Artificial Intelligence, Advanced Technology, and Learning and Teaching Algebra*

Patrick W. Thompson
Department of Mathematics
Illinois State University

Thompson, P. W. (1989). Artificial intelligence, advanced technology, and learning and teaching algebra. In C. Kieran & S. Wagner (Eds.), *Research issues in the learning and teaching of algebra* (pp. 135–161). Hillsdale, NJ: Erlbaum.

*Revised version of a paper presented at the NCTM's Research Agenda Project on Algebra, Athens, Georgia, March 25-28, 1987. Research reported in this paper was supported in part by NSF Grant No. MDR 87-51381 and by a grant of equipment from Apple Computer, Inc. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of NSF or Apple Computer.

Artificial intelligence is mentioned frequently these days, many times in the context of an advertisement that tries to sell a new expert system or expert system shell. Given the current interest but general lack of knowledge about the field, perhaps it would be useful to define what is meant by "artificial intelligence."

Originally, artificial intelligence (hereafter, AI) was the study of how to make machines behave intelligently. By "behave intelligently," it was meant that a machine perform a task that normally would require a human to do because the task required reasoning of some sort (McCorduck, 1979). This performance definition was soon abandoned, largely because as a task became doable by a machine, many people took that as indicative that it actually did not require intelligence. Now, AI is defined largely by a particular programming style: knowledge to perform a task is represented explicitly within a program, as data. The form such data takes is some discrete structure, such as a tree or digraph, or a collection of rules, like

"If some condition exists (in memory), then some conclusion should be drawn," or

"If some condition exists (in memory), then some action should be taken,"

where some condition and some conclusion are items of data or data patterns and some action is a procedure or procedure name.

"Reasoning" is done by a procedure that interprets knowledge-as-data and draws inferences from it; inferences so drawn are then added to the "knowledge base." The behavior of an artificially intelligent program is determined by the interaction of three components: the rules of inference built into the inference procedure, the knowledge with which the program starts, and the task itself. Of course, "the task" is itself a knowledge structure, for it is represented in the program as the program's knowledge of the initial conditions and of the conditions to be achieved. The art of AI is to analyze tasks in such a way that one can depict knowledge capable of leading to correct performance under widely varying circumstances.

There should be no more mystique about AI programming than about programming in general, although in fact this seems not to be the case. The only major difference between AI programming and programming in general is that an AI programmer has at his or her disposal very powerful programming languages——languages in which structure and relationship can be represented explicitly and easily, and in which there is no hard and fast distinction between data and procedure.

After saying the above, I must confess that I faced a fundamental difficulty in preparing this paper. Research in AI has focused, for the most part, on building programs that can solve problems, once posed, independently of human intervention (Feigenbaum & Barr, 1984). Also, AI programs tend to be quite task oriented. While such programs may have educational and pedagogical value, it is not apparent that the value of any program extends beyond a fairly restricted task domain.

An artificially intelligent program that can solve mathematics problems may be useful as a practical tool, but giving students the capability to obtain answers per se is not a primary goal of mathematics education. It is important to affect students' thinking in a way that develops skill, but a higher-level goal is to affect students' thinking so that it holds potential for their constructing new and more powerful ways of thinking in the future (Dewey, 1949). To achieve the higher-level goal, students must come to think in terms of such things as patterns, analogies, and metaphors. Research in AI has made significant strides in explaining patterns in thought, but it has not made much headway with the issues of analogy and metaphor.

In the remainder of this paper I excersize an author's perogative. Rather than attempt to survey the AI literature, annotating the survey with cryptic comments and leaving it to the reader to make sense of them from the originals, I instead focus on two general aspects of AI vis-a-vis mathematics education. I examine past research on the development of intelligent tutoring systems and give samples of a "new breed" of software. The section on intelligent tutoring systems attempts to set a tone: though AI has much to offer, we

should view its research as applied to education with as much skepticism as we would any other educational research enterprise. In the second section, I review several current projects that illustrate the power of AI concepts and methodology for developing systems that present algebraic content in substantially new ways.

Intelligent Tutoring Systems

Most research connecting AI with mathematics education has been to develop intelligent tutoring systems (ITS's), which are programs that are intended to mimic the behavior of a competent tutor (see Sleeman & Brown, 1982 for one of the few collections on ITS's). At the present moment, our reaction to ITS research might be like Sam Johnson's reaction to the dog that walked on its hind legs. He remarked that we should not be surprised that the dog walks well or poorly, but that it walks at all. At that level, we can be impressed by the progress of ITS research. At a more substantive level, were I to interview an applicant for a tutoring position who displayed the heavy-handedness and lack of perspective exhibited by many ITS's, he or she would definitely not get the job. This may be more a comment on system designers' concepts of learning and teaching mathematics than a comment on the potential of intelligent tutoring systems in mathematics education.

My strongest criticism of ITS research is that, by design, there is no teacher in the picture, except perhaps as someone who sets system parameters. This is acceptable as a research ploy when one pushes a concept to see how far it can go on its own. But as a paradigm I believe leaving the teacher out is a mistake. It is my experience that when one designs software with the assumption it will be used by a teacher as well as by students, one designs the software so that it supports the teacher in improving instruction, which results in richer learning by students.

For further information on ITS research, I refer the reader to the bibliography (most notably, Kearsley, 1987; Sleeman & Brown, 1982; Wenger, 1987).

The Hazards of Myopia

When searching the AI literature as it relates to education, one quickly realizes the paradigmatic value given to Brown and Burton's BUGGY model of errors in place value subtraction (Brown & Burton, 1978; Brown & VanLehn, 1981; Burton, 1982; VanLehn, 1983). In that model, students errors are depicted as bugs——"discrete modifications to correct skills which effectively duplicate the student's behavior" (Burton, 1982). A similar approach to analyzing errors in algebra has been taken by Matz (1980, 1982), Lewis (1981), and Sleeman (1982, 1984, 1985). In these studies, competence in algebra is characterized as the possession of a set of correct algebraic rules. For example, one rule might be

If the goal is DISTRIBUTE,
and the current expression is of the form A(X-B),
then write AX - AB.

Errors are characterized as manifestations of incorrect rules (mal-rules, to use Sleeman's term).

A rule orientation is entirely natural given the constraints of this research: that representations of knowledge be data within a program, and that representations of knowledge be prescriptions for action. A rule orientation also is natural given that the goal was to develop systems capable of solving the problems posed to students, that were able to solve unanticipated problems arising from interactions with the student, and that were capable of determining the cause (i.e., "violated rule") of students' errors.

Aside from very serious epistemological problems with BUGGY, more pragmatic problems exist as well. BUGGY-like models of competence, whether used in place-value subtraction or high school algebra, are quite fragile and contribute less to our understanding

than at first might be apparent. I say this for four reasons: (1) the principle construct of BUGGY-like models, a "rule," has not been clarified; (2) it is not clear that BUGGY-like models in fact model anything of interest; (3) tacit assumptions about aims of instruction may not be valid; and (4) BUGGY-like models commonly ignore relationships between areas of mathematics.

What are rules? A rule, in AI, is a condition-action pair of the forms shown on page [2] (Negotia, 1985). This use of "rule" is so general that it cannot have any psychological meaning, except as a descriptor of behavior. Rules can be used to describe reasoning (i.e., "mental behavior"), but they could just as well be used to describe the behavior of a spring under various distortions. In the final analysis, rules (in AI) are programmers' abstractions of initial and final states under some class of transformations. They tell us more about system designers than about students' mathematics.

Are BUGGY models of interest? What BUGGY-like systems model is, in effect, students' errors arising from their use of means-end analysis to accomplish tasks of which they have no understanding and whose goal is to imitate some observed behavior in order to produce an answer (see Erlwanger, 1973). Put another way, BUGGY-like models are models of common errors made by students who reason without meaning. BUGGY-like models may accurately describe the current state of mathematics learning, but we already know that students are prone to pushing symbols without engaging their brains. In what way does a detailed understanding of how students perform tasks mindlessly help us improve mathematics education?

Perhaps the most damaging consequence of defining competence merely as possession of correct rules is that we fail to look at incompetence as stemming from impoverished conceptualizations of material from which "correct" rules should have been abstracted. A student who consistently transforms $a(x+b)$ into $ax+b$, for example, probably would benefit more from instruction on order of operations and structures of expressions in arithmetic than from direct instruction on how to "distribute" across parentheses in algebra.

Assumptions about instruction. BUGGY models are fragile because they assume, overtly or covertly, that instruction emphasizes mainly the learning of procedures for making marks on paper. In place-value subtraction, for instance, when instruction emphasizes the intricacies of counting by hundreds, tens, and ones from arbitrary starting points and relationships between counting and structured materials (e.g., Dienes blocks), one rarely sees evidence of the bugs reported by Brown and Burton (Thompson, 1982). One still sees errors, but they are more apparently conceptual, with little resemblance to the errors predicted by BUGGY.

Similarly, there are reasons to believe that if algebra instruction emphasizes the concept of an expression as an entity having an internal structure, and if "rules" are proposed as structure-modifying transformations which leave some aspect of an expression invariant, then common errors as reported by Lewis, Matz, and Sleeman seem not to be so common (Lesh, 1987; Thompson & Thompson, 1987). I will discuss this claim again in a later section.

Relationships to other concept fields. Studies in algebra that incorporated BUGGY-like models of competence typically have not considered relationships between algebra and arithmetic. After first reading Matz (1982), I gave one of her examples to a ninth grader:

$$\text{Solve for } x\text{: } \frac{x+1}{x+4} = \frac{5}{6} \quad \text{(Matz, 1982, p. 50)}$$

to which he gave the answer predicted by Matz, namely $x+1=5$ and $x+4=6$, or $x=4$ and $x=2$. I then said, as if offering a new problem: "I am thinking of a number: if you add 1 to it and then add 4 to it and make a fraction with these two numbers, the fraction reduces to five-sixths," while writing the same equation as before.

This student's strategy in solving the restated question

appeared to be based on the idea that for $\frac{x+1}{x+4}$ to reduce to $\frac{5}{6}$, $\underline{x}$ had to be congruent to 4 modulo 5, and at the same time $\underline{x}$ had to be congruent to 2 modulo 6. Of course, this is my characterization and not his. After listing multiples of 5 and multiples of 6, he got a correct answer: $\underline{x}$=14. This student knew a lot about ratio and proportion, but he did not relate his knowledge to the problem when it was first presented. I grant that he did not use the procedure typically expected for solving this problem, but he did solve it. Moreover, the typical procedure could have been taught as a generalization of the process by which one "cross multiplies" fractions to test their equivalence.

I did not offer this example to "disprove" BUGGY-like models of algebraic competence. Rather, I only wished to reinforce the point that when algebra is taught as a system for representing relationships among quantities, one sees very different, and perhaps more desirable, types of behavior than expected under the more stereotypical circumstances assumed by users of BUGGY-like models. Moreover, if one is sensitive to the role of prior arithmetical learning in learning algebra and is sensitive to conceptual contexts for the formation of algebraic "rules," then when designing an intelligent tutoring system for algebra one will design the system so as to make it possible for students to reveal impoverished conceptualizations as well as buggy rules. I discuss the last point more fully in (Thompson, 1987).

Presenting Content in Substantially New Ways To Support Human Understanding

A new approach to software design is emerging from AI. The spirit of the approach is captured well in Draper and Norman's introduction to User Centered System Design.

> ...we do not wish to ask how to improve upon an interface to a program whose function and even implementation has already been decided. We wish to attempt User Centered System Design, to ask what the goals and needs of the users are, what tools they need, what kind of tasks they wish to perform, and what methods they would prefer to use. We would like to start with the users, and to work from there.
>
> (Draper & Norman, 1986, p. 2)

Norman and Draper's collection constitutes an attempt to combine various perspectives from the cognitive sciences on topics that range from being a better typist to developing mental models of advanced technological devices, and to focus these perspectives on how best to shape the interaction between humans and computers. They note that in the pluralism of approaches there exists the germ of a new kind of interaction.

> One approach, a rather traditional one, is to start with considerations of the person, the study of the human information processing structures, and from this to develop the appropriate dimensions of the user interface. .... Another approach is to examine the subjective experience of the user and how it might be enhanced. When we read or watch a play or movie, we do not think of ourselves as interpreting light images. We become a part of the action: We imagine ourselves in the scenes being depicted. We have a "first person experience."...Well, why not with computers? The ideal associated with this approach is the feeling of "direct engagement," the feeling that the computer is invisible, not even there; but rather, what is present is the world we are exploring, be that world music, art, words, business, mathematics, literature, or whatever your imagination and task provide you. [italics added]
>
> (Draper & Norman, 1986, p. 3)

The idea proposed by Draper and Norman, that we examine and enhance the users' subjective experiences, together with emerging refinements in the technology of object-

oriented programming, can be synthesized into powerful, supportive environments to make real what many competent teachers of algebra wish they could accomplish. "If only I could get them to see in their minds what I see in mine!"

I will present aspects of four projects, all currently ongoing, that illustrate the idea of "direct engagement" between students and computerized, algebraic environments. I will describe them in some detail, as the nature of the software is probably unfamiliar to a large segment of mathematics education. The content areas illustrated are problem solving, equation solving, concept of expression, and concept of equation.

Problem Solving

A question of critical importance is how to prepare students for the transition from arithmetic to algebra. This question did not originate in AI; it has been addressed from various perspectives throughout the century (Stanic & Kilpatrick, in press; Kieran, this volume). One of the most promising approaches is to emphasize issues of problem representation over problem solution in arithmetic problem solving (Herscovics & Kieran, 1980; Kieran, this volume; Miwa, this volume).

The approach illustrated here draws on a number of sources: AI (object-oriented systems; see Kay, 1984), cognitive science (semantic networks; see Resnick & Ford, 1981), and mathematics education. The focus of the project is to have students represent arithmetic and algebra word problems by representing the quantities involved and relationships among them. The computer program used in this project, which runs on a Macintosh Plus, was inspired by a program created by Valerie Shalin, Nancy Bee, and Ted Rees to run on a Xerox Dandelion (described in Greeno, 1985 and in Greeno, Brown, et al., 1985).

An object-oriented computer program is one that presents a user with what he or she identifies as objects (e.g., a graph or an image) and which possesses an internal representation of those objects and their properties. The program allows the user to act on objects in well defined and natural ways, and responds to those actions as one would predict on the basis of (perhaps informed) intuition.

A semantic network is a graph that depicts items of knowledge and relationships among them (Sowa, 1984). Semantic networks have been used in AI to imbue programs with "knowledge" about some domain. They have been used as theoretical constructs in cognitive science to formulate hypotheses about knowledge structures that are expressed in behavior. The novelty of the approach taken by Shalin et al., and extended here, is that the computer is used as a medium for students to externalize their knowledge structures and thereby have the potential to refine them.

The computer program illustrated here, called Word Problem Assistant (WPA), presents students with a menu of icons with which to represent the quantities involved in a problem. There are four icons to represent four distinct kinds of quantities: numbers of things, rates, differences, and ratios. At the time of selecting an icon, the student types a natural-language description of the quantity and types the unit in which the quantity is measured (Figure 1).

Figure 1 About Here

The handles on the icons are a reminder of the kind of quantity being represented. The labels within cells (other than the unit cell) tell the student the information that is appropriate for entry. To enter information, a student puts the pointer over the cell of interest, clicks the mouse button, and then types the information. Figure 2 shows a student's representation of the quantities in this problem: A biologist released 200 marked fish into a lake. He later captured six samples of fish. On the average, the number of marked fish in each sample was 1/60 of the sample size. Approximately how many fish are in the lake?

Figure 2 About Here

To show relationships among the quantities in a problem, a student simply uses the mouse to draw arrows between them. Figure 3 shows that the quantity No. Marked Fish is determined by Total No. Fish and Mark Rate. Figure 3 also shows a feature of WPA: whenever there is sufficient information to fill a cell, WPA does it, and puts a bullet (•) in the line to show that the contents of that cell were inferred. After the student drew the arrows, WPA inferred that Total No. Fish should be the quotient of Mark Rate and No. Marked Fish, since No. Marked Fish is the product of Total No. Fish and Mark Rate.

Figure 3 About Here

Figure 3 shows another feature of WPA. It is that operations are not stated explicitly. Instead, operations are implied by the kinds of units that compose the relation. The intent is that students' attention be focused on the quantities involved and relationships among them instead of on what formulas to apply. In that way, formulas are experienced as arising from an understanding of a problem instead of as something that should be sought early on in solving the problem. This approach also is consistent with Kintsch and Greeno (1985), who propose that a necessary feature of a "good" mental representation is that it include information sufficient to determine the operations necessary to solve the problem.

To make explicit the independence of problem structure and specific problem, WPA includes these features: (1) If a value or description is changed, then all affected inferences are likewise changed. Thus, one can vary values to correspond with variations of the original problem. (2) One can change the pattern of information. For example, instead of entering values for Mark Rate and No. Marked Fish, we could enter values for Total No. Fish and Mark Rate. In this way, we can make explicit the idea of different problems being variations on a single theme.

WPA also was designed to make explicit the notion of a problem parameter, and to make explicit the distinction between a parameter and a variable. Two example will illustrate this distinction.

Imagine this scenario: A teacher proposed the following problem to a class while using WPA with a projector or large screen monitor. While driving, John accelerated at 7 ft/sec/sec for 10 seconds. Afterward, he drove at a constant speed. He drove 1000 ft in the first 5 seconds after he stopped accelerating. What was John's speed as he began to accelerate? Then, the teacher guided the discussion, leading to the representation given in Figure 4.

Figure 4 About Here

In setting this problem up, the teacher created icons for each quantity of the problem, plus the two others not mentioned (Added Speed and Final Speed). She entered values into Distance, Const. Speed Time, Acceleration, and Acceleration Time. Then, she drew arrows as shown. WPA inferred descriptions and values for Final Speed, Added Speed, and Initial Speed. After solving this problem and discussing the solution, she said, "We will have to solve another problem just like this one, except with different numbers. But Mrs. Snodgrass will have the computer; we won't be able to use WPA. Let's generate a formula so that we can solve tomorrow's problem without having to first set it up." Then the teacher typed letters into the Description cells of the quantities initially known. As she typed letters, WPA propagated the letters in place of the previously stored values, resulting in the representation shown in Figure 5. The letters in this representation are problem parameters, to be given values at some later time. They are not essential to the solution of any specific problem having this structure.

Figure 5 About Here

The distinction between variables and parameters arises from distinctions among problems in terms of the necessity of using letters to represent them. The following problem requires the use of letters in its representation: John is now four times as old as Sally. In five years, John will be three times as old as Sally. How old are John and Sally? Figure 6 shows a representation in WPA of this problem. All values were entered; descriptions were inferred by WPA. The Comparison 2 quantity holds the seed of the solution, as it has both a description and a value. The student can solve the problem by selecting the Comparison 2 icon and then selecting Solve Equation in the Options menu.

Figure 6 About Here

The reason that letters are required in this problem is that while we may assign a range of values to John's age, whatever value we assign must satisfy two constraints to make the network consistent. This suggests that solving problems that have the structure of simultaneous equations would be a cognitively supportive context in which to teach the concept of letter-as-variable (which would include "graphing" approaches to solving equations; see Lesh, 1987; Kaput, this volume).

There are many features of WPA that I would like to discuss, especially those that concern its use in instruction, but cannot for lack of space. To summarize, in traditional settings we try to communicate this message to students: A good representation of a problem is the most important step toward a solution. With WPA, that message is a fact.

Equation Solving

McArthur (1985) and McArthur, Stasz, and Hotta (in press) describe the first version of a system that will eventually grow into an intelligent tutor for solving algebraic equations. In its current form it does not tutor. Rather, it includes what McArthur calls "an inspectable expert." That is to say, a student can ask the "expert" to show the details of its reasoning.

Figure 7 shows a sample screen.[1] In McArthur's system, a portion of the display shows a reasoning tree. A reasoning tree depicts the steps taken in solving an equation, where branches from any step indicate that the student has attempted more than one solution path. The number of options available to a student is quite large, as can be seen from Figure 7. McArthur, Stasz, and Hotta (in press) give a full description of how these options are used. I will discuss only those options that make the expert "inspectable" and that support students' understanding of reasoning in equation solving.

Figure 7 About Here

A student begins a problem by clicking on New Question (middle left side of the screen in Figure 7). After entering the initial equation to solve, the student clicks New Line and types a new line. Under this mode of operation, the display shows what would be shown were the student recording a solution with paper and pencil.

The difference between using McArthur's system and using paper and pencil is in the editing features and checking features provided by the former. Figure 8 shows the display after the student selected Go Back and selected the equation $(-6)x = -17+15+(-9)x$

[1] Figures depicting McArthur's system are taken from McArthur, Stasz, and Hotta (in press).

as the step to go back to. The program created a branch from that step to indicate that any new equations constitute a different solution path than the one he or she was on.

Figure 8 About Here

To activate the expert, a student can select a step (by clicking on a line that connects two equations) and then select Step Correct? from the menu options. The expert will respond yes or no to correctness, and if the step is correct, it will comment on the appropriateness of the step for achieving the goal of solving the equation. The student can also ask the expert to supply a next step, by clicking on the Do Next Step menu option. If the student doesn't understand what the expert has done, he or she can request further detail by clicking the Elaborate Step menu option and then clicking on the line connecting the two equations in question.

Figure 9 shows the expert's elaboration of a step. The student typed the equation $-6x = -2 + -9x$ (continuing from Figure 8). Then he asked the expert to supply the answer. Then he asked the expert to elaborate on its solution. To request the elaboration, the student clicked Elaborate Step, and then clicked on the equations $-6x = -2 + -9x$ and $x = \frac{-2}{3}$. The expert then supplied more detailed steps for deriving the second equation from the first.

Figure 9 About Here

As with Draper and Norman's exhortation, we must ask of the intended subjective experiences to be had by students who use McArthur's system. Clearly, one is that students will develop a sense of heuristic search in applying expression-transforming operations: If one path isn't productive, go back to solid footing and try another approach.

A second field of experiences, based on the use of STEP OK?, DO NEXT STEP, and ELABORATE STEP, is intended to develop the idea that knowledge itself is something that can be reasoned about.

> ... these activities teach the student that an important part of learning a cognitive skill is learning to study your own reasoning processes. [italics in original]....[W]hen students are asked why they do poorly on a mathematics test in grade school, common attributions are "I'm dumb in math.", or "I had a bad test." or "The test was unfair." Very few identify specific knowledge that they might have lacked, or even understood that their correctable lack of knowledge might have been responsible for failure.
>
> (McArthur et al., in press [p. 17 in pre-publication manuscript])

The interactions between students and McArthur's system are possible without a computer, but just barely so. Even so, the modes of presentation and interaction used in this system demand the flexibility and rapidity provided only with a computer display.

Concept of Expression

In (Thompson, 1987) I outlined design considerations for developing what I call mathematical microworlds——computer programs that embody a mathematical system in a way that promotes mathematical explorations. I also gave an example of a program under development that embodied an intelligent mentor to aid students in their explorations of isometric transformations of the plane. The key to incorporating an intelligent mentor into a mathematical microworld is to study students' conceptions of the subject matter as they

interact with the microworld. By studying conceptions in the context of students' work with a microworld, one finds not only their fundamental misconceptions, but also how those misconceptions are reflected in students' interactions with the microworld. One can then design a "bug catalog" that is sensitive to qualitative, fuzzy misunderstandings. The first step to this goal is to study students' conceptualizations.

The program, called EXPRESSIONS, is intended to emphasize cognitive and structural features of algebra that are not treated typically, and which appear to be sources of many students difficulties in algebra. These are:

1. Expressions and equations have internal structure, and the structure of an expression or equation constrains what may be done to it (Greeno, 1982; cited in Kieran, this volume).
2. Expressions and equations (henceforth, "expressions") are objects, and as such they may be acted upon.
3. A field property or an identity is a statement of relationship between an expression, the application of the property or identity, and the result of that application.

A premise that unifies the three listed above is:

4. Multiple representations of an object, with actions performed with one being reflected by changes in the other, facilitate students' development of relational understanding (Skemp, 1978) of the content.

Expressions are presented in two forms: in conventional (sentential) notation and in the form of an expression tree. An expression in sentential notation can be displayed with or without superfluous parentheses, and multiplication can be made to be represented explicitly or implicitly.

An action upon an expression is selected by clicking the button with its name. Buttons are located along the right edge of the computer's screen (Figure 10).

Figure 10 About Here

The part of the expression to be acted upon is selected by clicking on the place within the tree that defines the to-be-acted-upon expression. Figure 10 shows the screen after a student entered $(a + b) + (c + d)$ and clicked ASSOCL, which stands for "re-associate from left to right." Figure 11 shows the screen after the student clicked the addition sign at the top of the tree, which indicated that the entire expression was to be acted upon.

Figure 11 About Here

To apply an identity (e.g., $x - y = x + {}^{-}y$), the student clicks ID, selects the identity, and then clicks the top operation of the expression or subexpression to which the identity is to be applied (Figure 12).

Figure 12 About Here

One aim of having students operate upon expressions is to build up a repertoire of identities for future use. Any sequence of operations upon an expression produces an identity, since the available operations are equivalence-preserving. When operating upon an expression, one has, at any time, a relationship of equivalence between the initial and current expressions. By clicking the REMEM button, the initial and current expressions are added to the list of identities, and can be used thereafter. To see the derivation of an identity, the student clicks twice on it when presented in the Identity window (see Figure 12).

Students can operate upon equations in any of three ways. (1) They may operate upon either side with field properties or identities. (2) They may use the buttons ADDL and ADDR to add the same quantity or expression to both sides, or use MULTL and MULTR to multiply both sides by the same quantity or expression. (3) They may replace the buttons ADDL, ADDR, MULTL, and MULTR with the single button OPERATE. Figure 13 illustrates the use of the MULTL button.

Figure 13 About Here

The OPERATE button (not shown so far) allows students to apply the principle that if f(x) is a function and u=v, then f(u)=f(v). Thus, if they want to square both sides of an equation and then add 3 to the result, they would click the OPERATE button and type "? ^ 2 + 3"——meaning that each side of the equation is to be composed with the function $f(x)=x^2 + 3$. To solve the equation $3^{(x^2 + 4)} = 19$, they could click OPERATE, type "log(?)", and apply built-in identities having to do with logarithms.

EXPRESSIONS has been used with eight average, leaving-seventh graders and with a class of preservice elementary school teachers. Students' use of EXPRESSIONS was guided by a workbook that contains graduated activities and exercises (see Thompson, 1985, 1987 for an explanation of graduated exercises in a Piagetian tradition). The exercises began with order of operations in arithmetic expressions, moved to derivations of equivalence of arithmetic expressions by use of field properties, and then moved to derivations of identities of algebraic expressions by use of field properties and identities.

Analyses of videotapes and computer-recorded interactions taken during the seventh grade field trial is given in Thompson and Thompson (1987). Here I will note two observations. First, contrary to my expectation, students found expression trees to be quite intuitive. In fact, they soon preferred working with expression trees to working with expressions in sentential form (all exercises were presented in sentential format).

Second, when planning the pilot, I did not have a clear sense for how to motivate the use of letters in expressions. There was not time for an elaborated treatment, so I decided to just begin using letters and listen closely to the students reactions——and plan accordingly in succeeding studies. To my amazement, students were not bothered by the introduction of letters in expressions. They quickly saw letters as placeholders for substructures in a tree. For example, six of the eight students explained that in applying the associative property, (a+b)+c = a+(b+c), to (3+5)+(7+9), the subexpression (7+9) "is just like c".

The fact that students saw two expressions like (a+b)+c and (3+5)+(7+9) as appropriate for the application of the ASSOCL transformation suggests that they were developing a generalized concept of variable——letters could stand for numbers, but in general they were placeholders within a structure and anything could be put in their place, including other expressions. In another instance, two students, when considering what identities to apply to (a-b)/c in order to obtain a/c–b/c, remarked that "(a-b) is like u" when comparing their expression to the identity u/v = u * 1/v.

Frequently, students are bothered by the introduction of letters in expressions. Why? In their experience, an expression is there to be evaluated (Kieran, this volume), and they cannot evaluate an expression having a letter in it. The seventh graders who used EXPRESSIONS had a different kind of experience. In their use of EXPRESSIONS an expression was there to be manipulated. The presence of a letter in an expression did not affect their ability to manipulate it.

My hypothesis about why students used substitution spontaneously when comparing a complex expression with one that had the same structure but less complexity is this: These students realized the general purpose of letters because of the nature of their activities with arithmetic expressions. When transforming an arithmetic expression to obtain some goal expression, the particular numbers did not matter——only the current and

goal expressions' structures constrained their choices. Also, the arithmetic exercises included many instances of transformations that left subexpressions intact. In working these exercises, students became used to ignoring, for the moment, the details of a subexpression and treating it as a single item in the super-expression. They were thinking in terms of substituting an expression for a variable. A cognitive foundation had been laid for explicit substitution of expressions for letters.

Concept of Equation

Feurzeig (1986) described a multifaceted project, to be used with sixth-graders, that focuses on three aspects of algebraic understanding. These are: algebra as a language for describing actions on and relationships among quantities, algebra as one instance of a class of functional languages, and manipulative skills with expressions and equations. The portion dealing with manipulative skills has resulted in a program similar in spirit to McArthur's program. In this paper, I will discuss only the aspect of Feurzeig's project that focuses on algebra as a language for describing actions on and relationships among quantities. This focus manifested itself in what Feurzeig calls the Marble Bag Microworld.

> The objects in this microworld are pictures of marbles and bags (bags contain some unknown number of marbles); the operations include addition or subtraction of specified numbers of bags or marbles, and multiplication or division of the current collection by a specified integer. Students are introduced to marble bag stories and diagrams. They are shown how to create and solve simple marble bag stories (story problems). They are introduced to standard algebraic notation as a rapid way of writing marble bag stories. As a student works on a problem, the system can show the correspondence between the iconic, English, and standard algebraic representations of the operations and results.
>
> (Feurzeig, 1985, p. 231)

The Marble Bag Microworld described by Feurzeig consists of a display as shown in Figure 14.[2] The student acts within the microworld by clicking the mouse pointer on an appropriate object. For example, the display in Figure 14 was made by clicking on the icons for bags and marbles in the upper-left corner. The microworld provided one bag in the STORY window and the line "Think of a number. X" in the History window. I clicked three times on the bag icon, then clicked DONE, and the microworld mimicked my actions with the line "Multiply by 4. 4X." And so on. Problems posed to students might take the form of, "Construct a story that ends up with the expression 2(4x)+2."

Figure 14 About Here

Solving equations is presented as the reverse process of that shown in Figure 14. To present "equation solving," the computer generates a marble-bags story in English. The computer presents a sentence of the story, the student translates the sentence into a display of marbles and bags, and the computer records the interaction in the History window.

The screen in Figure 15 shows the result of a sequence of such interactions. The computer's instructions were given in the panel underneath the word **Instructions** (middle-left of screen). As I responded to each instruction (by clicking on marble or bag icons in the upper left corner), the English-language instruction, along with its algebraic counterpart appeared in the **History** window. The first instruction was "Think of a number" (the **Story** panel was blank). I clicked on the bag icon, then clicked Done. A

[2] All figures in this section were generated as screen dumps from a prototype copy of the marble bags microworld, which was lent to me graciously by Wally Feurzeig.

marble bag appeared in the **Story** panel; the first line in the History window recorded this action. The next three lines in the **History** window are a record of my responses to the subsequent three sentences in the story. The **Instructions** panel shows the fifth sentence of the story: "Subtract your original number" (to which one would respond by clicking on a marble-bag icon in the **Story** window).

When the story has been told, the microworld says something like, "There are now 37 marbles. How many marbles did I start with?" To determine the computer's initial number of marbles, the student operates on the display of marbles and bags in the **Story** panel. The microworld describes the student's actions in English and in algebraic notation, which ends up producing a history of the story's creation and of its inverse-creation. Feurzeig's intention is that these activities provide a cognitive foundation for students' understanding of operations on equations as ways to "unravel" an equation into an initially unknown value.

Figure 15 About Here

The microworld also has a "balance beam" mode. In this mode the student can operate on one side of an equation or on both sides. The balance tilts toward one side or the other if a student does something that does not maintain equivalence between the two sides. Neither "unraveling" nor the balance beam is a preferred approach. Rather, they are offered as two complementary aspects of equation solving.

The most impressive feature of Feurzeig's project (not just the Marble Bags microworld) is the variety of metaphors it uses to approach the idea of algebra as a language descriptive of operations upon quantities and relationships among quantities. While Feurzeig's team has yet to make the AI side of their programs apparent to an observer, the intelligence of their design decisions is readily apparent.

Conclusion

I began the previous section with an extended discussion of Draper's and Norman's idea of software that provides a user with the sense of having a "first person" experience. The spirit of the pieces in Draper's and Norman's collection is that of providing people with tools. The examples cited differ somewhat from that spirit. While one might think of them as tools for the intellect (Papert, 1980), they were designed with more in mind than as an aid to cognition. Rather, they were designed with the intent of shaping cognition. If their is an "aid" aspect to these projects, it is that the software was designed so as to aid students in coming to think the way the designers intended them to think.

Had I focused on tools per se, I would have had to mention hand-held, symbolic manipulating calculators recently introduced by Hewlett-Packard and soon to be released by ATT. Had I discussed these, I would have argued that, just as with arithmetical calculators in elementary school, these machines are largely irrelevant to mathematics education unless students have appropriate mental models and knowledge structures that enable them to make judicious use of the technology.

The examples given here differ significantly from previous work with intelligent tutoring systems. Developers of ITS's tended to focus on strategic and manipulative skills within quite rigid boundaries of traditional algebra instruction. The focus of the examples presented here is at a much higher conceptual level. Were the developers of the examples presented here to embed their systems within ITS's, however, past ITS research would be very fruitful in terms of the techniques used therein, especially the "issue-oriented" techniques developed by Brown and Burton (1982).

The examples presented here also differ from past approaches that attempted to present mathematics as descriptive of phenomena, or as descriptive of structural features of phenomena. Rather, they attempt to present mathematics as phenomena. In a very real sense, these projects attempt to shape students' experiences so as to foster powerful, yet

intuitive conceptualizations of the subject matter. The relationship between experience and conceptualization is stronger than one might think, especially in regard to mathematics (diSessa, 1983). One could say that, in mathematics, to experience is to conceptualize. The critical issue, from a software design perspective, is to present an environment which allows conceptualizations at a number of levels, and which promotes students' advancement through those levels (Norman, 1979; Rumelhart & Norman, 1978; Thompson, 1985, 1987; Thompson & Dreyfus, in press).

Conceptualizations of subject matter are powerful or weak for very specific reasons. A weak conceptualization allows students to over generalize, under generalize, and do "silly" things. A powerful conceptualization provides both generality and constraints on that generality. For example, it is one thing to know a rule that transforms $a(x-b)$ into $ax - ab$, or that transforming $a(x-b)$ into $ax - b$ is not correct because the result should be $ax - ab$. It is quite another thing to know that transforming $a(x-b)$ into $ax - b$ is nonsense.

An expert is an expert not because she does not make mistakes, but because she notices mistakes soon after they are made. Mistakes violate constraints. But constraints often cannot be rules, for rules are too specific. Rather, constraints are more like metaphors, such as "$(x - b)$ is a single thing, like a bucket." It is inconceivable that experts possess a myriad of "DO NOT" rules for every conceivable mistake, or that they avoid mistakes by always applying rules correctly. It is much more plausible that they possess metaphors——and more broadly, conceptualizations——that allow them to judge quickly the sense or nonsense of something by the way it "fits" with their more metaphoric ideas of what they are doing.

Constraints are of no use to students unless they are internalized, unless the constraints are their constraints. The projects illustrated here attempt to focus students attention in ways that they will construct metaphors, generalizations, and concepts for themselves. To what extent they succeed, and the reasons for success or failure, are questions that need to be researched.

Another aspect that unifies the projects described here, but which has yet to become a research focus, is that of multiple, linked representational systems (Kaput, 1985, 1986). In WPA, students act on representations of problems and those actions have consequences for the expressions describing the quantities in the representation. In McArthur's system, students act on particular expressions or equations and the system fits the results of those actions into its representation of their solution paths. In Marble Bags, students act in one representational system and the results of those actions are reflected in both the system in which the action occurred and in an alternative representational system. In EXPRESSIONS, students use a set of trans-representational operations to effect changes in one representational system by indicating the operand in another system——one in which the structure of the operand is made explicit and visual.

The idea of multiple, linked representational systems appears to be powerful, but we have little idea of the actual effect their use has on students' cognitions. There is preliminary evidence that their use has a positive effect on skill (Greeno et al., 1985; Lesh, 1987; Thompson & Thompson, 1987), but we do not know in any broad detail the nature of students' subjective experiences or how those experiences are reflected in cognition.

Also, we need to look seriously and vigorously at the question of to what extent achieving the aims embodied in these projects makes any practical difference in the mathematical lives of students. We can ask if WPA, for instance, makes students better problem solvers away from the program, or if it in fact succeeds in preparing students to think algebraicly. To answer these questions will not be easy. I tend to think that before we can obtain satisfactory answers, we will need to rethink our goals and objectives of school mathematics. I suspect that we will find that, yes, what students learn is valuable and should be learned, but that what they learn is not assessed on today's standardized tests and is not contained in today's textbooks.

The programs described in the previous section are not “big” AI programs. However, I would argue that the issues they address are actually more significant for learning/teaching algebra (or mathematics, for that matter) than are the issues addressed by many larger projects, especially those aiming at developing intelligent tutors in algebra. This is not to say that these programs cannot benefit from past research on intelligent tutoring systems. Rather, before attempting to create intelligent tutors or intelligent mentors, we must have a better understanding of the nature of students’ experiences with this type of software and a better understanding of how students’ thinking can be affected when using it.

Bibliography

Anderson, J. R., & Skwarecki, E. (September, 1986). The automated tutoring of introductory computer programming. Communications of the ACM, 29(9), 842-849.

Barwise, J., & Etchemendy, J. (1986a). Tarski's World. Stanford, CA: Stanford Board of Trustees. (Distributed by Kinko's Academic Courseware Exchange.)

Barwise, J., & Etchemendy, J. (1986b). Turing's World. Stanford, CA: Stanford Board of Trustees. (Distributed by Kinko's Academic Courseware Exchange.)

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

Brown, J. S., & VanLehn, K. (1981). Repair theory: a generative theory of bugs in procedural skills. Cognitive Science, 4(4), 379-426.

Brown, J., & Burton, R. (1982). Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman & J. Brown (Eds.), Intelligent Tutoring Systems. New York: Academic Press.

Bundy, A. (1983). The computer modeling of mathematical reasoning. New York: Academic Press.

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

Burton, R., & Brown, J. (1976). A tutoring and student modeling environment for gaming environments. In R. Coleman & P. Lorton (Eds.), Computer Science and Education, ACS SIG/CSE Bulletin, 8 (1), 236-246.

Charniak, E., & McDermott, D. (1985). Introduction to artificial intelligence. Reading, MA: Addison-Wesley.

Cronbach, L. J. (1965). Issues current in educational psychology. Monographs of the Society for the Research in Child Development, 30(Serial No. 99), 109-126.

Davis, R. B. (1975). Cognitive processes involved in solving simple algebraic equations. Journal of Children's Mathematical Behavior, 1(3), 7-35.

Davis, R. B., Jockusch, E., & McKnight, C. (1978). Cognitive processes in learning algebra. Journal of Children's Mathematical Behavior, 2(1), 10-320.

Dewey, J. (1964). Experience and education. New York: Collier Books. (Originally published 1945.)

diSessa, A. (1983). Phenomenology and the evolution of intuition. In D. Gentner & A. L. Stevens (Eds.), Mental models. Hillsdale, NJ: Erlbaum.

diSessa, A. (September, 1986). Boxer: A reconstructible computational medium. The automated tutoring of introductory computer programming. Communications of the ACM, 29(9), 859-869.

Draper, S. W., & Norman, D. A. (1986). Introduction. In D. A. Norman, & S. W. Draper (Eds.), User centered system design. Hillsdale, NJ: Erlbaum.

Erlwanger, S. (1973). Benny's conception of rules and answers in IPI mathematics. Journal of Children's Mathematical Behavior, 1(2), 5-27.

Goldberg, A. (1984). Smalltalk-80: The interactive programming environment. Reading, MA: Addison-Wesley.

Greeno, J. (1985). Advancing cognitive science through development of advanced instructional systems. Paper presented at the Annual Meeting of the Amereican Educational Research Association, Chicago, April.

Greeno, J., Brown, J. S., Shalin, V., Bee, N. V., Lewis, M. W., & Yrtolo, M. (1985). Cognitive principles of problem solving and instruction. Abstract of Final Report to the Office of Naval Research.

Feigenbaum, E., & Barr, E. (1981). Handbook of artificial intelligence (Vol. 1). Stanford, CA: Heiristech Press.

Feurzeig, W. (1986). Algebra slaves and agents in a Logo-based mathematics curriculum. Instructional Science, 14, 229-254.

Herscovics, N. & Kieran, C. (1980). Constructing meaning for the concept of equation. Mathematics Teacher, 73, 572-580.
Johnson-Laird, P. N. (1977). Procedural semantics. Cognition, 5, 189-214.
Johnson-Laird, P. N. (1983). Mental models. Cambridge, MA: Harvard University Press.
Kaput, J. (1985). Representation and problem solving: Methodological issues related to modeling. In E. Silver (Ed.), Teaching and learning mathematical problem solving: Multiple research perspectives. Hillsdale, NJ: Erlbaum.
Kaput, J. (1986). Information technology and mathematics: Opening new representational windows. Journal of Mathematical Behavior, 5, 187-207.
Kay, A. (1984, March). Computer software. Scientific American, 52-59.
Kearsley, G. (Ed.) (1987). Artificial intelligence and instruction: Applications and methods. New York: Addison-Wesley.
Kintsch, W., & Greeno, J. (1985). Understanding and solving arithmetic word problems. Psychological Review, 92, 109-129.
Lawler, R. W. (1985). Computer experience and cognitive development. New York: John Wiley & Sons.
Lenat, D. The nature of heuristics (Cognitive and Instructional Science Series CIS-12 [SSL-81-1]). Palo Alto, CA: Xerox PARC.
Lesh, R. (1987). The evolution of problem representations in the presence of powerful conceptual amiplifiers. In C. Janvier (Ed.), Representational systems. Hillsdale, NJ: Erlbaum.
Lesh, R., Post, T., & Behr, M. (In press). Dienes revisited: Multiple embodiments in computer environments. In I. Wirszup (Ed.), Proceedings of the international conference on mathematics education: Organized by the University of Chicago School Mathematics Project. Chicago: University of Chicago Press.
Lewis, C. (1981). Skill in algebra. In J. R. Anderson (Ed.), Cognitive skills and their acquisition (pp. 85-110). Hillsdale, NJ: Erlbaum.
Lewis, M. W., Milson, R., & Anderson, J. R. (1987). Designing an intelligent autoring system for high school mathematics ICAI: The TEACHER'S APPRENTICE project. In G. Kearsley (Ed.), Artificial intelligence and education. Reading, MA: Addison-Wesley.
Matz, M. (1980). Towards a computational model of algebraic competence. Journal of Children's Mathematical Behavior, 2, 95-166.
Matz, M. (1982). Towards a process model for high school algebra errors. In D. Sleeman, & J. S. Brown (Eds.), Intelligent tutoring systems. New York: Academic Press.
McArthur, D. (1985). Developing computer tools to support peforming and learning complex cognitive skills. In D. Berger, K. Pedzek, & W. Ganks (Eds.), Applications of cognitive psychology. Hillsdale, NJ: Erlbaum.
McArthur, D., Stasz, C., & Hotta, J. (in press). Learning problem-solving skills in algebra. Educational Technology, Winter 1987.
McCorduck, P. (1979). Machines who think. San Francisco: Freeman.
Negotia, C. V. (1985). Expert systems and fuzzy systems. Menlo Park, CA: Benjamin Cummings.
Newell, A., & Simon, H. A. (1972). Human problem solving. Englewood Cliffs, NJ: Prentice-Hall.
Norman, D. A. (1980). Cognitive engineering and education. In D. T. Tuma & F. Reif (Eds.), Problem solving and education. Hillsdale, NJ: Erlbaum.
O'Shea, T., & Eisenstadt, M. (Eds.). (1984). Artificial intelligence: Tolls, techniques, and applications. New York: Harper & Row.
Papert, S. (1980). Mindsorms: Children, computers, and powerful ideas. New York: Basic Books.
Resnick, L. B. & Ford, W. W. (1981). The psychology of mathematics for instruction. Hillsdale, NJ: Erlbaum.

Skemp, R. R. (1978). Relational and instrumental understanding. Arithmetic Teacher, 26, 9-15.

Sleeman, D. H. (1982). Assessing competence in basic algebra. In D. H. Sleeman & J. S. Brown (Eds.), Intelligent tutoring systems (pp. 186-199). New York: Academic Press.

Sleeman, D. H. (1984) An attempt to understand students' understanding of basic algebra. Cognitive Science, 8, 387-412.

Sleeman, D. H. (1985). Basic algebra revisited: A study with 14-year olds. International Journal of Man-Machine Studies, 22, 127-149.

Sleeman, D. (1987). MICRO-SEARCH: A data-driven system to help students solve non-deterministic tasks. In G. Kearsley (Ed.), Artificial intelligence and education. Reading, MA: Addison-Wesley.

Sleeman, D. H., & Smith, M. J. (1981). Modeling students' problem solving. Artificial Intelligence, 16 (2), 171-187.

Sowa, J. F. (1984). Conceptual structures: Information processing in mind and machine. Reading, MA: Addison-Wesley.

Stanic, G. M. A., & Kilpatrick, J. (in press). Historical perspectives on problem solving in the mathematics curriculum. In R. Charles & E. Silver (Eds.), Teaching and evaluating mathematical problem solving. Reston, VA: National Council of Teachers of Mathematics.

Thompson, P. (1982). A theoretical framework for understanding young children's concepts of whole number numeration. Unpublished doctoral dissertation, University of Georgia, Athens [DAI 43A: 1868; December, 1982].

Thompson, P. (1985). Experience, learning, and problem solving: Considerations in the design of mathematics curricula. In E. A. Silver (Ed.), Learning and teaching mathematical problem solving: Multiple research perspectives. Hillsdale, NJ: Erlbaum.

Thompson, P. (1987). Mathematical microworlds and intelligent computer-assisted instruction. In G. Kearsley (Ed.), Artificial intelligence and education: Applications and methods. New York: Addison-Wesley.

Thompson, P., & Dreyfus, T. (In press). Integers as transformations. Journal of Research in Mathematics Education.

Thompson, P., & Thompson, A. (1987). Computer presentations of structure in algebra. In A. Bergeron (Ed.), Proceedings of the Eleventh Annual Meeting of the International Group for the Psychology of Mathematics Education. Montréal: University of Quebec.

Van Lehn, K. (1983). On the representation of procedures in repair theory. In H. P. Ginsburg, (Ed.), The development of mathematical thinking. New York: Academic Press.

Wenger, E. (1987). Artificial intelligence and tutoring systems. Los Altos, CA: Morgan Kaufmann.

Winograd, T. (1977). A framework for understanding discourse. In M. A. Just, & P. A. Carpenter (Eds.), Cognitive processes in comprehension. Hillsdale, NJ: Erlbaum.

Winograd, T., & Flores, F. (1986). Understanding computers and cognition: A new foundation for design. Norwood, NJ: Ablex.

Winston, P. (1984). Artificial intelligence(second edition). Reading, MA: Addison-Wesley.

Figure 1

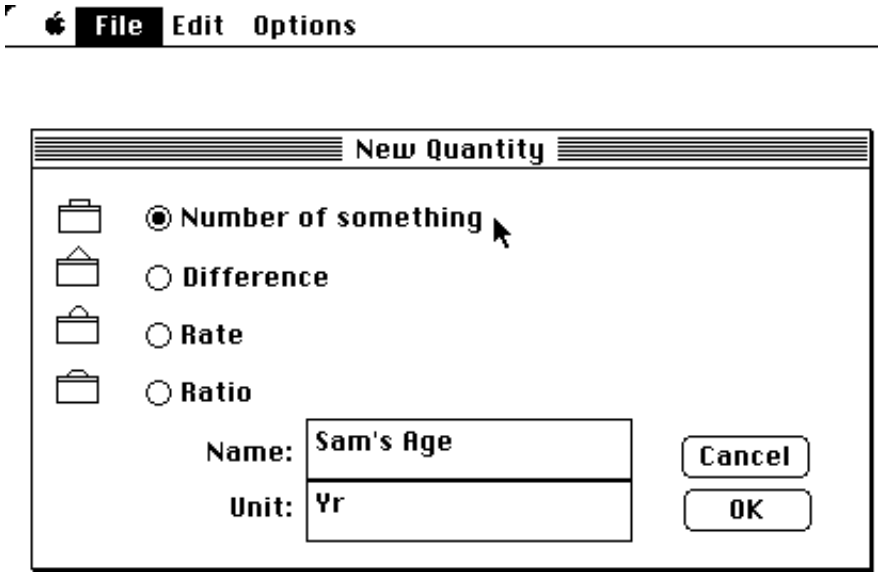

File
Edit
Options
New Quantity
Number of something
Difference
Rate
Ratio
Name:
Sam's Age
Unit:
Yr
Cancel
OK

Figure 2

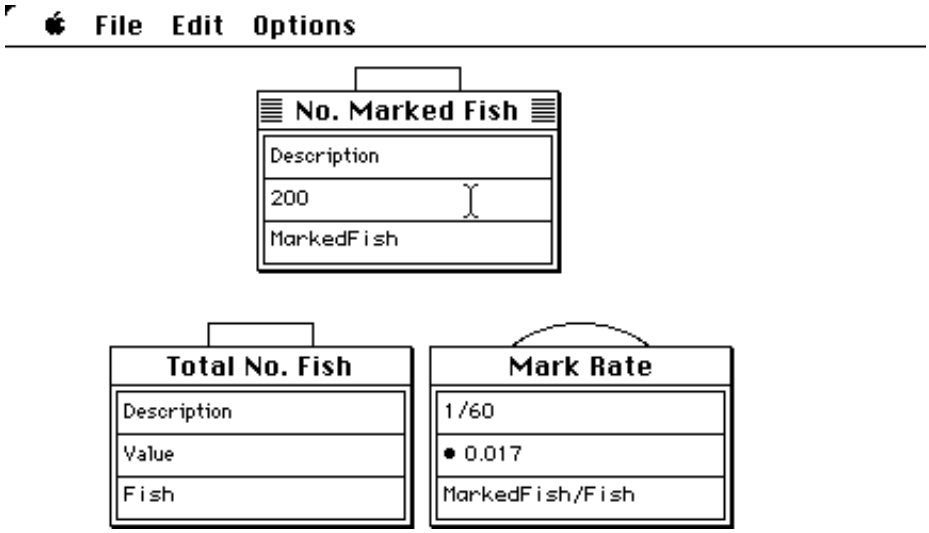

File Edit Options
No. Marked Fish
Description
200
MarkedFish
Total No. Fish
Description
Value
Fish
Mark Rate
1/60
0.017
MarkedFish/Fish

Figure 3

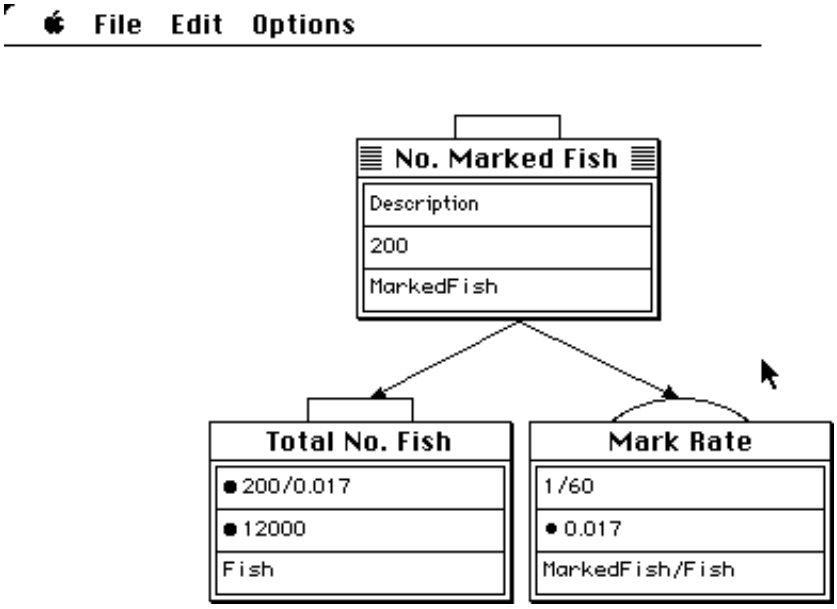

File Edit Options
No. Marked Fish
Description
200
MarkedFish
Total No. Fish
200/0.017
12000
Fish
Mark Rate
1/60
0.017
MarkedFish/Fish

Figure 4

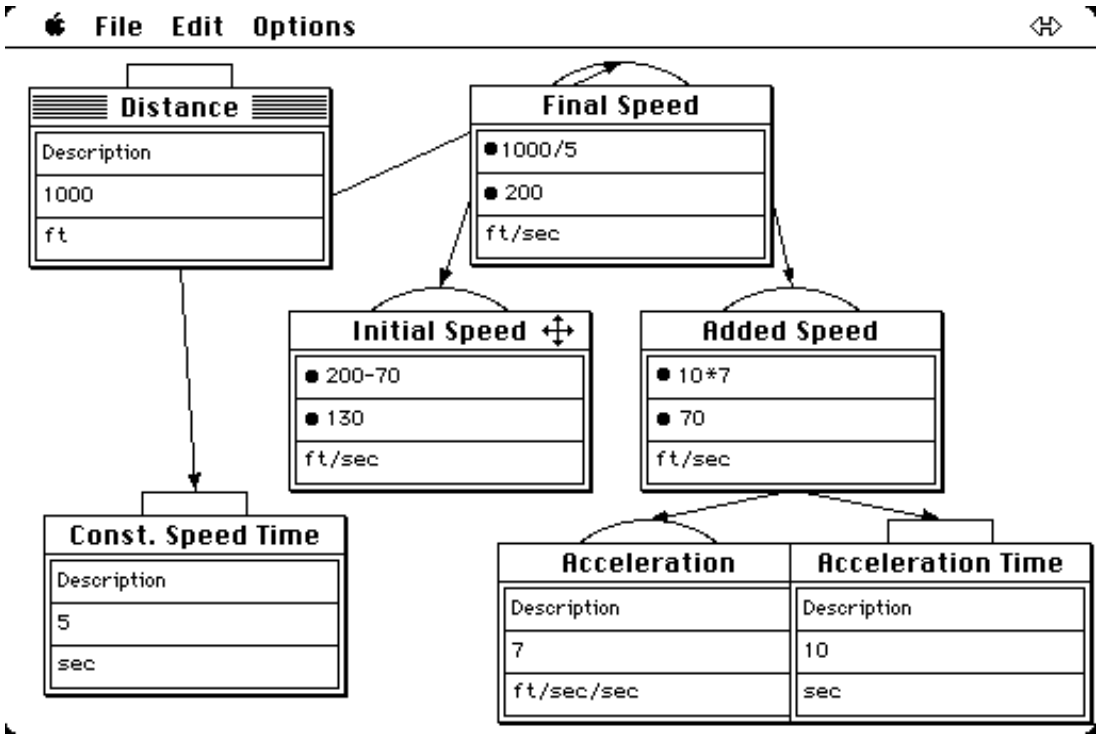

File Edit Options
Distance
Description
1000
ft
Final Speed
1000/5
200
ft/sec
Initial Speed
200-70
130
ft/sec
Added Speed
10*7
70
ft/sec
Const. Speed Time
Description
5
sec
Acceleration
Description
7
ft/sec/sec
Acceleration Time
Description
10
sec

Figure 5

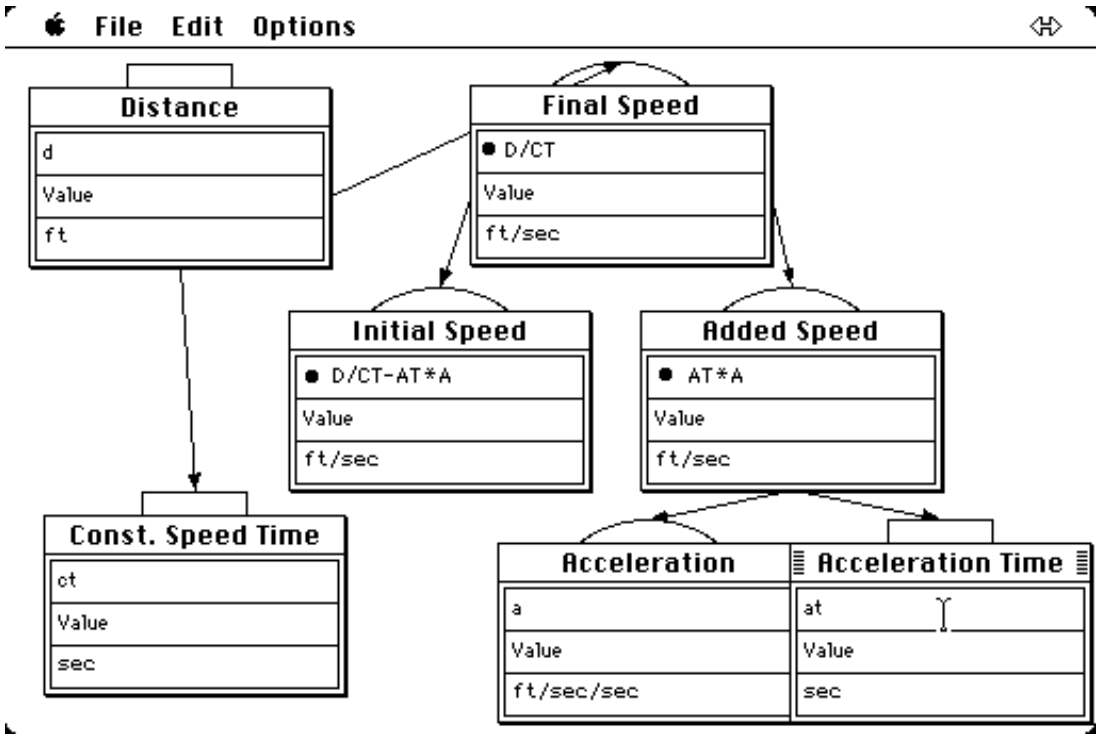

File Edit Options
Distance
d
Value
ft
Final Speed
D/CT
Value
ft/sec
Initial Speed
D/CT-AT*A
Value
ft/sec
Added Speed
AT*A
Value
ft/sec
Const. Speed Time
ct
Value
sec
Acceleration
a
Value
ft/sec/sec
Acceleration Time
at
Value
sec

Figure 6

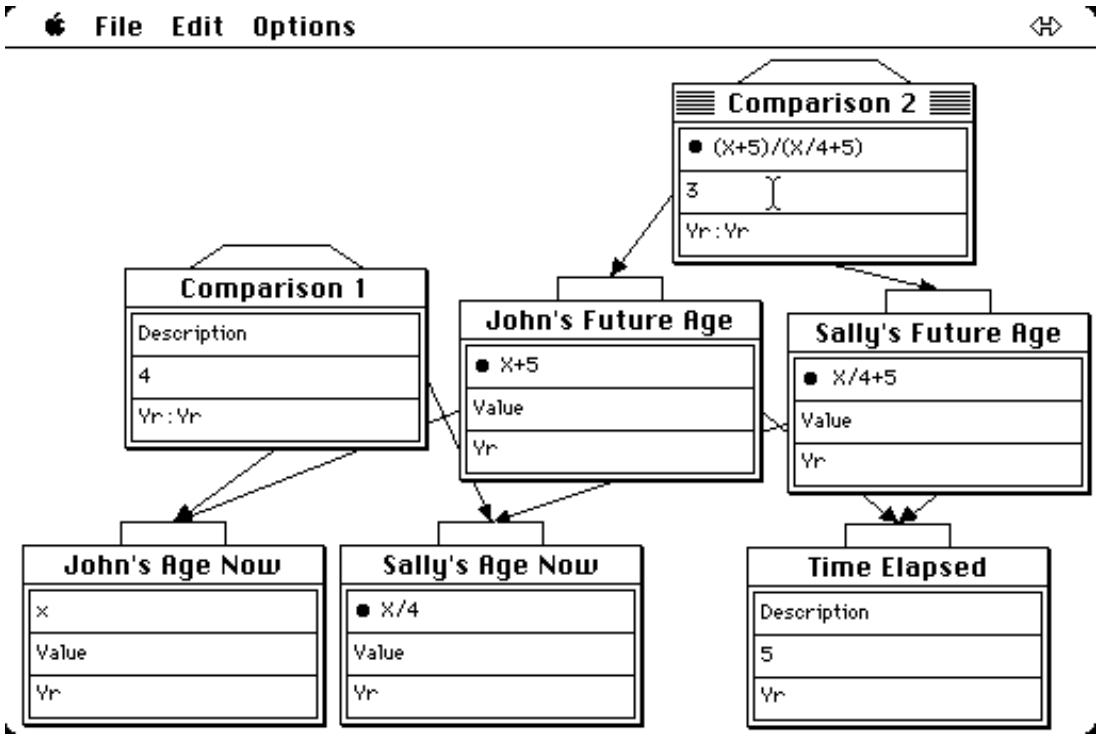

File Edit Options
Comparison 2
(X+5)/(X/4+5)
3
Yr : Yr
Comparison 1
Description
4
Yr : Yr
John's Future Age
X+5
Value
Yr
Sally's Future Age
X/4+5
Value
Yr
John's Age Now
x
Value
Yr
Sally's Age Now
X/4
Value
Yr
Time Elapsed
Description
5
Yr

Figure 7

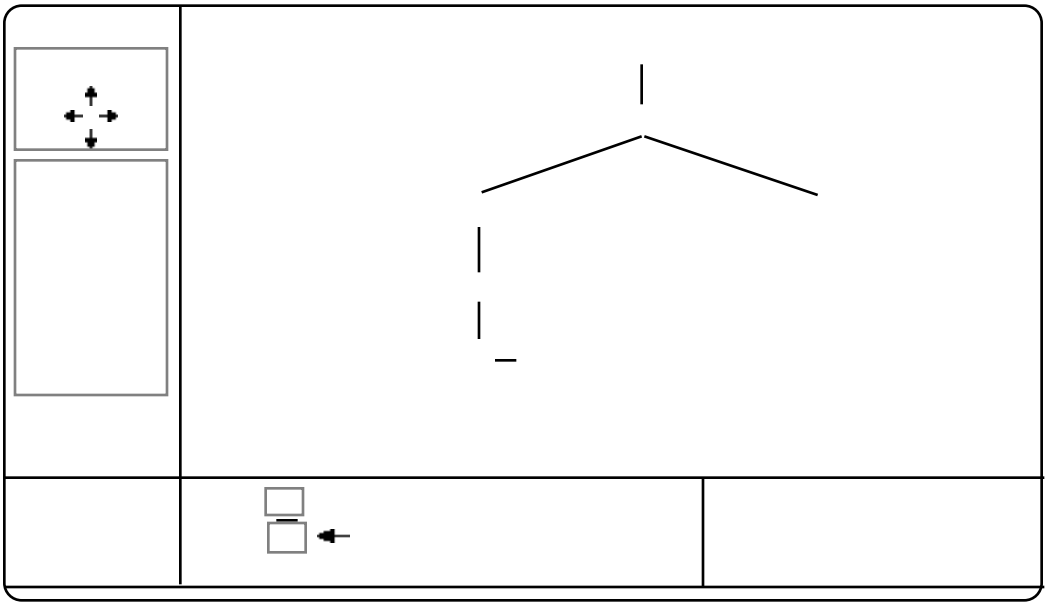


Figure 8

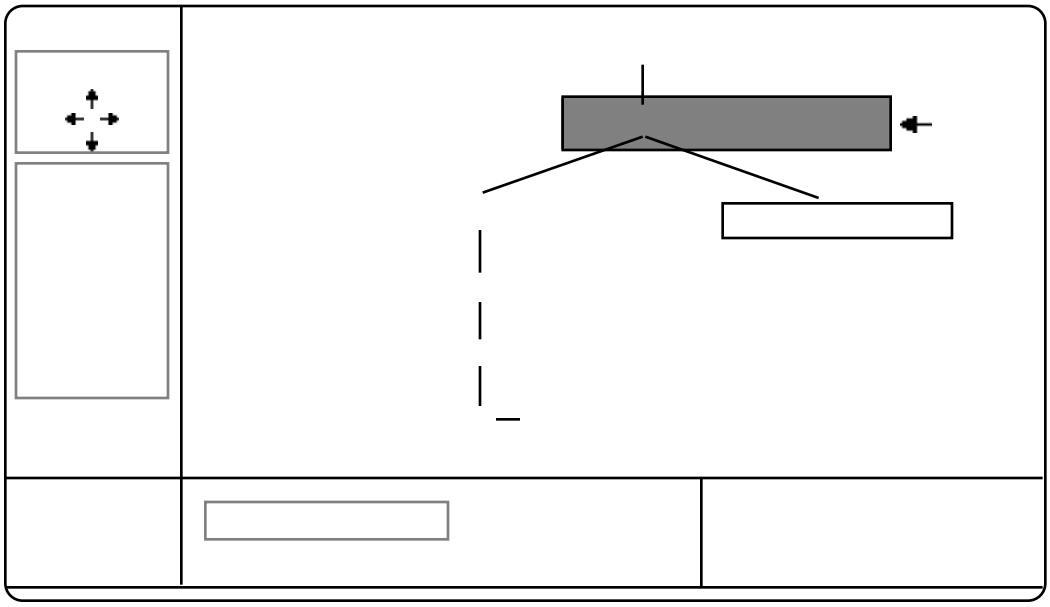


Figure 9

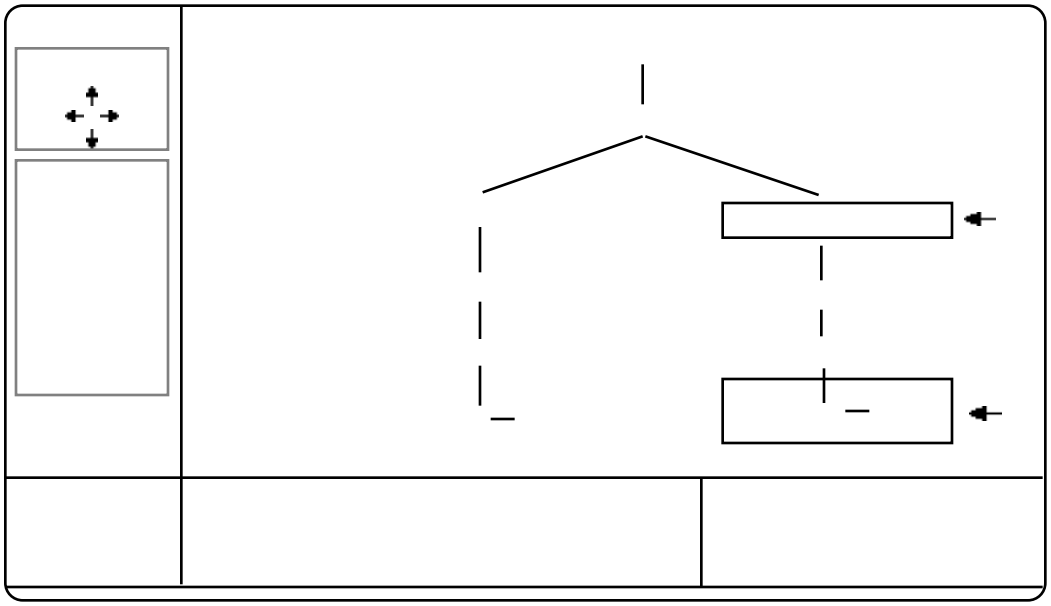


Figure 10

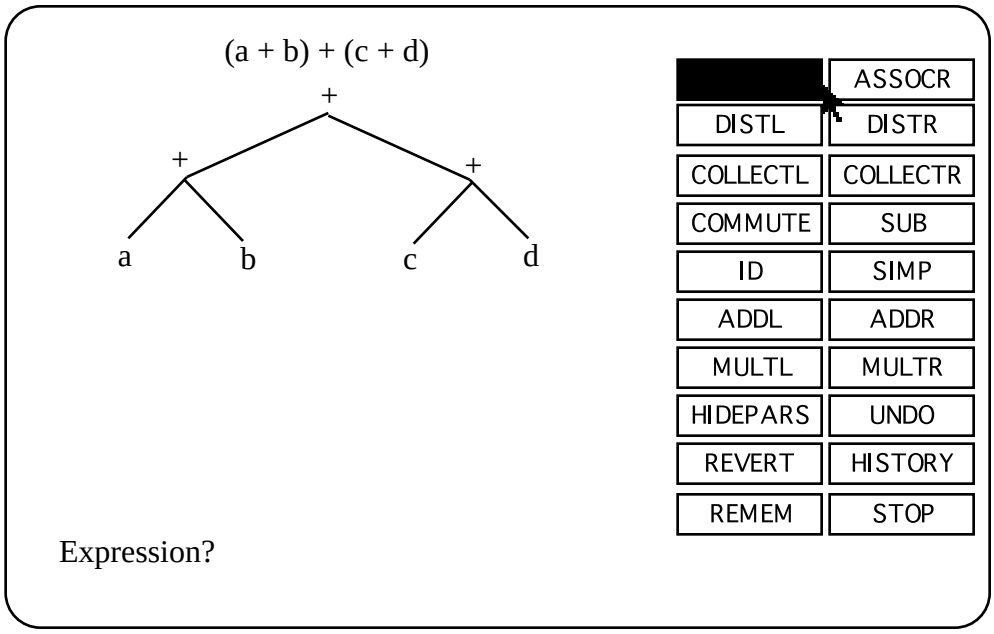

(a + b) + (c + d)
+
+
+
a
b
c
d
ASSOCR
DISTL
DISTR
COLLECTL
COLLECTR
COMMUTE
SUB
ID
SIMP
ADDL
ADDR
MULTL
MULTR
HIDEPARS
UNDO
REVERT
HISTORY
REMEM
STOP
Expression?

Figure 11

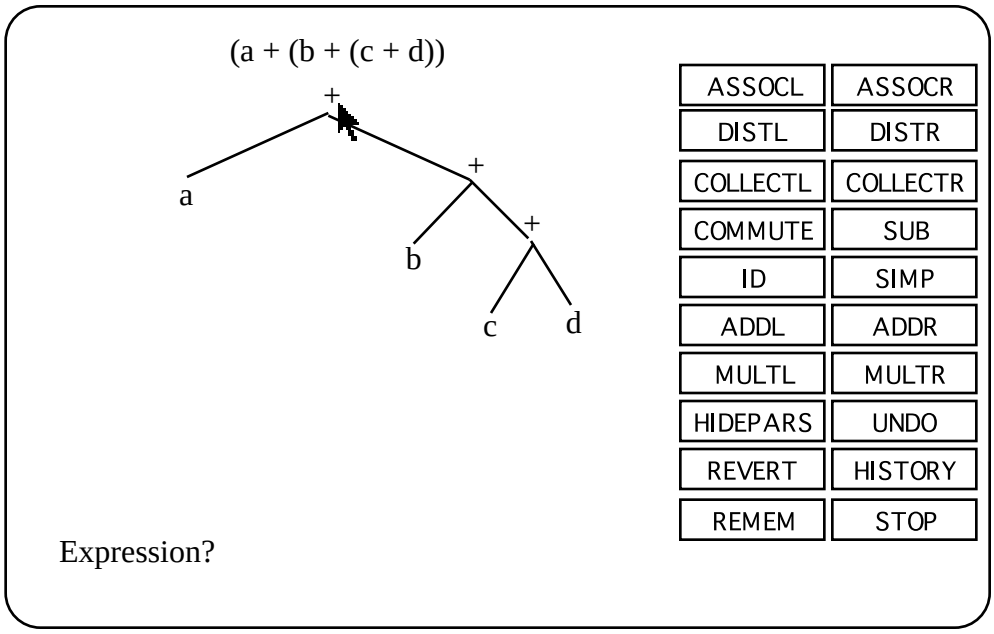
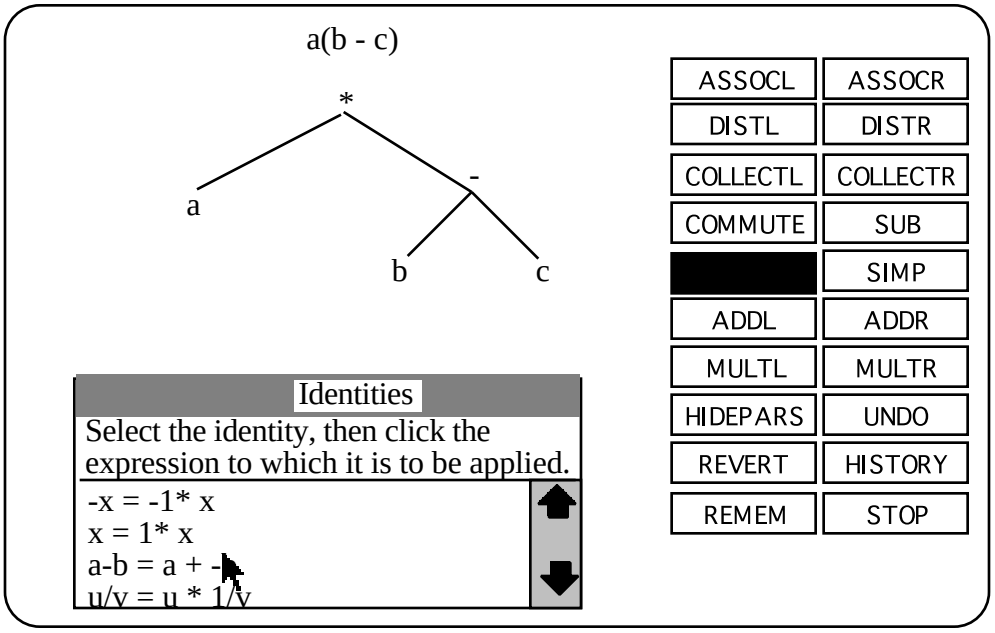

(a + (b + (c + d))
+
a
+
b
+
c
d
ASSOCL
ASSOCR
DISTL
DISTR
COLLECTL
COLLECTR
COMMUTE
SUB
ID
SIMP
ADDL
ADDR
MULTL
MULTR
HIDEPARS
UNDO
REVERT
HISTORY
REMEM
STOP
Expression?

Figure 12


a(b - c)
*
a
-
b
c
ASSOCL
ASSOCR
DISTL
DISTR
COLLECTL
COLLECTR
COMMUTE
SUB
SIMP
ADDL
ADDR
MULTL
MULTR
HIDEPARS
UNDO
REVERT
HISTORY
REMEM
STOP
Identities
Select the identity, then click the expression to which it is to be applied.
-x = -1* x
x = 1* x
a-b = a + -
u/v = u * 1/v

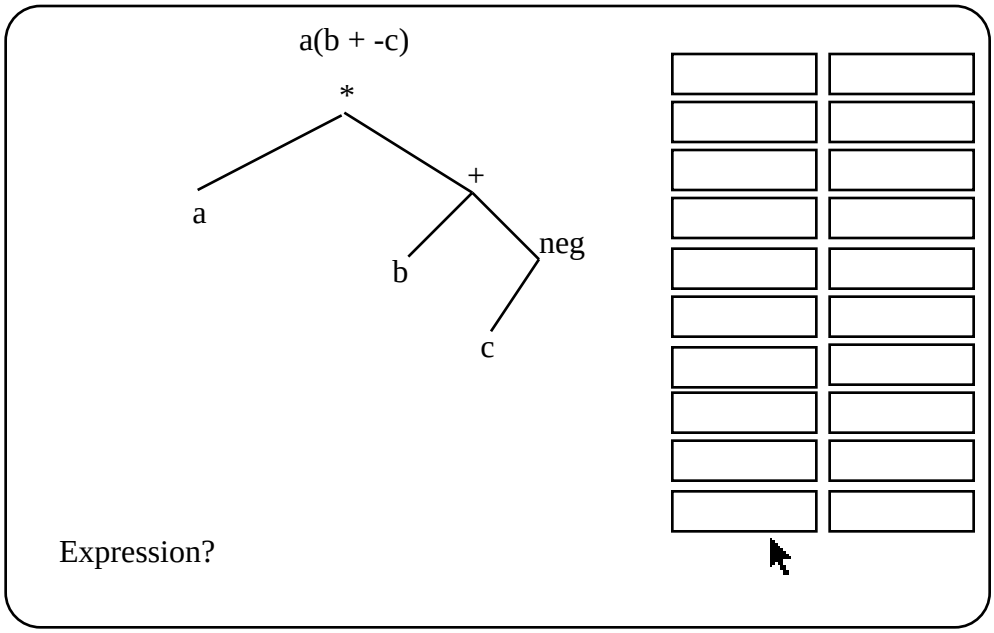

a(b + -c)
*
a
+
b
neg
c
Expression?

Figure 13

File Windows Options

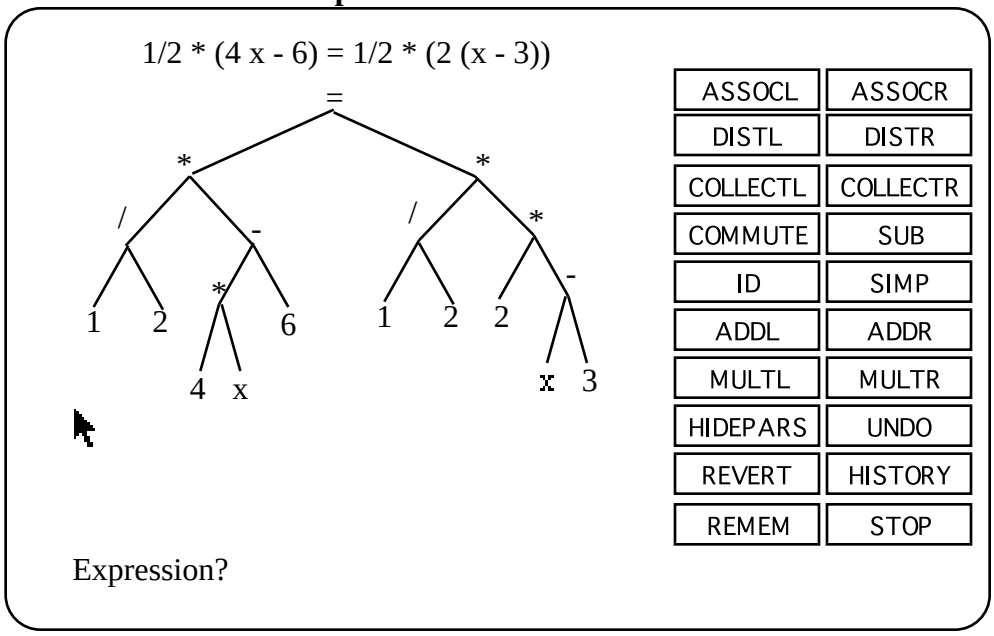


Figure 14

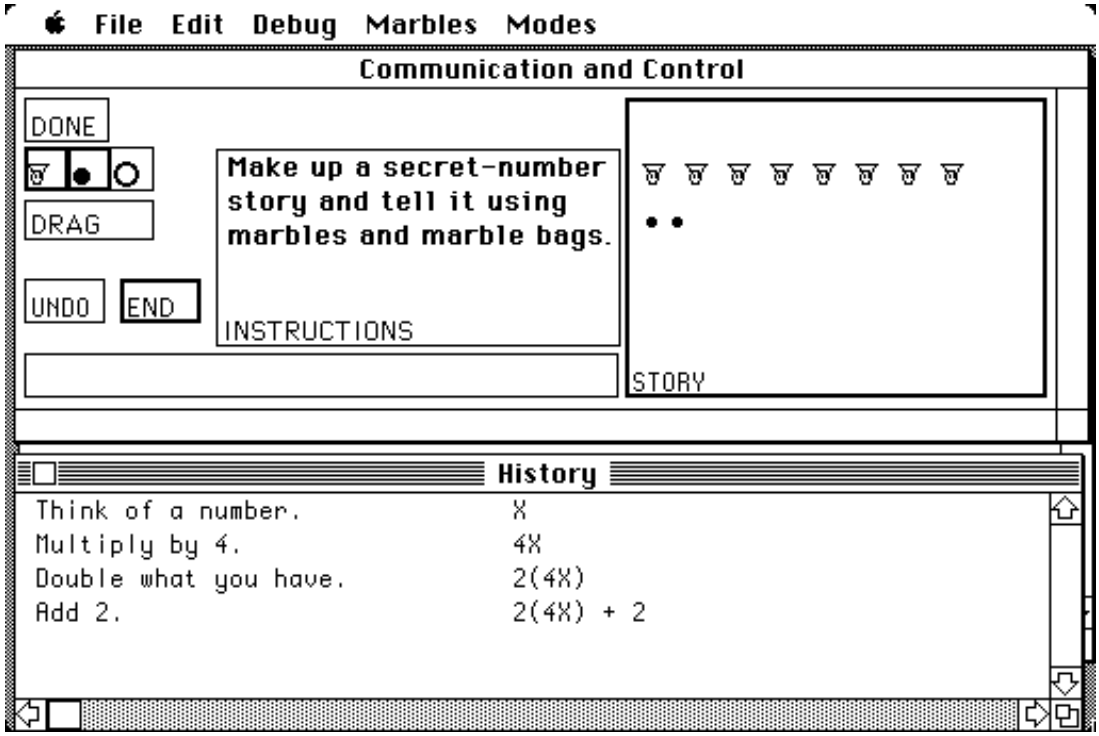
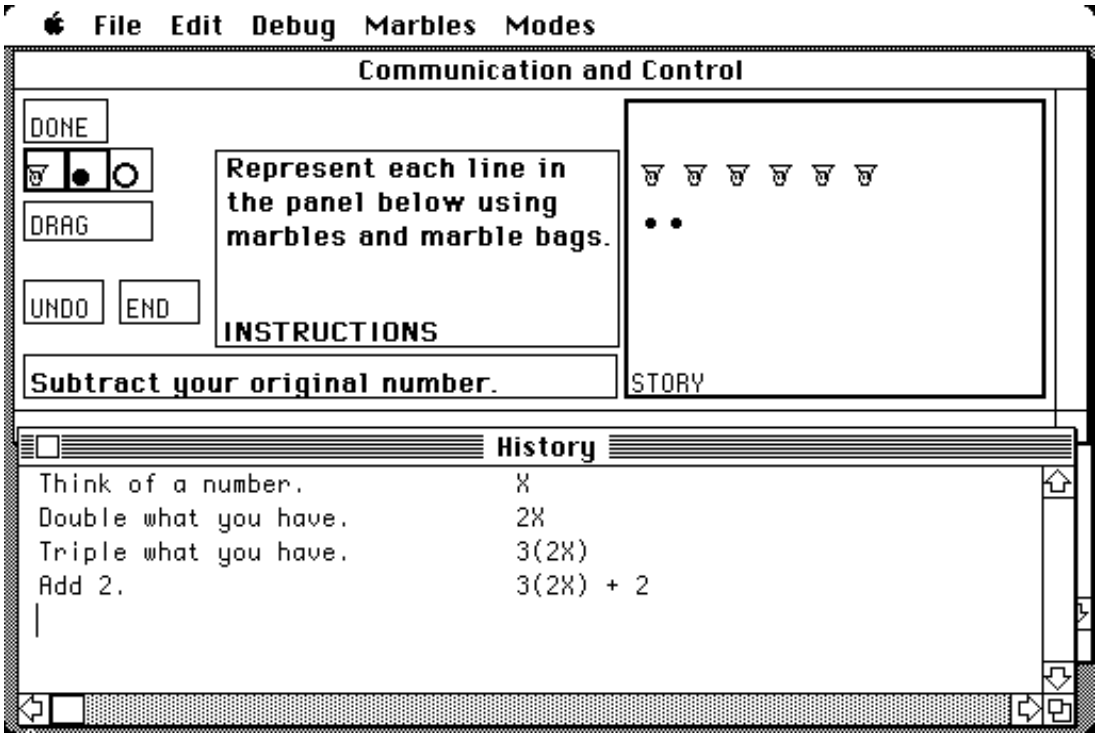

File Edit Debug Marbles Modes
Communication and Control
DONE
DRAG
UNDO
END
Make up a secret-number story and tell it using marbles and marble bags.
INSTRUCTIONS
STORY
History
Think of a number. X
Multiply by 4. 4X
Double what you have. 2(4X)
Add 2. 2(4X) + 2

Figure 15


File Edit Debug Marbles Modes
Communication and Control
DONE
DRAG
UNDO
END
Represent each line in the panel below using marbles and marble bags.
INSTRUCTIONS
Subtract your original number.
STORY
History
Think of a number. X
Double what you have. 2X
Triple what you have. 3(2X)
Add 2. 3(2X) + 2