienced in using computers for instruction (Shavelson et al., 1986). In addition, a new technology and the organization into which it is placed tend to adapt to each other. For example, new uses for a technology are identified that were not envisioned in the original design (Bikson and Eveland, 1986). Thus, initial pedagogical goals of computer software can change in the hands of students and teachers (Newman, 1989). Because few ICAI systems have ever been used in regular school settings, we have little information on how they should best be used and whether their uses might differ from other types of computer-assisted instruction. Observations provided data from which to speculate about these important issues.

Student Scripts

The tutor records each action the student takes in solving every problem with the tutor. The actions are time-stamped, yielding a very detailed script of all student decisions. These scripts have been saved to files, and we have now begun to develop methods to manage and analyze them. In addition to performing standard statistical analyses (for example, average time to work a problem), we also plan to subject these data to various nonstandard techniques. In particular, we are experimenting with the use of INGRES, a database management system normally used to query commercial data. This system represents a potentially valuable tool for our purposes because, unlike statistical packages, it permits us to analyze temporal patterns in students' decisionmaking.

PRELIMINARY ANALYSIS OF STUDENT OUTCOMES

Below, we present data gathered from our initial fielding of the passive version of the algebra tutor (as described in Sec. III) at a local high school. Our analysis focuses on results obtained from the achievement test, attitude questionnaire, and classroom observations. Because we are still experimenting with the methods for collecting and analyzing student scripts, we do not report here tutor-recorded data.

In addition, because this pilot study did not include a comparison group of students who did not use the tutor, this design does not provide direct evidence of the tutor's teaching effectiveness. As we discussed above, we did not intend to conduct a traditional summative evaluation of student learning with this first prototype version of the tutor. However, our study design does permit us to assess some statistical and theoretical properties of our achievement test, to examine the relationship between achievement and various student characteristics, to observe how the tutor was used in the classroom, and to suggest ways for improving the software. This initial study did not systematically address the broader issue of how ICAI can best be implemented in school settings. Tutor implementation is the subject of a subsequent paper (Robyn et al., 1989).

Study Sample

We installed six Sun Microsystems workstations in two classrooms at a local high school. Five first-year algebra classes, taught by two teachers, used the tutor for most of a semester. These teachers volunteered to participate in the study and were paid an honorarium for their participation. In addition to allowing the tutor and researchers to occupy their classrooms, the teachers had input with regard to the curriculum topics covered by the tutor, the achievement tests, the classroom implementation procedures, and so on.

Parental consent forms were sent home with students to obtain permission to participate in the study. Some 140 students of average or below-average algebra skill participated. Approximately 14 percent of these students were taking the course for the second time. Many students were upperclassmen who had elected to postpone fulfilling their algebra requirement for as long as possible. For these students, this first-year algebra class was the most advanced mathematics course they would take in high school. The teachers considered the students in these five classes to be at the lower end of the distribution of mathematical ability as compared to students in general. Thus, the tutor was used by a group of students with a clear need for the added instructional benefits it was intended to provide.

Training and Procedures

Project staff members gave an overview of the tutor and its use to each class, trained each student individually, and provided assistance whenever the computers were in use. Student instructional manuals were also made available to the students, either at home or in the classroom. Questionnaire responses indicate, however, that the manuals were not in wide use: only 11 percent of the students reported having read them.

Students used the tutor during regular class times, but total available class time and individual student use varied considerably. Time constraints were dictated largely by the needs of the teacher and his or her regular teaching plans. Typically, the teacher's major activity was lecturing (both reviewing and presenting new material), followed by seat work. Students were permitted to use the tutor only during seat-work time. Length of lecture time or time devoted to other activities (for example, calling the roll, managing the class) varied greatly, thus affecting the amount of time remaining for tutor activity. In one teacher's class, for example, tutor activity averaged only some ten minutes per class. In addition, lack of machines, relative to the number of students in the classroom, reduced time with the tutor: Approximately 25 students had to share three computers. Although students often worked together, there were neither enough machines to go around nor enough classroom time to use them very effectively.

Students completed the achievement test and attitude/background questionnaire described above at the beginning and end of the semester. Project staff members filed class reports daily. Both precourse and postcourse measures were completed by 80 students.

Differences between Student Groups

Our first task was to determine if the 80 students who completed both premeasures and postmeasures differed from those who completed only one of the two tests. Students were lost because of absenteeism, late entry into the class (for the pretest), or dropping out of the class (for the posttest). If the groups do not differ, we can have some basis for using change scores (posttest minus pretest score) for the 80 students completing both tests to estimate learning for all the students.

To assess possible group differences, we first compared two sets of achievement scores: pretest scores for students completing just the pretest with students completing both tests, and posttest scores for students completing just the posttest with students completing both tests. In each case, normal probability plots produced shapes that were indistinguishable for the two groups being compared, indicating that the students who took only one test did not differ from those who took both. Mean test score comparisons for the separate groups also showed no significant differences (see Table 3).

We also compared students taking both tests with other students on demographic characteristics (sex, age, grade in school), attitudes about algebra, previous mathematics grades and algebra courses, attitudes about learning with computers, and experience with computers. Again, we found no differences between the groups. Groups were different, however, with respect to whether learning graph equations helped them: Of students taking both tests, 46 percent felt the computer helped "a good bit" or "a great deal," while only 29 percent of other students felt the same way (chi square = 7.86, $p < .05$).

By and large, then, these analyses suggest that the student groups were quite similar. Because *changes* in student learning and attitudes are of primary concern, we examine them below for the subset of 80 students who completed both pretests and posttests.

Table 3

MEAN ACHIEVEMENT TEST SCORES FOR ALL GROUPS

Tests Students Took	N	Pretest	Posttest
Both tests	80	4.71 (2.64)	5.71 (3.04)
Pretest only	34	4.67 (2.71)	(a)
Posttest only	33	(a)	5.46 (2.57)

NOTES: N = sample size. Means appear first, with standard deviations in parentheses.
[a]Not applicable.

Student Characteristics and Attitudes

At pretest, all students ranged from 14 to 18 years of age (average = 15.8) and were in the 10th, 11th, and 12th grades (51.25 percent, 40 percent, and 8.75 percent, respectively). The sample was 54 percent male. The composition of students differed somewhat for the two teachers. One teacher had more boys and 10th graders, while the other had more girls and 11th and 12th graders. Approximately half the students received a grade of B or higher in their last mathematics course, and some three-fourths expected these grades in the current course. Although students significantly reduced expectations at posttest (only 55 percent expected an A or B), in actuality more than three-fourths obtained a grade of C or less. This suggests that these students' estimates of their algebra skills are highly inflated and that such estimates may be poor indicators of actual performance. Perhaps corresponding to these course grades, students' interest in algebra and feelings about algebra (as rated on four-point scales) significantly decreased over the course of the semester.

Overall, these students had extensive and varied previous experience with computers: 52 percent had taken a course about computers or computer programming, half had used computers in other classes, and approximately two-thirds (65.8 percent) had used them outside of school. Computer uses included doing homework (25 percent), programming (43 percent), and playing games (95 percent). Some 85 percent thought they would like learning algebra on a computer.

We asked students a series of six attitude questions about computers and learning with computers—questions derived from our previous work in evaluating telecourses (Shavelson et al., 1986). They rated on a six-point scale (1 = strongly disagree, 6 = strongly agree) statements such as: "I believe many algebra courses could be improved by the use of computers" or "Computers are poor substitutes for algebra teachers." Although on average students held positive attitudes toward learning with computers, these positive attitudes consistently declined from pretest to posttest for each item (over all items, $F = 7.67$, $p < .0001$). We observed the same decline in attitudes in our telecourse evaluation. These changes may reflect similar declines in liking algebra, in having an interest in algebra, and in course grades. Except where noted, no significant differences existed in the characteristics and attitudes of students across the five classes or two teachers.

Student Learning

Analysis of data from the 80 students who took the achievement test before and after the course indicates that the test was very difficult. Although the difference was statistically

significant, students' average score increased from only 4.71 to 5.71, out of a possible score of 19 ($F = 5.13$, $p < .0001$). Analysis of variance tests revealed no significant differences between classes or teachers.

Item statistics for the achievement tests appear in Table 4. Item difficulty—the proportion of examinees who got that item correct—ranged from 0 (no one found the correct answer) to .9.

The data in Table 4 support the notion that items are arrayed along a difficulty continuum, from linear equations to inequalities to simultaneous equations. An examination of the set of debugging (1–3) and flexibility (10–13) items shows student performance following this progression. This did not appear to be the case with the word problems, however, since the simple linear case proved more difficult than the others. We see a similar progression in the two sets of items: 4 and 6, and 5 and 7. Within linear equations, the item with single parentheses (item 4) was easier than that with embedded parentheses (item 6). The simple inequality (item 5) was a little easier than the inequality with parentheses (item 7), particularly at posttest. Also interesting is the sharp increase in performance from pretest to posttest in items 3 and 13. Although these items were still difficult for most students at posttest, clearly some proportion of students had mastered simultaneous equations, which were taught during the semester.

The word problems (with the exception of item 15) and the reversibility problems also proved very difficult for these students. Word problems or simple story problems are notoriously difficult for students, as indicated by previous research and data from national surveys of mathematical problem solving (for example, Carpenter et al., 1980). None of these students correctly answered two of the reversibility items (18 and 19) on either test, although the

Table 4

ITEM STATISTICS FOR THE ACHIEVEMENT TEST

		Difficulty[a]		Discriminability[b]	
Item	Problem Type	Pretest	Posttest	Pretest	Posttest
1	Debugging, linear	.80	.92	.29	.41
2	Debugging, inequality	.39	.44	.61	.61
3	Debugging, simultaneous	.01	.16	.10	.31
4	Linear, single parentheses	.36	.51	.43	.39
5	Simple inequality	.26	.32	.66	.51
6	Linear, embedded parentheses	.25	.31	.30	.44
7	Inequality, parentheses	.24	.24	.57	.53
8	Simultaneous, easier	.02	.10	.17	.35
9	Simultaneous, harder	.01	.02	.10	.25
10	Flexibility, linear	.60	.65	.60	.61
11	Flexibility, linear w/paren.	.42	.50	.37	.44
12	Flexibility, inequality	.25	.19	.56	.53
13	Flexibility, simultaneous	.04	.25	.25	.46
14	Word, linear	.09	.10	.23	.34
15	Word, inequality	.49	.56	.29	.43
16	Word, simultaneous	.26	.20	.20	.31
17	Reversibility, linear	.21	.22	.52	.41
18	Reversibility, inequality	.00	.00	.36	.24
19	Reversibility, simultaneous	.00	.00	−.06	.21

[a]Proportion of students who got the item correct.
[b]Point-biserial correlation between item and total test score.

number of students who attempted to solve them doubled from pretest to posttest (for item 18, 17–41 percent attempted them; for item 19, 12–21 percent attempted them).

In sum, although students showed a statistically significant improvement in achievement, the increase was of little *practical* significance. Clearly these students not only had little algebra skill when they began, but by the end of the semester could only complete approximately one-fourth of the test items correctly. Note that students apparently did more poorly overall on the more "traditional" items (4–16) than on some of the nontraditional debugging and flexibility items. These results are encouraging for future test development, because only nontraditional items can measure the kinds of skill acquisition we hope the tutor can impart.

Correlates of Achievement

Correlations between selected student characteristics (ability, computer experience) and the algebra achievement posttest and change score appear in Table 5. A correlational analysis informs us of which factors are associated with end-of-course algebra achievement. Measures of initial "ability," including a student's pretest score, final grade in his or her previous math course, final grade in the current course, and estimates of algebra difficulty were all significantly correlated with posttest performance. Although 11 of the 80 students completing both tests had taken algebra before, previous experience was not related to posttest performance. Of all these characteristics, only the pretest score was significantly correlated with change in achievement: Students scoring higher at pretest had smaller change scores. These students, perhaps, were already "pushing their limit" of algebra learning. Previous computer experience was consistently (but weakly) correlated with lower posttest performance.

Table 5

SELECTED CORRELATIONS BETWEEN STUDENT CHARACTERISTICS (AT POSTTEST) AND POSTTEST AND CHANGE SCORES

	Posttest		Change Score	
Characteristics	N	r	N	r
Pretest	80	.62[a]	80	−.30[b]
Grade last math course	76	.29[b]	76	.04
Taken algebra before	80	.19	80	.10
Final grade	80	.46[a]	80	.14
Estimate: difficulty of algebra	80	.28[b]	80	−.06
Experience with computers:				
Taken course	80	−.16	80	−.22
Used at home	79	−.19	79	−.04
Used for homework	79	−.19	79	−.06
Played games	80	−.18	80	−.05
Programming	79	−.27[b]	79	−.02
Literacy	79	−.27[b]	80	−.16

NOTES: Literacy is a composite score created by summing over the experience items. N = number of subjects; r = correlation coefficient.

[a] $p < .01$.

[b] $p < .05$.

Student attitudes toward computers and learning algebra with computers (summarized as a single "opinion" score by first reversing negatively worded item responses in a positive direction, then adding responses to the six items) were significantly correlated with their experiences in using the tutor. For example, they found the tutor more enjoyable to use ($r = .60$), felt the tutor helped them to solve algebra problems and to graph equations ($r = .56$ and .40, respectively), found the tutor easy to learn to use ($r = .38$), and would use computer tutors again ($r = .45$). This suggests that students who were positively disposed toward learning with computers tended to enjoy their experience with the algebra tutor. However, students' positive attitudes and experiences with the tutor were negatively (although not significantly) related to test performance. Positive attitudes toward learning with computers, experiences with this algebra tutor, and previous computer experience appear unrelated to actual achievement in algebra. Perhaps this "nonfinding" is not a general cause for concern. If students with less computer experience or negative attitudes did poorly, we would need to worry about how to make the tutor more motivationally appealing or how to help students overcome negative feelings about the technology. On the other hand, student comfort with the technology, though perhaps necessary for learning, does not ensure that the technology will be effective as an instructional tool.

A final interesting pattern appeared with regard to feelings about algebra and performance. At pretest, girls had significantly lower expectations about their grades ($r = -.24$) and disliked algebra ($r = -.25$) more than boys. At posttest, girls still judged algebra to be more difficult than did boys ($r = -.28$). However, gender was basically unrelated to pretest, posttest, or change scores ($r = .07$, $-.03$, and $-.11$, respectively). Because these correlations suggest a possible interaction between gender, feelings about algebra, and performance, we conducted several analyses of variance tests to explore these patterns further. However, none showed any significant main effects (gender, algebra interest, feelings about algebra, difficuly of algebra) or interaction effects with respect to change or posttest scores.

Explaining Differences in Achievement

We used multiple regression to test the independent effects of student characteristics and prior ability. Using posttest and change scores as dependent variables, we specified models based on relationships revealed in the correlational analyses, as well as differences found in student characteristics and attitudes, or other variables that potentially affect learning outcomes (for example, grade in last math course). As a step toward developing these models, we performed several selected analysis of variance tests. These tests served two purposes. First, they identified any interaction effects that should be included in the regression model. For example, differences in gender and grade distribution between teachers and differences in gender and attitudes about algebra suggested possible interactions between teachers, gender, and attitudes. Significant interactions, or those where the mean square is larger than the mean square for the main effect, are clear candidates for inclusion in the models. Second, these tests were useful as a complement to the regression analysis, which can obscure interesting underlying patterns among variables. Because this study is formative and exploratory, we are also interested in uncovering these patterns. That said, these tests did not reveal any significant main effects or interactions. Several interaction terms, however, had mean squares large enough to be interesting. We include these in the models below.

The multiple regression equation predicting the posttest score appears in Table 6. This model explains a significant though modest portion of the variance in posttest score. Most

Table 6

REGRESSION RESULTS FOR THE POSTTEST SCORE

Variable	b	t
Constant	1.03	.38
Pretest score	.70	5.59[a]
Teacher	.36	.11
Gender	1.40	1.07
Interest in algebra	−.16	−.18
Feelings about algebra	1.02	.96
Easiness of algebra	1.92	1.92[b]
Grade 1	.51	.20
Grade 2	−.80	−.31
Teacher/gender interaction	−1.11	−.87
Teacher/grade 1 interaction	−1.10	−.38
Teacher/grade 2 interaction	1.64	.57
Teacher/feelings interaction	.21	.89
Teacher/easiness interaction	−2.10	.12
Gender/easiness interaction	−1.79	.17
Gender/feelings interaction	−.68	.65

NOTES: Adjusted R-square = .36; F = 3.98[a]; N = 80. With the exception of pretest score, most predictor variables are dummy variables, with the absence of the variable as referents.

[a] $p < .0001$.

[b] $p < .06$.

predictive is the pretest score: Students scoring higher on the pretest also score higher on the posttest. Students who feel algebra is "easy" also receive higher posttest scores, although this effect only approaches significance. None of the other factors in the model predict end-of-course algebra achievement.

Using the same model, we sought to determine the relative importance of the same factors on students' change score. Results of this model (see Table 7) were nearly identical with that reported above. Simply stated, the first model says that posttest score is composed of a constant plus .70 times the pretest score, plus other factors. The second says that change score is composed of a constant *minus* .29 times the pretest score, plus other factors. Together they say that students scoring high on the pretest score higher on the posttest, but that students with higher algebra knowledge to begin with did not increase their score as much relative to students with less knowledge. The second model explains little of the variation in change score.

The preliminary data reported here tell us little about the relationship between student characteristics, overall achievement, and the nuances of learning algebra with an intelligent tutoring system. For example, we have not yet determined how these variables relate to time using the computer, time to solve different types of problems, frequency of use of the different tutor options, and the students' success or failure on individual component skills that comprise each problem the student attempted. Methods for automatically gathering these and other data were in development in the first year of the tutor implementation; the data gathered were not complete enough to conduct finer-grained analyses that might shed some light on these and other questions. These methods will be in place for future implementations and thus increase

our ability to understand how the tutor promotes student learning. In addition, future evaluation designs will include appropriate comparison groups in order to examine the tutor's teaching effectiveness.

STUDENT EVALUATION OF THE TUTOR

At posttest, 98 percent of the students taking the test (from a sample of 119 students) reported having used the tutor. Two-thirds of them reported enjoying the tutor "a good bit" or "a great deal." Some 85 percent thought the tutor fairly or very easy to learn to use. Students were less enthusiastic, however, in their assessment of whether it helped them learn to solve algebra problems or graph equations: Approximately 35 percent thought it helped "a good bit" or "a great deal." Only 22 percent thought the tutor useful for "learning about computers." Despite the somewhat disappointing assessment of the tutor's usefulness for learning, 60 percent said they would use computer tutors again if given the opportunity. Given the global nature of this assessment, determining the precise source of the students' disappointment with the tutor's teaching capabilities is difficult. It might result from specific aspects of the tutor or from the students' overall class experiences during the semester.

Looking at average student ratings of specific tutor options in Table 8, we see that students found some options more useful than others. Options that provided precise feedback on the correctness of a step or answer were judged most helpful. Judged least helpful was the option that allowed the student to obtain the tutor's solution for any step in the reasoning tree. A student might choose this option, for example, if he or she could not figure out an error in a step and wanted to see the tutor's solution. *Elaborate Step* also received a lower rating; this option shows the tutor's more detailed reasoning from one step to another. These results suggest that students want immediate, specific feedback. This desire is understandable since

Table 7

REGRESSION RESULTS FOR THE CHANGE SCORE

Variable	b	t
Constant	.53	.19
Pretest	−.29	−2.35[a]
Teacher	.36	−.11
Gender	1.39	1.07
Interest in algebra	−.16	−.18
Feelings about algebra	1.01	.96
Easiness of algebra	1.92	1.92
Grade 1	.51	.20
Grade 2	−.81	−.30
Teacher/gender interaction	−1.11	−.87
Teacher/grade 1 interaction	−1.10	−.38
Teacher/grade 2 interaction	1.64	.57
Teacher/feelings interaction	.21	.14
Teacher/easiness interaction	−2.11	−1.56
Gender/easiness interaction	−1.79	−1.40
Gender/feelings interaction	−.68	−.46

NOTES: Adjusted R-square = .05; F = 1.30; N = 80. Predictor variables are dummy variables, with the absence of the variable as referents.

[a] $p < .02$.

Table 8

STUDENT EVALUATIONS OF THE TUTOR HELP OPTIONS

Option	Number Using	Percentage Reporting That the Tutor Helped "a Good Bit" or "a Great Deal"	Average Rating
Answer Ok?	111	76.57	3.09
Step Ok?	109	75.23	3.10
Hint Next Step	96	59.37	2.68
Elaborate Step	81	50.62	2.59
Explain Step	95	64.21	2.87
Do Next Step	88	65.91	2.92
Do Some Step	71	46.49	2.55

NOTE: Rating based on a scale from 1 ("no help") to 4 ("helped a great deal").

students rarely get such feedback of their problem solving in the classroom or when doing homework. Students commented that they liked the *Step Ok?* option in particular because it allowed them to check their work as they went along. Students often experience frustration when they work through a problem in class and discover that their answer is wrong. The *Step Ok?* option permits them to detect an error right away and to try to recover.

Our classroom observations suggest that students may find less usefulness in tutor-generated elaborations of explanations because they do not know how to use that information to aid in their problem solving. Collins and Brown (1987) posit that intelligent tutoring systems can become a powerful tool for learning through reflection. We adopted this principle in the design of the passive tutor because students can compare the details and structure of their own performance with that of the expert system, thereby discovering elements that need improving. Our data suggest, however, that expert help may be at a student's disposal, but this availability does not guarantee that he or she can or will make use of it. Rather, students may require explicit training to use the tutor as a tool for learning through reflection. Designing and providing such training should be a goal of our future research.

THE TUTOR'S LIMITATIONS

To determine why the tutor did not more significantly improve students' algebra performance, we analyzed our recorded classroom observations and found that two main problems apparently foiled students' effective use of the tutor. The first was the tutor's limitation as an aid for learning specific pieces of knowledge needed to solve symbolic algebra problems. The second problem was the tutor's inability to improve the students' model of math.

The Tutor's Limitations as a Learning Tool

Many of our observations suggest that the passive tutor is a less effective learning tool for students who are weak in algebra than for those who already have an intermediate knowledge. Students who had achieved a relatively high level of competence in target skills before encountering the tutor were able to recover from their slips using the tutor's highly specific error feedback and to correct minor misconceptions by exploiting tutor explanations. However,

the tutor's coaching appeared to be insufficient in helping students who had not achieved a high degree of competence on the necessary algebra skills before using the tutor.

The reasons for the tutor's shallowness as a learning tool appear to include its

- *Lack of error explanations.* The tutor did not explain why the student made a mistake; he or she had to infer the reason. Instead, the tutor provided hints about what to do or explained examples of what the expert would do. Even these aids, however, were not invoked automatically and had their own imperfections.
- *Awkward wording of hints and explanations.* To help students infer the correct skill when theirs was inadequate, the tutor provided hints and explained ways of allowing the student to "get inside" the expert's head. Although this procedure may have worked to a limited extent, its effect was blunted by the unclear wording of many hints and explanations.
- *Lack of interactivity at the microscopic level of reasoning.* Although the tutor provides much richer feedback than most CAI programs, it cannot yet provide the support that teachers can in coaching students through very difficult reasoning steps. Our tutor waits for students to make an algebraic transformation and then comments on it. A good teacher can often help break out the separate reasoning steps before a visible transformation is done; this type of help may be necessary when students' understanding of problem-solving skills is formative.
- *Passivity.* In the initial version of the tutor, the student had to request assistance. Perhaps when students are focusing on acquiring new math skills they should not also have to focus on metacognitive skills necessary to control the tutorial interaction.[1]

The Tutor's Failure to Improve the Students' Model of Math

Our observations also suggest that in addition to having deficiencies in specific knowledge of algebra concepts, the students also had a deficient overall model of algebra. Students tended to have basically a cookbook model of math. They believed, for example, that if a problem could not be solved by using one of the known rules, it could not be solved at all; they viewed one procedure or axiom as being unrelated to others they had learned; and they believed that the problems they solved with procedural rules had nothing to do with problems in the real world. Unfortunately, our current tutor version does nothing to impart a more comprehensive model.

The ultimate goal of an algebra tutor should be to demonstrate to students that symbol manipulation is part of a deductive system and, along with the whole deductive system, can be regarded as a tool for representing models of natural situations and making (formalized) inferences about such situations. If students were to realize that symbolic reasoning is done in a deductive system, they would understand that the theorems they learn are logically connected; this understanding, in turn, would allow them to reason better about symbol manipulation techniques. For example, they could verify answers to questions by checking the logical consistency of the answer or by using another deductive method.

In addition to our current tutor doing little to replace the cookbook model students now possess, the software in use at present largely reinforces the current curriculum's weaknesses.

[1]Other possible explanations are noncognitive in nature, such as the difficulty in operating the computer interface or motivational problems. However, we found little evidence for the first; students could easily manipulate the interface within a few minutes of instruction. We have anecdotal evidence that motivation may have been a problem. Of course, low motivation is likely tied to the cognitive deficiencies we mentioned above.

It helps students efficiently practice techniques for algebraic symbol manipulation without getting them to see what the techniques mean, where they come from, or how they relate to making inferences about real-world problems.

Several aspects of the software are at fault:

- The lesson control facility permits the student to solve the same kinds of symbol manipulation problem indefinitely, so the student never comes to terms with the problem of when to use which technique or procedural rule.
- The tutor sticks largely to one technique for solving a problem instead of showing alternate methods.
- Although the tutor provides feedback about the student's symbolic reasoning in solving a problem, it does not assist in understanding the justification behind a reasoning technique or procedural rule the student might use. Nor does it help the student to break down the rule deductively and discover its soundness or to see its relationship to other rules.
- The tutor provides no semantic basis for algebraic symbol manipulation. It poses purely symbolic problems, expects symbolically reasoned answers, and never uses natural situations as the basis for symbolic problems.

CHANGES TO THE CURRENT TUTOR

Based on our analysis of the preceding problems, we implemented various changes and additions to the tutoring software. In this section, we present changes intended to make the tutor a more effective learning tool for lower-performing students; the following section then discusses the issue of improving students' model of algebra.

The individual changes we implemented attempt to improve the tutor by:

- *Making the interface easier to deal with.* The assumption here is that many novice students did not learn rapidly with the tutor because they spent too much effort struggling with the interface.
- *Providing more comprehensible and detailed feedback when the student makes a mistake or requests assistance.* The assumption is that students may have failed to progress because they did not receive enough information at the right time to overcome their misconceptions.
- *Taking the initiative out of the students' hands (when necessary) to allow them to focus on learning one concept or accomplishing one task.* The assumption behind this point is that a passive tutor, by requiring students to do everything themselves, places too many cognitive demands on the students at one time. Consequently, novice students may be trying to learn too many other skills that detract from learning algebra.

An example of a specific change we effected in later versions of the tutor was to automate purely arithmetic steps because students make many arithmetic errors, which take time to correct and contribute nothing to the learning of algebra skills. Such automation should be a reasoned tutorial decision, however, and should not be done on an unlimited basis. Likewise, we automated simple nonarithmetic steps because we discovered that students spend considerable time doing necessary but superficial steps. Another change was to allow students to generate their own questions. The rationale for this change was that it permits greater student involvement and allows students to focus on a particular issue. This may also increase

students' motivation to use the tutor since they can enter and solve their homework problems. A fourth example of a change was improving the problem solver so that its reasoning was more intelligible. Previously, for instance, it could not demonstrate how or when to divide both sides by a value.

In addition to these changes, we also made numerous interface changes, which are modifications in the way the student and tutor interact to exchange information. These changes were to the menus, display, text, input, and recording of student data. Many of these changes are minor in nature, as demonstrated by the following illustrative examples. One menu change was to eliminate the *Quit* option because students can abuse this choice and can effect quitting in other ways. Another menu change was to add *Explain Your Last Step* because students want to see an explanation of the tutor's step right after it has been done. An example of a change in the text was making the explanation and hint text clearer and shorter. And an example of a change to the input was making it easier to correct a typo in the work space; this allowed students to do transformations faster and to learn more efficiently.

ADDITIONS TO THE TUTOR

Changes to the tutor that maintain its current purpose will not be able to address the second general major shortcoming: deficiencies in students' models of math. The current tutor is clearly limited to helping students acquire skills at solving symbolic algebra problems, and the changes outlined above are aimed at improving it as a tool for teaching such formal reasoning. Nevertheless, we have made one addition that should contribute to a better understanding of algebra's deductive nature: permitting a controlled discovery of procedural rules. In other words, instead of giving students a procedure to follow, the tutor makes them synthesize it. For example, to construct a procedural rule for solving equations of the form $ax + b = c$, the student would need to discover a series of operations (subtract, simplify, divide by a, simplify) that would transform the equation to one of the form $x = d$. That way, students will see how the techniques they use to solve problems arise and how they are connected to other things they have learned. Another valuable idea we have not implemented is having menu options that allow the student to ask the tutor to explain a reasoning step instead of a procedure.

Neither the additions suggested above nor the changes outlined earlier situate algebra in the real world or establish it as a tool for representing natural situations and making inferences. A demonstration that algebraic statements can represent formal models of natural situations and that deductive reasoning has use in making important real-world decisions will mean mapping the formal symbolism of algebra onto real situations. In short, we must provide an intuitive semantics for algebra.

V. CONCLUSIONS AND FUTURE DIRECTIONS

We conclude this report with our view of where we believe ICAI *should* and *should not* direct its future efforts. The view here is selective, for we do not attempt to enumerate all promising future directions. Moreover, our view may not be shared by all researchers in the field. It is rooted in our central goal of developing computer environments—relatively passive tools, as well as more active tutors—to support students' learning of higher-level mathematical problem-solving skills.

OVERCOMING COMPETENCE LIMITATIONS

Great strides have been made and are being made in intelligent computer-assisted instruction. Many systems embed relatively complete representations of subject matter knowledge (for example, Brown, Burton, and deKleer, 1982; Clancey, 1979, 1983) or have well-developed skills for student diagnosis (for example, Sleeman and Smith, 1981). However, ICAI systems still fall short of human tutoring skills, and unless these shortcomings are rectified, ICAI systems will not achieve the two-sigma improvements in performance attributed to human tutors (Bloom, 1984). The main area in which ICAI systems are inadequate, as Ohlsson (1986) points out, is pedagogical expertise: Their knowledge of teaching per se is generally very limited, at both the tactical and strategic levels. Tactical knowledge refers to tutoring skills that are local in scope (for example, a decision to interrupt a student at a particular step while solving a single task), while strategic knowledge refers to more global understanding (for example, a decision to pose a certain sequence of tasks to a student).

In our future research, we will attempt to improve the tactical capabilities of our tutor by adding to its repertoire of techniques for dealing with local student difficulties. Enhancements to our lesson-updating module (see Sec. II) will provide strategic improvements. However, improving the pedagogical expertise of ICAI systems is a challenging and complex task. To approach human effectiveness, ICAI systems must represent much of the pedagogical knowledge that human tutors possess. Although cognitive psychology provides us with relatively well-defined information-processing models of *domain* expertise and *diagnostic* expertise, no corresponding literature on information-processing models of *pedagogical* expertise exists (for a review of some limitations of educational research in this regard, see McArthur, Stasz, and Zmuidzinas, in press; and Shavelson et al., 1989). Consequently, the implementation of sophisticated pedagogical expertise in ICAI tutors will likely be a slow process. The process will necessarily involve the *cooperative* effort of cognitive psychologists, who will carefully analyze the performance of expert tutors, and ICAI researchers, who will then formalize, implement, and perform computational experiments on the resulting information-processing models.

INVESTIGATING NEW FORMS OF LEARNING

Formalizing human pedagogical expertise is an important future direction for ICAI. We have begun our own research in this area, funded by National Science Foundation Grant MDR-8751104; for more details, see McArthur and Stasz (1987), and McArthur, Stasz, and Zmuidzinas (in press). However, we recognize that ours is not the only approach to developing

effective ICAI programs and may not be the most cost-effective method. Along with many ICAI researchers, we have hitherto taken one-to-one human tutoring—in which one student and one tutor work a series of problems—as the form of learning most desirable to automate. But we now wish to consider the possibility that instead of (or, in addition to) trying to optimize ICAI performance within this paradigm, considering other paradigms might be advantageous.

One-to-one tutoring is a powerful method for transmitting information between humans because it exploits many communications media, conventions, and devices people share because of their physical structure, learning, and culture. Among other things, it relies on gesture, natural language, informal sketches, and mutually understood conventions for communication. Many of these features are not now shared by computers (for instance, natural language) and perhaps never will be (for example, gesture, meaningful understanding of communication conventions arising from real-world commerce). Thus, although one-to-one tutoring may be a naturally powerful form of learning between humans, it may not be as effective between machines and humans.

The foregoing argument leads to an important conjecture about future directions for ICAI research: In addition to enhancing the computer's ability to engage in traditional forms of learning, we should seek *new forms of learning*—ones that take better advantage of ICAI's potential strengths as a communicator of knowledge. Several research groups have begun such investigations—most notably, Collins, Brown, and Newman (1989), who have considered importing an apprenticeship style of learning into a cognitive and computer-based context. In addition, exploratory learning environments such as STEAMER (Hollan, Hutchins, and Weitzman, 1984) and QUEST (White and Frederiksen, 1987) are beginning to define a new form of learning with computers that exploits some natural strengths of machines (for instance, the ability to simulate processes and present multileveled graphical representations of complex systems). The extensions to the goal-commands version of our tutor and the principles we offer in Sec. III also fall into this category.

FOCUS CHANGES

The possibility of developing fundamentally new forms of learning with computers suggests that they may radically alter the way we learn. In addition, it is important to consider that they may radically alter *what* we learn. As we noted in Sec. I, because computers can now automate much algebraic symbol manipulation, such skills are now less important for students to learn. To convert this insight into concrete ideas about the content of new computer-based mathematics courses requires careful consideration of three questions:

- Which mathematical skills are easily and effectively automated?
- Which other skills are important for students to acquire?
- For which of these other skills can effective computer-based learning environments be designed?

As with the analysis of pedagogical skills, these questions must be answered by a cooperative enterprise involving mathematics educators, professional mathematicians, and researchers in ICAI.

Our work has only scratched the surface of this problem. We are now concentrating on developing tutors and tools to help students learn higher-level mathematical problem-solving skills, but many other important aspects of mathematical thinking are untouched in ICAI.

One topic that can be learned particularly effectively with computers is mathematical modeling. Developing computer-based tools to support this topic is a main part of our follow-on research (funded under NSF Grant MDR-8751515). Mathematical modeling is a multifaceted skill. Students must learn how to gather data about some real-world process or situation, formulate and test mathematical models that fit the data, and use the models to make inferences and decisions about the real world. Our approach to learning such skills is to provide students with "microworlds" that can faithfully simulate real-world processes. We plan to supplement such microworlds with tools that enable students to gather and represent data (for example, tables of values, Cartesian graphs) and to record and then test equational models (for instance, automated symbol manipulation).

Implementing this new form of learning with computers has several potential advantages. In addition to focusing on nontraditional components of algebra competence, it places algebra in a natural context of use. This contrasts with many classroom environments in which symbol manipulation is taught as a stand-alone skill, without regard for its application to real-world problems (see Fey, 1984). Also, microworlds can represent a departure from the traditional drill-and-practice style of learning. Instead of working on a series of small, disconnected problems, students have an opportunity to exercise a variety of different skills as they work for an extended period of time on a set of meaningfully connected tasks.

INTEGRATION AND EVALUATION

When we first began this project, we envisioned developing an intelligent tutor for algebra that might change how algebra was taught. As the previous sections have illustrated, our expectations have slowly shifted: We now hope to affect both what is taught and the style of teaching. This shift has forced us to face a new collection of issues concerning the integration of software into a classroom setting and its evaluation in that setting. The preliminary analysis of our initial successes and failures in the classroom (Robyn et al., 1989) has reinforced the finding that new learning tools can be ineffective when placed in a context whose pedagogical goals are not consistent with the ones implicit in the tools.

We raise these issues here not because we have found a solution but because they are general problems that will be faced by any project attempting to import advanced technology into the classroom. In Sec. IV, we described some novel tools we are designing to help us evaluate our various tutoring versions and tools. However, the problems of integration and evaluation are sufficiently complex that they deserve to be researched as projects of their own rather than as adjuncts to the development of an ICAI system.[1] Recalling our previous recommendations, this suggests that an ideal research environment for developing effective ICAI systems would encompass several distinct but mutually reinforcing efforts:

- Development of information-processing models of the pedagogical skills in one-to-one tutoring, as well as models of other forms of instruction.
- Examination of alternative forms of learning using computers.
- Computer implementation of ICAI tutors and tools embodying models of human pedagogical skills and alternative forms of learning.

[1]In this connection, we have recently designed and pilot tested a "minicourse" based in part on the results of our initial study (Sec. IV). This course integrates our computer software into a complete curriculum for learning linear equations. A report on this study is under way.

- Design and testing of curricula and courses that integrate new ICAI systems with complementary noncomputer activities for students and teachers.
- Development of new measures for evaluating student performance in the courses and curricula.

Appendix

BACKGROUND QUESTIONNAIRE AND ACHIEVEMENT TEST

RAND

Algebra Tutor Project

RAND ALGEBRA TUTOR PROJECT: DIAGNOSTIC TEST

CONGRATULATIONS! Your class has been selected as one of five SAMOHI Algebra 1 classes to participate in a special project. A research team from The RAND Corporation wants you to work with them in the development of an intelligent computer tutor for algebra.

Very few students have the opportunity to use intelligent computers to help them learn algebra. SAMOHI will be one of the first schools in the country to have such sophisticated computers in their classrooms. The RAND research team is very excited about this project, but to be successful, we need your help. We hope that you will cooperate with us, and that you will be interested and enthusiastic about the project.

We'd like you to take this Diagnostic Test today. The first part is a questionnaire that asks about your background and your opinions about algebra and learning with computers. The second part is a test that will give us an idea about how much algebra you know already. If you have any questions while working on the test, please raise your hand.

THANKS FOR YOUR HELP!

RAND

Algebra Tutor Project

STUDENT I.D. _____

Student Questionnaire

DIRECTIONS: **Please fill in the parentheses next to each question with the appropriate number. Select only ONE answer to each question; please answer all questions.**

1. What is your sex? ( )

   (1) Male
   (2) Female

2. How old were you on your last birthday? ( )

3. What grade are you in? ( )

   (1) 10th grade
   (2) 11th grade
   (3) 12th grade

4. What was your final grade in your last MATH course? ( )

   (1) I withdrew from my last math course.
   (2) F
   (3) D
   (4) C
   (5) B
   (6) A

5. What grade do you think you'll get in this course? ( )

   (1) F
   (2) D
   (3) C
   (4) B
   (5) A

6. Have you completed Algebra 1 before? ( )

   (1) no
   (2) yes

RAND

Algebra Tutor Project

7. How interested are you in learning Algebra 1? ( )

(1) No interest
(2) Little interest
(3) Some interest
(4) Great interest

8. How do you feel about Algebra? ( )

(1) I dislike it
(2) I mildly dislike it
(3) I mildly like it
(4) I like it

9. How difficult is Algebra for you? ( )

(1) Easy
(2) Hard
(3) Uncertain
(4) I have not worked on Algebra

10. Have you ever taken a course about computers or computer programming? ( )

(1) no
(2) yes

11. Have you ever used computers in other classes? ( )
(for example, in English, Math, or history class?)

(1) no
(2) yes

12. Have you ever used computers outside of school? ( )
(for example, at home, at a friend's house, at camp?)

(1) no
(2) yes

13. Have you ever used a computer to do your homework? ( )

(1) no
(2) yes

RAND

Algebra Tutor Project

14. Have you ever played computer games? ( )

(1) no
(2) yes

15. Have you ever programmed a computer? ( )

(1) no
(2) yes

16. Do you think you'll like learning Algebra on a computer? ( )

(1) no
(2) yes

Please indicate the degree to which you agree with the following statements about computers. There are no wrong or right answers, we just want your opinion.

17. I believe many algebra courses could be improved by the use of computers.

(1) Strongly disagree ( )
(2) Disagree
(3) Slightly disagree
(4) Slightly agree
(5) Agree
(6) Strongly agree

18. Using computers to teach algebra is a bad idea.

(1) Strongly disagree ( )
(2) Disagree
(3) Slightly disagree
(4) Slightly agree
(5) Agree
(6) Strongly agree

19. Computers can do a lot more teaching than most people realize.

(1) Strongly disagree ( )
(2) Disagree
(3) Slightly disagree
(4) Slightly agree
(5) Agree
(6) Strongly agree

RAND

Algebra Tutor Project

20. It is a waste of time to try to learn using computers.

(1) Strongly disagree ( )
(2) Disagree
(3) Slightly disagree
(4) Slightly agree
(5) Agree
(6) Strongly agree

21. Computers can show students what's important to be learned.

(1) Strongly disagree ( )
(2) Disagree
(3) Slightly disagree
(4) Slightly agree
(5) Agree
(6) Strongly agree

22. Computers are poor substitutes for algebra teachers.

(1) Strongly disagree ( )
(2) Disagree
(3) Slightly disagree
(4) Slightly agree
(5) Agree
(6) Strongly agree

THANKS FOR YOUR HELP! PLEASE CHECK TO MAKE SURE THAT YOUR STUDENT I.D. NUMBER IS ON THE FIRST PAGE.

RAND

Algebra Tutor Project

Achievement Test

DIRECTIONS: The purpose of this test is to find out how much you know about algebra. Some of these problems are difficult. No one expects you to know all the answers. If you cannot solve a problem, don't be discouraged, just go to the next problem. Please try to solve all the problems and show all your work.

1. Hector made an error when he solved this problem for x. See if you can find the error and put a circle around it. Then try to solve the problem correctly.

$$x + 4 = 10$$

$$x + 4 + 4 = 10 - 4$$

$$x + 8 = 6$$

$$x = -2$$

2. Jane did not solve this problem correctly. Try to find the error and put a circle around it. Then try to solve the problem correctly.

$$5 < -3x - 7$$

$$5 + 7 < -3x - 7 + 7$$

$$12 < -3x$$

$$12/-3 < -3x/-3$$

$$-4 < x$$

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

3. Hyon and Jim worked on this problem together, but they made an error. See if you can find and circle their error. Then solve the problem correctly.

Solve the following equations for x: $2x + y = 4$
$x + y = 2$

$$2x + y = 4$$
$$x + y = 2$$

$$2x + y = 4$$
$$y = x + 2$$

$$2x + (x + 2) = 4$$

$$3x + 2 = 4$$

$$3x + 2 - 2 = 4 - 2$$

$$3x = 2$$

$$x = 2/3$$

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

4 - 5 Directions: Solve each problem for x.

4. Solve for x:

$$7(x + 9x + 4) = 12$$

5. Solve for x:

$$-7x < 70$$

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

6 - 7 Directions: Solve each problem for x.

6. Solve for x:

$$3[x + 4(2x + 2)] = 10$$

7. Solve for x:

$$-4(x + 1) < 4(3 + 1)$$

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

8 - 9 Directions: Solve each problem for x and y.

8. Solve for x and y:

$$-7x + y = 3$$

$$5x - 4y = -1$$

9. Solve for x and y:

$$-7x + 3y = 3$$

$$5x - 4y = -1$$

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

10 - 11 Directions: Try to solve each problem below. Then, if you can, try to solve the problem in a different way and mark the method that you like the best.

For example, you could solve x+4 = 2x+2 one way:

Example	**Explanation**
x+4 = 2x+2	
4 = x+2	**You add -x to both sides of the equation**
2 = x	**Then you add -2 to both sides of the equation**

or another way:

x+4 = 2x+2	
x+2 = 2x	**You add -2 to both sides of the equation**
2 = x	**Then you add -x to both sides of the equation**

I like the second way best.

10. Solve for x:

$$7x - 2 = 5x - 8$$

11. Solve for x:

$$4(x + 1) - 2(x + 1) = 6$$

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

12 - 13 Directions: Try to solve each problem below. Then, if you can, try to solve the problem in a different way and mark the method that you like the best.

For example, you could solve x+4 = 2x+2 one way:

Example	**Explanation**
x+4 = 2x+2	
4 = x+2	**You add -x to both sides of the equation**
2 = x	**Then you add -2 to both sides of the equation**

or another way:

x+4 = 2x+2	
x+2 = 2x	**You add -2 to both sides of the equation**
2 = x	**Then you add -x to both sides of the equation**

I like the second way best.

12. Solve for x:

$$-3x > -9(3 + 5)$$

13. Solve for x and y: $x + y = 2$

$$x - y = 6$$

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

14 - 16 Directions: Here are some word problems. Write each problem as an equation and then solve the equation.

14. If the width of a rectangle is 3 inches less than its length, and the perimeter of the rectangle is 94 inches, what is the length of the rectangle in inches?

15. You are taking a history course. There will be 4 tests. You have scores of 89, 92, and 95 on the first three. You must make a total of 360 to get an A. What *scores* on the last test will give you an A?

16. A telephone coin box contains 15 coins. The box contains only nickels and dimes, and the total value of the coins is 95 cents. Find out how many coins are nickels and how many are dimes.

RAND

Algebra Tutor Project

IF YOU CANNOT SOLVE A PROBLEM, JUST GO TO THE NEXT QUESTION.

17 - 19 Directions: We will give you the answer to a problem and ask you to write an equation.

For example, write an equation in which the answer is x = 3. The equation should have at least five appearances of x:

Example	**Explanation**
x = 3	
x + 2x = 3 + 2x	**Add 2x to both sides**
x + 2x - 9x = 3 + 2x - 9x	**Add -9x to both sides**

17. Write an equation in which the answer is x = 2. The equation should have at least one set of parentheses.

18. Write an inequality in which the answer is x < 0. The inequality should have at least two negative coefficients.

19. Write a system of equations in which the answers are x = 4 and y = 6.

REFERENCES

Bikson, T. K., and J. D. Eveland, *New Office Technology: Planning for People*, Work in America Institute, Inc., 1986.

Bloom, B. S., "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educational Researcher*, Vol. 13, No. 6, June/July 1984.

Brown, J. S., "Process versus Product—a Perspective on Tools for Communal and Informal Electronic Learning," in *Education and the Electronic Age*, proceedings of a conference sponsored by the Educational Broadcasting Company, Boston, Mass., 1984.

Brown, J. S., R. R. Burton, and J. deKleer, "Pedagogical, Natural Language and Knowledge Engineering and Pedagogical Techniques in SOPHIE I, II, and III," in D. H. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems*, Academic Press, New York, 1982.

Carpenter, T. P., M. K. Corbitt, H. S. Kepner, M. M. Lindquist, and R. E. Reys, "Students' Affective Responses to Mathematics: Secondary School Results from National Assessment," *Mathematics Teacher*, October 1980, pp. 531–539.

Clancey, W. J., "Tutoring Rules for Guiding a Case Method Dialogue," *International Journal of Man-Machine Studies*, Vol. 11, 1979, pp. 25–49.

——, "GUIDON," *Journal of Computer-Based Instruction*, Vol. 10, 1983, pp. 215–251.

Clark, R. E., "Confounding in Educational Computing Research," *Journal of Educational Computing Research*, Vol. 1, No. 2, 1985, pp. 137–148.

Collins, A., and J. S. Brown, "The Computer as a Tool for Learning through Reflection," in H. Mandl and A. Lesgold (eds.), *Learning Issues for Intelligent Tutoring Systems*, Springer, New York, 1987.

Collins, A., J. S. Brown, and S. E. Newman, "Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics," in L. B. Resnick (ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, Lawrence Erlbaum, Hillsdale, N.J., 1989.

Cook, T. D., and D. T. Campbell, *Quasi-Experimentation: Design and Analysis Issues for Field Settings*, Rand-McNally College Publishing Company, Chicago, Ill., 1979.

Dienes, Z., *Building Up Mathematics*, 4th ed., Hutchinson Educational Ltd., London, 1960.

Fey, J. T., *Computing and Mathematics: The Impact on Secondary School Curricula*, report on the National Council of Teachers of Mathematics Conference, Washington, D.C., 1984.

Hollan, J. D., E. L. Hutchins, and L. Weitzman, "STEAMER: An Interactive Inspectable Simulation-Based Training System," *AI Magazine*, Vol. 5, No. 2, 1984, pp. 15–27.

Kaput, J., "Towards a Theory of Symbol Use in Mathematics," in C. Janvier (ed.), *Problems of Representation in the Teaching and Learning of Mathematics*, Lawrence Erlbaum, Hillsdale, N.J., 1986.

Kaput, J., C. Luke, J. Poholsky, and A. Sayer, *The Role of Representations in Reasoning with Intensive Quantities: Preliminary Analyses*, Educational Technology Center Technical Report, Cambridge, Mass., September 1986.

Lesh, R., M. Behr, and T. Post, "Representations and Translations among Representations in Mathematics Learning and Problem Solving," in C. Janvier (ed.), *Problems of Representation in the Teaching and Learning of Mathematics*, Lawrence Erlbaum, Hillsdale, N.J., 1987.

McArthur, D., "Developing Computer Tools to Support Performing and Learning Complex Cognitive Skills," in K. Pedzek, D. Berger, and B. Bankes (eds.), *Applications of Cognitive*

Psychology: Computing and Education, Lawrence Erlbaum, Hillsdale, N.J., 1987. (Also published by The RAND Corporation, N-2980-NSF, 1989.)

McArthur, D., C. Burdorf, T. Ormseth, A. Robyn, and C. Stasz, "Multiple Representations of Mathematical Reasoning," *Proceedings of the International Conference on Intelligent Tutoring Systems*, Montreal, June 1988. (Also published by The RAND Corporation, N-2758-NSF/RC, 1988.)

McArthur, D., M. W. Lewis, T. H. Ormseth, A. Robyn, C. M. Stasz, and D. A. Voreck, *Algebraic Thinking Tools: Supports for Modeling Situations and Solving Problems in Kids' Worlds*, The RAND Corporation, N-2974-NSF, July 1989.

McArthur, D., and C. Stasz, "Tutoring Techniques in Algebra," presented at the American Education Research Association annual conference, Washington, D.C., April 1987.

McArthur, D., C. Stasz, and J. Hotta, "Learning Problem-Solving Skills in Algebra," *The Journal of Educational Technology Systems*, Vol. 15, No. 3, pp. 303–324. (Also published by The RAND Corporation, N-2595-NSF, 1987.)

McArthur, D., C. Stasz, J. Hotta, O. Peter, and C. Burdorf, "Skill-Oriented Task Sequencing in an Intelligent Tutor for Basic Algebra," *Instructional Science*, Vol. 17, 1988, pp. 281–307. (Also published by The RAND Corporation, N-2966-NSF, 1989.)

McArthur, D., C. Stasz, and M. Zmuidzinas, "Tutoring Techniques in Algebra," *Cognition and Instruction*, in press.

Newell, A., "The Knowledge Level," *Artificial Intelligence*, Vol. 18, No. 1, 1982, pp. 87–127.

Newman, D., *Formative Experiments on Technologies That Change the Organization of Instruction*, paper presented at the annual meeting of the American Educational Research Association, Chicago, Ill., 1989.

Ohlsson, S., "Some Principles of Intelligent Tutoring," *Instructional Science*, Vol. 14, 1986, pp. 293–326.

Pea, R., "Cognitive Technologies for Mathematics Education," in A. H. Schoenfeld (ed.), *Cognitive Science and Mathematics Education*, Lawrence Erlbaum, Hillsdale, N.J., 1987.

Rachlin, S. L., A. Matsumoto, and L. Wasa, "Teaching Problem Solving within the Algebra Curriculum," paper presented at the annual meeting of the American Educational Research Association, Chicago, Ill., 1985.

Robyn, A., C. Stasz, T. Ormseth, and D. McArthur, "Implementing Computer-Assisted Instruction in First-Year Algebra Classes," paper presented at the annual meeting of the American Education Research Association, San Francisco, Calif., April 1989.

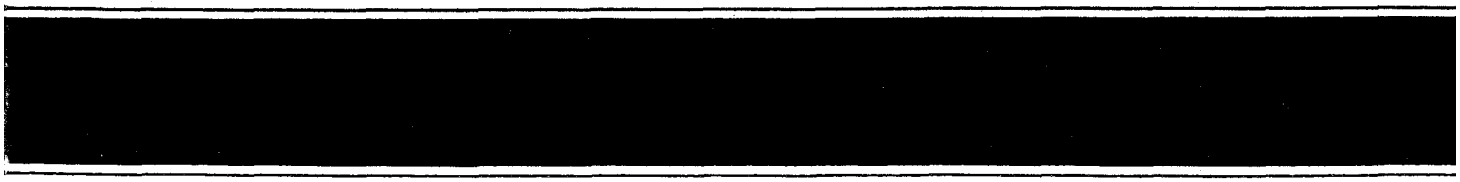
Schoenfeld, A. H., "Students' Beliefs about Mathematics and Their Effects on Mathematical Performance: A Questionnaire Analysis," paper presented at the American Educational Research Association annual conference, Chicago, Ill., 1985.

——, "What's All the Fuss about Metacognition?" in A. H. Schoenfeld (ed.), *Cognitive Science and Mathematics Education*, Lawrence Erlbaum, Hillsdale, N.J., 1987.

Shavelson, R. J., C. Stasz, S. Schlossman, N. Webb, J. Y. Hotta, and S. Goldstein, *Evaluating Student Outcomes from Telecourse Instruction: A Feasibility Study*, The RAND Corporation, R-3422-CPB, 1986.

Shavelson, R. J., N. Webb, C. Stasz, and D. McArthur, "Teaching Mathematical Problem Solving: Insights from Teachers and Tutors," in R. Chambers and E. Silver (eds.), *Teaching and Assessing Mathematical Problem Solving: A Research Agenda*, Lawrence Erlbaum, Hillsdale, N.J., 1989.

Sleeman, D. H., "PIXIE: A Shell for Developing Intelligent Tutoring Systems," in R. Lawler and M. Yazdani (eds.), *Artificial Intelligence and Education: Learning Environments and Intelligent Tutoring Systems*, Ablex, Norwood, N.J., 1987.

Sleeman, D. H., and M. J. Smith, "Modeling Students' Problem Solving," *Artificial Intelligence*, Vol. 16, 1981, pp. 171–188.

White, B. Y., and J. R. Frederiksen, "Qualitative Models and Intelligent Learning Environments," in R. Lawler and M. Yazdani (eds.), *Artificial Intelligence and Education: Learning Environments and Intelligent Tutoring Systems*, Ablex, Norwood, N.J., 1987.