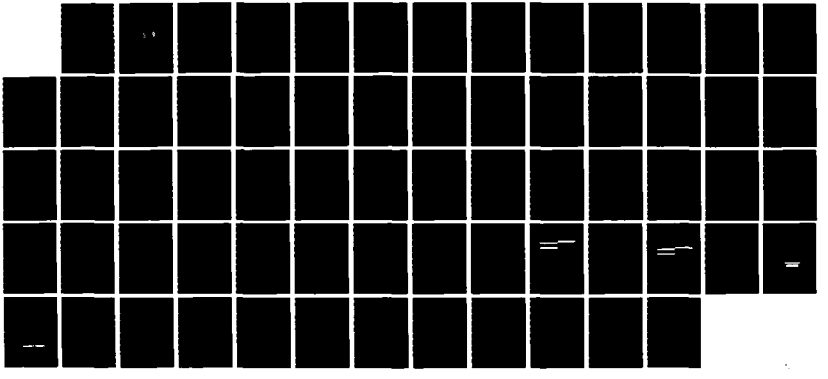
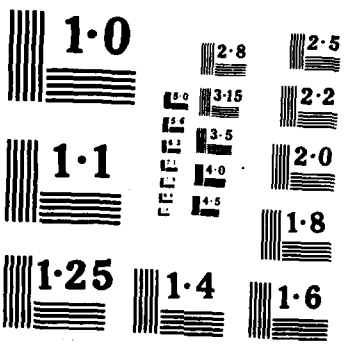
AD-A170 044 COGNITIVE PRINCIPLES OF PROBLEM SOLVING AND INSTRUCTION 1/1
(U) CALIFORNIA UNIV BERKELEY J G GREENO ET AL. 1982
N00014-82-K-0613

UNCLASSIFIED F/G 5/10 NL




1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5

(12)

Final Report:

Cognitive Principles of Problem Solving and Instruction

James G. Greeno, University of California, Berkeley,
John Seely Brown and Carolyn Foss, Xerox Palo Alto Research Center,
Valerie Shalin, Nancy V. Bee, Matthew W. Lewis,
and Theresa M. Vitolo, University of Pittsburgh

Project NR154-497

This research was supported by the Office of Naval Research.
Contract SFRC N00014-82-K-0613.

Approved for public release; distribution unlimited.

Cognitive Principles of Problem Solving and Instruction[1]

James G. Greeno, University of California, Berkeley,

John Seely Brown and Carolyn Foss, Xerox Palo Alto Research Center,

Valerie Shalin, Nancy V. Bee, Matthew W. Lewis,

and Theresa M. Vitolo, University of Pittsburgh

Abstract

Research in this project studied cognitive processes involved in understanding and solving problems used in instruction in the domain of mathematics, and explored implications of these cognitive analyses for the design of instruction. Three general issues were addressed: knowledge required for understanding problems, knowledge of the conditions for applying problem-solving operators, and strategic knowledge for problem solving. Computational systems were developed in which graphical displays present explicit information to students that allows them to reflect on processes that are usually implicit, construct representations of problem information, specify assignments of variables, and learn from strategic errors. Research is described that provided analyses of cognitive processes on which instructional designs were based, and preliminary empirical results from use of the systems are presented.

[1]Final report of Project NR 154-497, supported by the Office of Naval Research Contract SFRC N00014-82-K-0613. Some of the results were presented in a symposium: "Advanced computer system applications in education: The next generation of educational technology research and development" at the American Educational Research Association meeting, Chicago, April 1985. Proceedings of that symposium will be published in Machine Mediated Learning.

1. Introduction

This report describes three projects involving cognitive anlayses of instructional tasks coupled with development of instructional systems. The overall objective of the research was to extend cognitive theory by studying cognitive processes involved in understanding and solving problems used in instruction in mathematics, and to explore implications of these cognitive analyses for the design of instruction.

In the past ten years, methods of cognitive science have been used to provide theoretical and empirical analyses of performance of several tasks used in technical training and school instruction. These analyses have made important contributions to fundamental cognitive theory as well as a better understanding of knowledge and skill that students must acquire to succeed. New results about strategic knowledge in specific task domains have been provided by analyses of problem solving in high school geometry (Anderson, 1982; Greeno, 1978). New characterizations of cognitive skill and learning and understanding of procedures have been provided in analyses of elementary arithmetic (Brown & Burton, 1980; Resnick, 1983; VanLehn, 1983). Important findings about knowledge for understanding problems and forming representations that support effective problem-solving activity have been obtained in studies of solving problems in basic electronics (Riley, 1984), word problems in elementary arithmetic (Briars & Larkin, 1984; Kintsch & Greeno, 1985; Riley, Greeno & Heller, 1983) and basic physics (Chi, Feltovich & Glaser, 1981; Larkin, McDermott, Simon & Simon, 1980).

Further scientific advances have been achieved by using cognitive principles of performance in designing instructional materials. In addition to the practical significance of improvied training, fundamental empirical and theoretical advances can be provided in the context of instructional design and development. Examples include the intelligent diagnostic system DEBUGGY (Burton, 1982), whose development required articulating a

space of variations for a cognitive procedure and a method for systematically searching in that space to diagnose the cognitive basis for a student's errors. Another example is WEST (Burton & Brown, 1982), in which an analysis of cognitive skill and strategy was developed as a set of issues that provide the variables used in modelling student knowledge and analyzing performance on problems. Instruction in general heuristics of problem solving in mathematics (Schoenfeld, 1979), and intelligent tutoring systems for geometry and LISP programming (Anderson, Boyle, Farrell, & Reiser, 1984) include analyses of strategic knowledge for solving problems and articulate representations of search processes. Instruction in representation of physics problems (Heller & Reif, 1984) provides an aanalysis of knowledge for representing information in problems that is significantly different from the empirical analysis that would be obtained from observations of experts, but which shows how, at the level of student capability, it is possible to represent problem information completely and correctly for the purpose of solving the problems. An instructional system in basic physics called Dyna-Turtle (diSessa, 1982) provided an analysis of stages in the acquisition of understanding of Newtonian motion, including implicit conceptualization of velocity as a vector.

The three research projects that we have conducted involve different aspects of knowledge that are important in knowledge for instructional tasks and that pose somewhat different kinds of theoretical and instructional problems. The first project studied the process of understanding problems. Building on earlier analyses of understanding problems in elementary arithmetic, we examined more advanced problems involving more than one step and operations of multiplication and division. A characterization of the patterns of information needed to understand the domain of problems was developed, its main distinctions were supported in an empirical study of problem difficulty, and a computational system was developed that enables students to construct graphical representations of information patterns in problems.

The second project studied aspects of basic skill in a task of transforming expressions in arithmetic. A cognitive analysis of knowledge required for algebra raised an issue that has not been discussed previously about the relation between requirements of new procedural domains. The transition from arithmetic to algebra includes learning new structures of procedures, as well as learning procedures that use new data types. We developed a task to provide instruction in the structure of transformation tasks, using expressions in arithmetic. An empirical study of learning of this task provided new results about the processes involved in solving problems involving choice of operators to transform symbolic expressions. These results were used in the design of components of an intelligent tutoring system for the task. The empirical results were used in the formulation of cognitive issues for analysis of performance and maintaining a model of student knowledge.

The third project studied strategic decision-making in solution of algebraic equations. An earlier analysis of strategies in algebra (Bundy, 1975) was used in designing a system that focuses students' attention on operators at general goals, rather than on mechanical operations. This was accomplished by providing two resources. First, a facility is provided that performs symbolic operations that the student selects, thereby allowing the student to attend more fully to higher-level goals. Second, a display is provided that traces the student's work on the problem; this enables the student to examine retrospectively the sequence of decisions that he or she made in solving the problem, and evaluate the decisions in light of their consequences for problem-solving success. Significant new understanding of the problem-solving process was obtained in developing the system, involving alternative characterizations of problem-solving operators at different levels of generality. The requirement of presenting a usable, articulate system led to theoretical insights about distinctions between different operators, and formulation of global operators that can be considered easily by students in planning and in reflective learning.

2. Knowledge for Representing Problems

Analyses of cognitive processes in solving word problems has provided progress in the integration of two large bodies of cognitive theory: the theory of language understanding and the theory of problem solving. Texts of problems are understood, and general principles of text comprehension apply. Problems are solved, requiring choices of operations to achieve problem goals, and these processes are consistent with general principles in the theory of problem solving. The task is distinctive because text processing and problem solving are both involved in an integrated way; the activity of problem solving is based on the information structure that results from language understanding.

Previous Analyses. Models developed earlier provide hypotheses about the kinds of information structure that successful problem solvers construct when they understand word problems in elementary arithmetic. As an example, consider Figure 1, which shows a semantic network corresponding to one hypothesis about the meaning of a problem: "Jay had some books; then he lost three of them; now he has five of them left. How many books did Jay have in the beginning?"

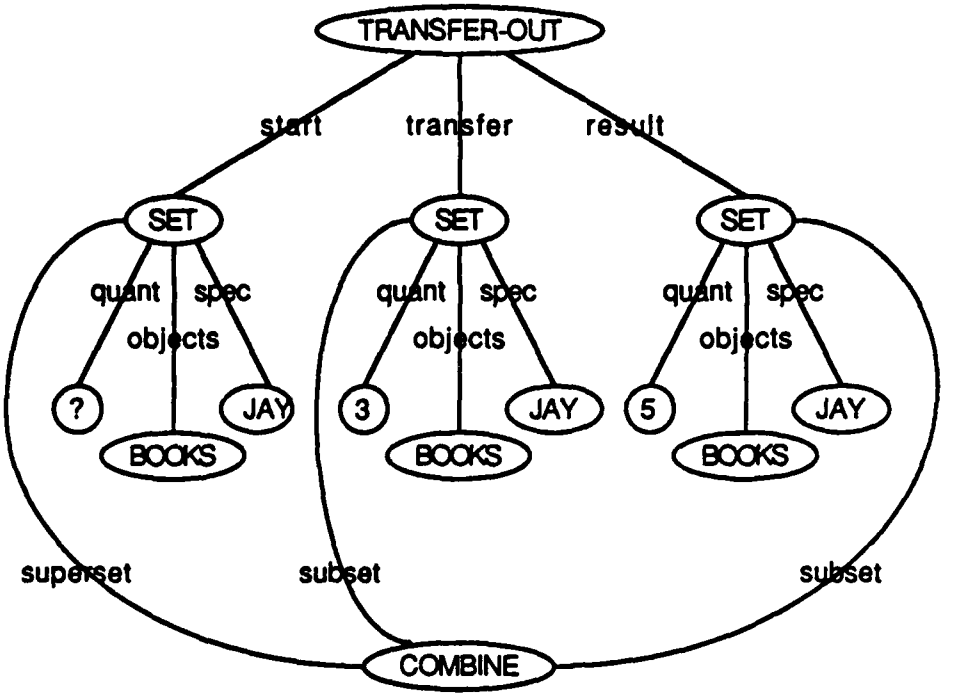


Figure 1. Hypothetical cognitive representation of a problem.

Models developed by Riley, Greeno, and Heller (1983) and by Kintsch and Greeno (1985) assume that problems are represented using schematic knowledge that organizes information into structures corresponding to sets and higher-order structures involving relations between among sets. In the example of Figure 1, sets are represented because of the phrases "some books," "three books," and "five books." Representations of sets are based on a schema that includes *quantity*, *objects*, and *specification* as its slots. In the example, all the sets involve books that are associated with Jay, so the objects and specifications are all the same. The quantities are the numbers given for the sets. The relations among the sets are based on schemata that correspond to different patterns of quantitative interaction. The example problem describes a situation where some objects are removed from a set, which the model recognizes as an instance of its TRANSFER-OUT schema. Other schemata are TRANSFER-IN, COMBINE, and COMPARE. The

TRANSFER-OUT schema has slots for a *start* set, a *transfer* set, and a *result* set. The representations of the three sets in the problem are fit into the structure.

The use of the COMBINE schema in Figure 1 illustrates an interaction between knowledge for representing problem information and knowledge for solving the problem. Based on observations of children solving problems, it was hypothesized that many children lack knowledge for linking the information in the TRANSFER-OUT schema with problem-solving operators when the unknown *quantity* is in the *start* set. The solution then requires finding another schema to represent relatiions between the sets, and the model adds the structure of subsets and a superset provided by the COMBINE schema. The model includes a link from the relations in this structure to the problem-solving operator of combining the quantities in subsets to find the quantity of the superset, and this is done to obtain the answer.

New Analysis of Quantitative Patterns. In the project that this paper reports, an analysis was developed for more complex problems, including multiplication and division operators and requiring more than one step of calculation. An example is:

> *The charge for parking is $.65 per hour. Ms. Jones parked for four hours. How much change did she get from $10.00?*

The model that Riley et al. (1983) and Kintsch and Greeno (1985) developed for simpler problems could be extended easily to apply to these slightly more complex problems. A natural extension would be to have a version of the COMBINE schema for the multiplicative case, with the *number of parts*, the *amount in each part*, and the *total amount* as the slots, and a multiplicative version of COMPARE for situations with two quantities that are related by a multiplicative factor, (e.g., Jay's age, Kay's age, and a relation such as Kay being three times as old as Jay). Problems requiring more than one step would be represented as networks formed with more than one schema. For example,

the parking-lot problem above would use the multiplicative COMBINE schema for the relation between four hours, $.65 for each hour, and the total amount charged, and it could use the additive COMPARE schema for the $10.00 that was paid, the total amount charged, and the difference between these amounts.

An analysis conducted in this project by Shalin and Bee (1985a) uses somewhat different theoretical concepts, for two reasons. First, there are classes of problems that are solved well by students that are problematic for the kind of schema-based model that has been hypothesized for the domain of word problems. An example is the following (cf. Nesher & Katriel, 1977):

> *During the day, 27 children went into the museum and 19 children went out. How many children went through the museum door?*

The problem requires use of the COMBINE schema, and knowledge could be added to the earlier models to activate that schema for problems like this, but the variety of problems that would require special additions to the model is quite large, and it seems implausible that students must acquire knowledge of all the special cases. We concluded that representational knowledge for these problems probably is based on a different principle, involving more general features of problem information than were captured by the schemata of earlier analyses.

The second reason for choosing a different assumption was instructional. Development of our analysis was connected to a goal of designing an instructional system that can provide training for improved skill in solving word problems. We wanted to develop a computer-based graphics system that would display information in problem representations so that students can see the patterns that they need to recognize in the meanings of problem texts. A system that included the labels of schemata such as ADD-COMBINE and MULTIPLY-COMPARE could have been designed, but we judged that

this might be more cumbersome than necessary and interfere with students' understanding of the essential relations in the problems. Whether this is so is, of course, an empirical question, but we chose to design a system that involved a simpler set of visual distinctions. The formal analysis that was conducted matches the concepts that are used in a system of graphical representation that we developed for instruction.

In Shalin and Bee's (1985a) analysis, a problem is characterized as a set of quantities. Several of the distinctions made in this analysis were discussed earlier by Schwartz (1976). A quantity is a distinguished amount of some kind, such as the number of a set or the measure of some physical substance. For example "five books" and "four hours" are quantities, but "five" and "four" are not. The analysis includes a theory of semantic types corresponding to quantities in problems. For the domain of problems in the analysis, there are four types: extensive, intensive, difference, and factor. The analysis also includes a theory of composition of quantities. For example, two extensive quantities can be combined to form another extensive quantity, or an extensive quantity and an intensive quantity can be combined to form an extensive quantity. The pattern of information in a problem is a set of quantities arranged in a network that shows how some of the quantities are the results of combining other quantities.

An extensive quantity is a simple amount, such as the number of members of a set or the number of units of some substance. If Jay has six books and Kay has nine books, then Jay's *six books* and Kay's *nine books* are both extensive quantities. A difference is an additive relation between two quantities of the same type; for example, Kay has three more books than Jay, and the quantity *three more books* is a difference. An intensive quantity is an amount that is the same in each of several units. For example, if each of Jay's books has eight chapters, then *eight chapters per book* is an intensive quantity. A factor is a unitless quantity that relates two other quantities that have the same units. For example, Jay has two-thirds as many books as Kay, and *two-thirds* is a factor.

The analysis specifies rules of composition for quantities. The basic compositional unit involves three quantities, two of which are composed to form the third. The compositions are constrained by the units of the quantities. Operations on numbers are determined by the types and units of the quantities in the composition.

Table 1

Additive Compositions

	Extensive	Difference
Extensive	Extensive	Extensive
Difference		Difference

Table 1 shows the additive compositions of extensive quantities and differences. In all of these compositions, the number of the combination is the sum of the numbers of the components. If the components are given, the combination is found by addition. If the combination and one of the components are given, the other component is found by subtraction.

In one kind of composition, two extensive quantities are combined to form another extensive quantity. When the quantities are cardinalities of sets, the units may all be the same, e.g., *two books* combined with *three books* resulting in *five books*. The units of the combination may also be a conjunction or superset of the units of the components, e.g., *four science books* combined with *two history books* result in *six science and history books*, or *three boys* with *five girls* resulting in *eight children.*

A composition with three extensive quantities can also involve measures of continuous variables, such as distances or masses of objects. Compositions of these quantities correspond to physical concatenations of objects or events, such as placing two objects on a scale or moving an object from one point to a second point and then to a third point. The units of these quantities are constrained to be commensurate. An example is: "Jay's history books weigh seven pounds and his science books weigh 19 poounds; his history and science books weigh 26 pounds altogether."

A second kind of composition has two extensive quantities and a difference. The units of the difference are a comparison between the units of the two extensive quantities. One example is: "Jay has four science books and two history books; he has two more science books than history books," where *four science books* and *two history books* are extensive quantities, and *two more science books than history books* is their difference. Another example is: "Jay has six books; Kay has three more books than Jay; Kay has nine books." An example involving continuous quantities is: "Jay's history books weigh seven pounds; his science books weigh 19 pounds; his science books weigh 12 pounds more than his history books.

Differences can also be composed additively to form another difference, as in "Kay has three more books than Jay, and El has four more books than Kay; therefore, El has seven more books than Jay." The constraint on relations is that the differences must involve three ordered quantities; one combined quantity is the difference between the smallest and the middle quantities, the other combined quantity is the difference between the middle and the largest quantities, and the combination quantity is the difference between the smallest and the largest quantities.

Table 2 shows multiplicative compositions. The number of a combination is the product of the numbers of the components, or the number of a component is obtained by dividing the number of the combination by the other component.

Table 2

Multiplicative Compositions

	Extensive	Difference	Intensive	Factor
Extensive	Extensive	Difference	Extensive	Extensive
Difference		∅	Difference	Difference
Intensive			Intensive	Intensive
Factor				Factor

The simplest compositions involve factors, which combine with other types of quantities to form combinations of the same type as the quantities they are combined with. An example involving two extensive quantities and a factor is, "Jay has five books; Kay has twice as many books as Jay; Kay has ten books," where the units are *five books, twice as many books,* and *ten books*. The units of the extensive quantities may be different, though commensurable, as with "Kay has two-thirds as many history books as science books; she has six science books; therefore she has four history books," where the quantities are *six science books*, *two-thirds as many history books as science books,* and *four history books.*

Factors can also combine with differences, for example: "The difference between Jay's and Kay's heights is three times the difference between El's and Em's heights; Jay is six inches taller than Kay; and El is two inches taller than Em." Factors can combine with intensive quantities, for example: "Car A gets 1.2 times as many miles per gallon as Car B; Car B gets 25 miles per gallon; and Car A gets 30 miles per gallon." Two factors can be

combined to form another factor; for example: "Kay has three times as many books as Jay, and El has one-half as many books as Kay; and El has three-halves as many books as Jay."

Another kind of composition has two extensive quantities and an intensive quantity, with the intensive quantity and one extensive quantity as the components and the other extensive quantity as their combination. Units of the intensive quantity are a relation denoted by "per" between the units of the two extensive quantities. An example is "Jay has six books; each book has eight chapters; Jay's books have 48 chapters altogether." The intensive quantity is *eight chapters per book*. This is combined with *six books*, and the combined quantity is *48 chapters*. An example involving continuous quantities is: "Kay rode her bicycle for 8 miles; her speed was 15 miles per hour; she rode for 32 minutes," where the component quantities are *15 miles per hour* and *32/60 hours*, and the combination is *8 hours*.

An intensive quantity can be combined multiplicatively with a difference, and their combination is another difference, for example: "There are five cats per sack; and Jay has three more sacks than Kay; therefore, Jay has 15 more cats than Kay," where the components are *five cats per sack* and *three more sacks;* and the combination is *15 more cats*. Two intensive quantities can be combined multiplicatively, and their combination is another intensive quantity; for example, "There are seven kits per cat; and there are five cats per sack; therefore, there are 35 kits per sack."

When two extensive quantities are combined multiplicatively, their combination is a Cartesian product, which is another extensive quantity; for example: "Jay has five shirts and seven ties; therefore, he has 35 shirt-and-tie combinations." An extensive quantity and a difference can be combined multiplicatively, and their combination is a difference; for example: "Jay has five shirts; and he has four more blue ties than red ties; therefore, he has 20 more combinations of shirts with his blue ties than combinations of shirts with his red

ties." We have not found a meaningful way to combine two differences multiplicatively, so Table 2 has the entry ∅ in that cell.

Table 3 shows additive compositions involving all four types of quantities. Extensive quantities do not combine additively with intensive quantities or factors, and intensive quantities do not combine additively with factors, as far as we can determine. Factors can be combined additively

Table 3

Additive Compositions

	Extensive	Difference	Intensive	Factor
Extensive	Extensive	Extensive	∅	∅
Difference		Difference	Intensive	Factor
Intensive			[Intensive]	∅
Factor				Factor

with other factors; for example: "Jay has three times as many books as El; and Kay has five times as many books as El; together, Jay and Kay have eight times as many books as El." A difference can be combined additively with an intensive quantity, and the result is another intensive quantity; for example: " Jay's average of hits per times at bat was .005 less than Kay's average of hits per times at bat; Jay's average was .318; and Kay's average was .323." Factors also can be compared additively, although the comparisons are somewhat awkward. An example is: " Jay has three times as many books as El; and Kay has five times as many books as El; so Kay has two more times as many books as El than Jay."

An entry in Table 3 of special interest is the additive combination of two intensive quantities. The physical operation corresponding to addition is concatenation, but when two objects or substances with different values of an intensive quantity are concatenated, the value of the quantity is not the additive sum of the components. For example, if two objects with densities D_1 and D_2 are fastened together, the resulting object does not have density $D_1 + D_2$. The quantity that results when two intensive quantities are combined depends on other quantities -- specifically, extensive quantities that are the amounts of the objects involved in the combination. For example, if two objects with densities D_1 and D_2 and masses m_1 and m_2 are fastened together, the resulting object has density

$$D_{12} = (D_1m_1 + D_2m_2)/(m_1 + m_2).$$

Instructional System for One-Step Problems. Students are required to solve word problems, and according to our analysis, this requires comprehending relations among quantities in the problems. Instruction usually does not include explicit attention to patterns of quantitative relations, and we hypothesize that a medium for representing quantitative patterns explicitly can aid students in learning to solve word problems. The instructional system that we have constructed is a graphical system, developed by Valerie Shalin, Nancy Bee, and Ted Rees, that allows students to construct diagrams that represent the patterns of information in word problems. To represent a problem, a student identifies the quantities that are mentioned in the problem and includes them in the representation. In more complex problems, quantities that are not mentioned also need to be identified and represented.

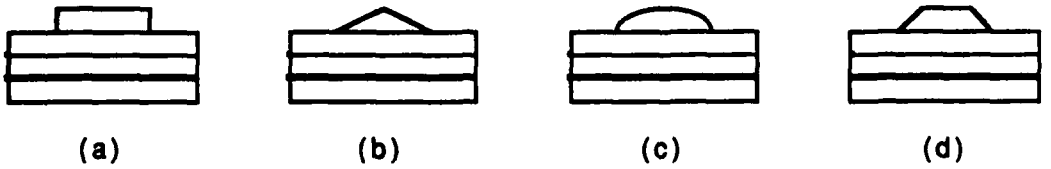


Figure 2. Menu of shapes for representing quantities.

Figure 2 shows the menu of shapes that are in the system. Shape (a) is for extensive quantities, shape (b) is for differences in additive comparisons, shape (c) is for intensive quantities, and shape (d) is for factors in multiplicative comparisons.

To represent a problem, a student constructs a semantic network that has one of the shapes in Figure 2 for each of the quantities in the problem, and links between the quantities corresponding to their compositional relations. As an example, consider the problem:

"Jay put 48 books into bags,with six books in each bag; how many bags were there?"

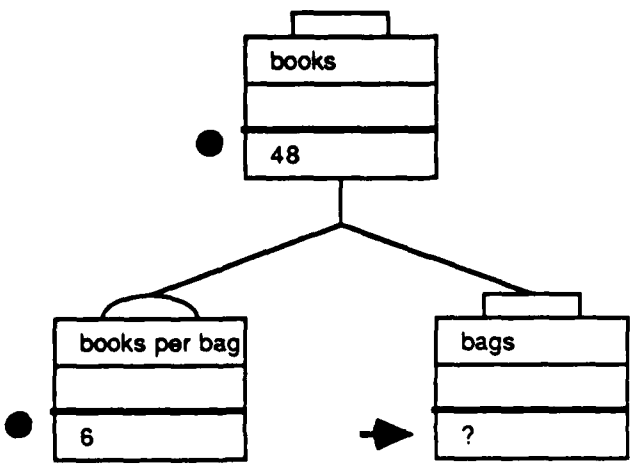


Figure 3. Semantic network for a one-step problem.

Figure 3 shows a diagram for this problem. The units of the three quantities are *books, bags,* and *books per bag*. A convention of the system is that in additive compositions, the components that are added are beside each other, and the quantity that they form as their sum is above them, and that in multiplicative compositions, the components that are multiplied are at one level, and their product is above them. Therefore, in Figure 3, the quantities for *books per bag* and *bags* are at one level, and the quantity for *books* is above them. Numerical values for *books* and *books per bag* are given, and the quantities with given values are marked with filled circles. The number of *bags* is unknown, and that quantity is marked with an arrow. The types and units determine that the relation is multiplicative, and because one of the component quantities is unknown, with the other two numerical values given, the answer is found by dividing the combined quantity by the given component. Figure 4 shows the diagram with these inferred steps included.

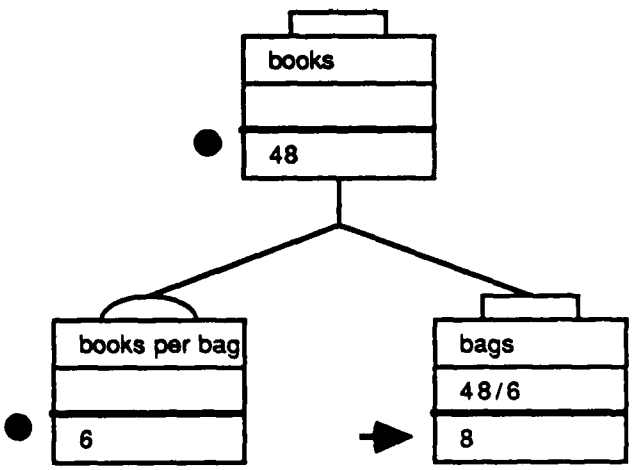


Figure 4. Semantic network for a one-step problem, including the solution.

Analysis and Instructional System for Two-Step Problems. Problems requiring more than one step can be represented by combining two or more triads like the one in Figures 3 and 4. An example is in Figure 5, for the problem

The charge for parking is $.65 per hour. Ms. Jones parked for four hours. How much change did she get from $10.00?

The *dollars charged per hour*, *hours*, and *dollars paid* are the given values, and the *difference of dollars paid more than dollars charged* is the unknown. Another quantity, the *dollars charged*, is not mentioned in the problem. The need for this quantity has to be inferred by a student in order to solve the problem. It is an extensive quantity, resulting from the combination of *dollars charged per hour* and *hours*. The representation includes the expression 4 * 0.65, indicating the operation for calculating the numerical value 2.60.

The unknown quantity in Figure 5 is the difference between two extensive quantities. It is combined additively with the smaller of those quantities; the spatial arrangement with the smaller quantity beside the difference and the larger quantity above them preserves the convention that addition moves upward in the diagram. Because the calculation uses the combined quantity and one of the components, the operation used is subtraction. Figure 5 shows two intermediate expressions

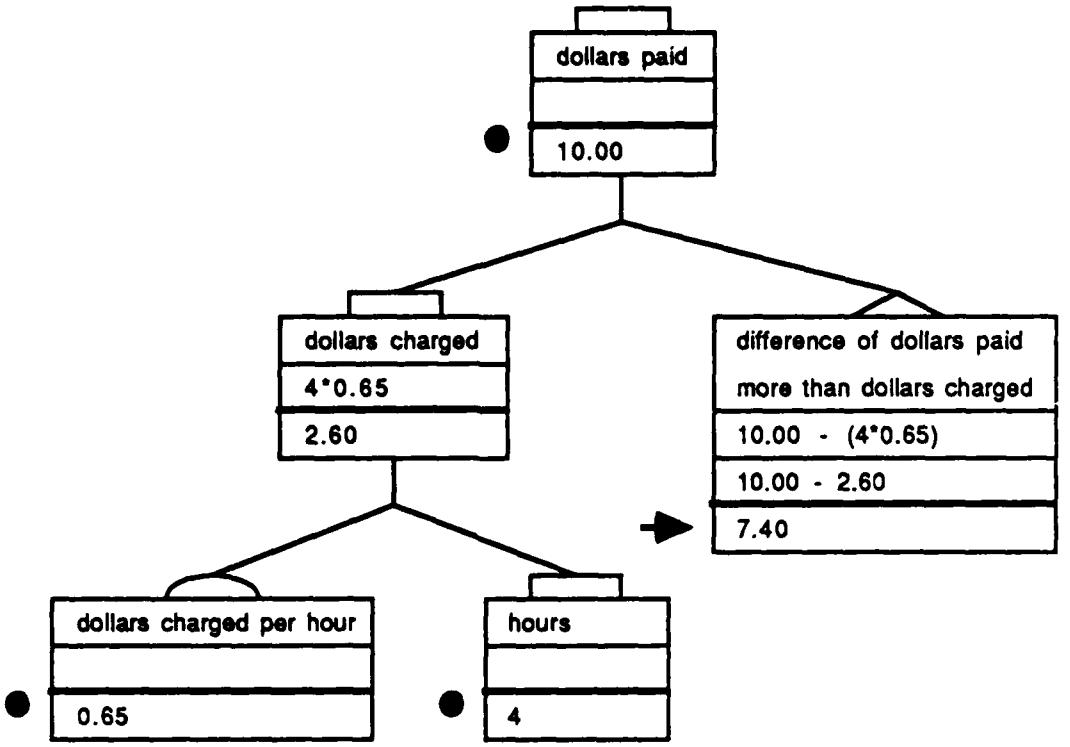


Figure 5. Graphical representation of a two-step problem.

in the quantity for the difference: 10.00 - (4 * 0.65) and 10.00 - 2.60. Both of these can be obtained from the representations for *dollars paid* and *dollars charged*, and inclusion of the alternative expressions calls attention to their equivalence. The answer for the problem, 7.40, is marked with an arrow.

Shalin and Bee analyzed a domain of word problems that contain three quantities with given values and an unknown that is found using two operations. The problems can all be solved by evaluating an expression of the form A op (B op C) or (A op B) op C, where each operator is from the set {add, subtract, multiply, divide}. The problems all have representations that involve five quantities, with two triads that are combined.

One form of the combination iproduces a hierarchy, such as Figure 5. This occurs when a single quantity is the combination of one triad and a component of the other triad in

the problem. There are two other patterns in which the quantities in two triads can be combined. One of these is illustrated in Figure 6, which represents the following problem:

> *Paul works in a doughnut shop. He made 14 boxes of glazed and powdered doughnuts, and 8 of these boxes were powdered doughnuts. If he put 12 doughnuts in each box, how many glazed doughnuts did he make?*

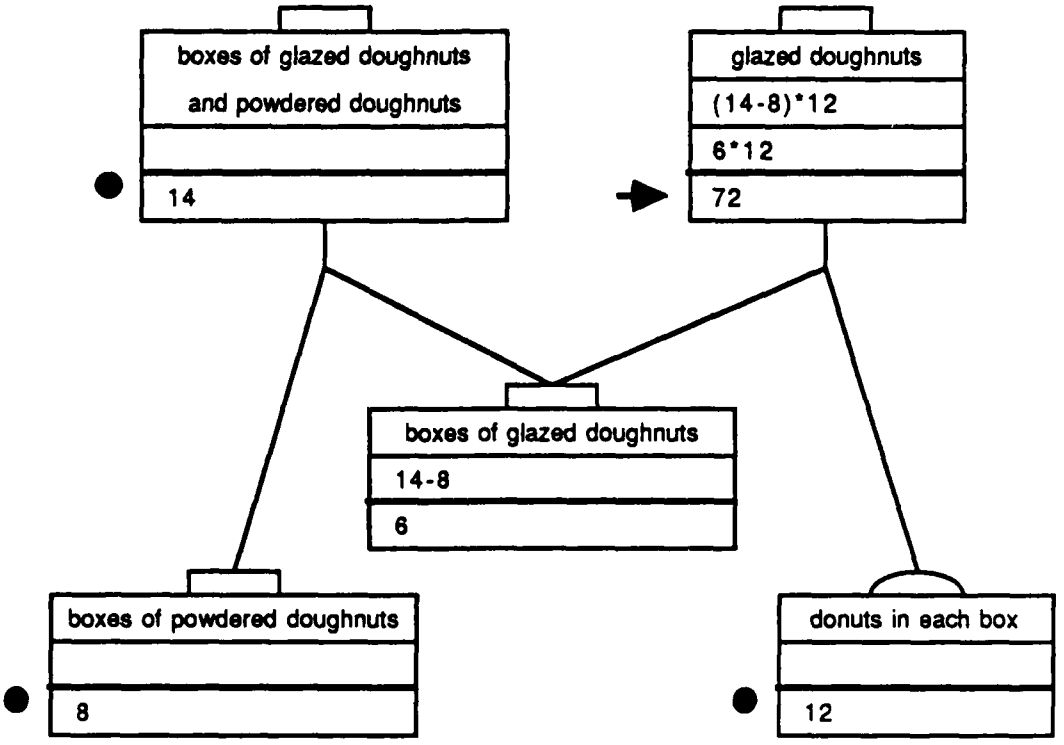


Figure 6. A problem with a shared-component structure.

The inferred quantity, *boxes of glazed doughnuts*, is a component in both of the triad structures. Its value is obtained by subtracting the number of boxes of powdered doughnuts from the total number of boxes, and then it is used along with the number of doughnuts per box to obtain the number of glazed doughnuts by multiplication.

The third kind of pattern is illustrated in Figure 7, for the problem:

Dr. Wizard has discovered a group of monsters living in a dark cave in South America. He has counted 7 monsters and there are 8 fingers on each monster. If there are 4 fingers on each monster hand, how many monster hands did he find?

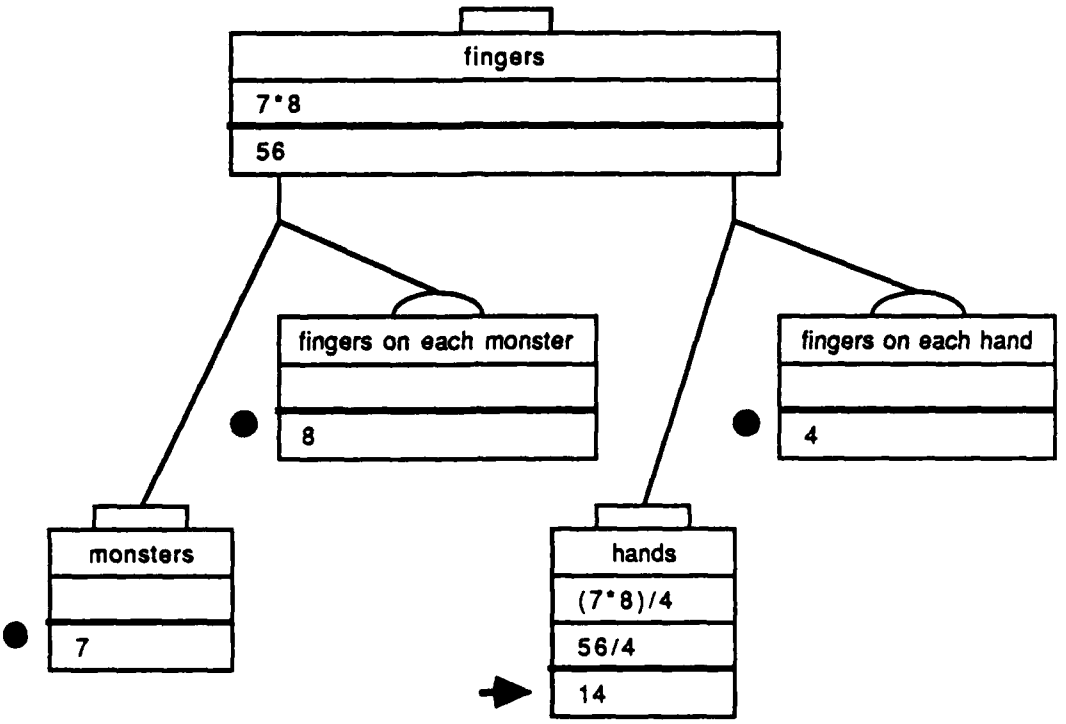


Figure 7. A problem with a shared-combination structure.

Here the inferred quantity, the number of *fingers*, is the combined quantity in both of the triad structures. It is obtained by multiplication of the *monsters* and *fingers on each monster* quantities, and then used along with *fingers on each hand* to obtain the number of hands, using division.

Empirical Study of Structural Patterns. Shalin and Bee (1985b) conducted an empirical study to test whether the differences in problem structures identified in their analysis correspond to psychologically significant distinctions. Problems were constructed with the different patterns, allowing for comparison across sets of problems with the same

operations. Sets were also approximately balanced for the numbers involved. If the difficulty of problems is related systematically to the patterns of quantitative structure, we can take this as at least presumptive evidence that the distinctions in the theoretical analysis correspond to significant factors in psychological processing of the different problems.

Shalin and Bee constructed 34 problems, representing 11 patterns of quantities. The problems included three combinations of operators: (a) problems with two additive compositions, (b) problems with one additive and one multiplicative composition, and (c) problems with two multiplicative compositions. They also included the three kinds of quantitative patterns: (a) binary trees, (b) patterns with a shared component, and (c) patterns with a shared combination. Examples of problems with one additive and one multiplicative composition include the problem of Figure 6, which has a shared component, and the following two problems

(binary tree)

Tom is restocking the shelves in his grocery store. He has 840 diet and regular cokes altogether, and there are 8 cokes in each carton of cokes. If this makes 65 cartons of regular coke, how many cartons of diet coke does Tom have?

(shared combination)

The Girl Scouts are selling cookies for their annual fund raiser. There are 24 cookies in each box of cookies and 34 boxes of chocolate and vanilla cookies. If there are 360 vanilla cookies in these boxes, how many chocolate cookies are there?

The representation of most problems is not unique; for example, the problem above about Girl Scout cookies can be represented without a shared combination by using six quantities involving three triads, leading to a solution in three steps rather than two. The patterns that were designated by Shalin and Bee for the different problems were those that provide

simpler representations and solutions, and they used word orders that should bias the representations in the way they specified.

The use of 11 patterns, rather than nine, and the use of 34 problems, occurred because of variations within some of the cells that were included to increase the comparability of some variables across problems, such as the use of intensive quantities and multiplicative factors in multiplicative compositions. Three versions of the problem set were constructed with different assignments of numbers to problems, providing an approximate balancing of numbers used in the different problem structures.

Eighty-two elementary school children worked for 40 minutes on the word problems. The problems were presented in booklets that had the problems in different orders, providing an approximate balance in the problems that children tried to solve during the session. Children were in the third, fourth, and fifth grades, but their instruction was individually paced, and children were working in text books at grade levels from three through seven. These text levels were used in analyzing the data. Each child's performance was scored by counting the number of each kind of problem he or she solved correctly, and dividing that by the number of that kind of problem the child attempted to solve during the session. The data are in Tables 4, 5, and 6, for the problems with different combinations of additive and multiplicative compositions.

The data provide a fairly consistent picture in which problems with binary tree patterns were easier than the other patterns. The patterns with shared components were easier than those with shared combinations in problems with two additive compositions, and in problems with one additive and one multiplicative composition; with two multiplicative compositions, the shared-combination pattern may have been easier than the shared-

component pattern. Analyses of variance were performed using the data for the different text grade levels in separate analyses. For all five grade levels, the effect of

Table 4

Mean Proportion Correct on Problems with Two Additive Compositions

Text Grade Level	Binary Tree Patterns	Shared Component Patterns	Shared Combination Patterns
3	.44	0	0
4	.75	.12	.05
5	.79	.56	.32
6	.92	.88	.46
7	.89	.67	.44
Average	.76	.45	.25

Table 5

Mean Proportion Correct for Problems with One Additive and one Multiplicative Composition

Text Grade Level	Binary Tree Patterns	Shared Component Patterns	Shared Combination Patterns
3	.03	0	.06
4	.21	.23	.19
5	.44	.45	.46
6	.88	.77	.73
7	.82	.86	.47
Average	.48	.46	.38

Table 6

Mean Proportion Correct on Problems
with Two Multiplicative Compositions

Text Grade Level	Binary Tree Patterns	Shared Component Patterns	Shared Combination Patterns
3	.22	0	.10
4	.17	.14	.13
5	.43	.14	.35
6	.83	.60	.78
7	1.00	.59	.82
Average	.53	.39	.44

problem structure was significant. The effect of the different kinds of compositions was significant only for the fifth grade level students, and structure interacted significantly with the kind of composition for the third, fourth, and fifth grade levels. The data support a conclusion that the distinctions between problem structures are related to the psychological requirements of solving the problems in a significant way.

Preliminary Results with the Instructional System. We hypothesize that if students are taught to represent word problems using the graphical display system to construct semantic networks, they will acquire knowledge that will facilitate their understanding of problem information and therefore improve their performance in solving problems. A systematic empirical test of this hypothesis has not yet been conducted. However, Shalin and Bee have conducted a preliminary study in which four students

learned to construct diagrams for one-step problems involving several of the additive and multiplicative compositions.

Subjects in the study were five students, recruited from a class in remedial algebra at the University of Pittsburgh. Pretests were given including 56 one-step problems, 28 of which were stated with numerals and 28 with letters (e.g., "Barbara and Kenneth have *a* books altogther; *b* of the books are Barbara's; how many of the books belong to Kenneth?"). In another test, 64 two-step problems were given, including 32 stated with numerals and 32 with letters. Time limits were imposed, with 15 minutes for each set of one-step problems, 25 minutes for the two-step problems with numerals, and 35 minutes for the two-step problems with letters. Students were told that they could skip problems and return to them later. The one-step problem pretest was given before any instruction; the two-step problem pretest was given after a few of instructional sessions had been given, during the second main unit of the instruction.

Students were given instruction using the graphics system in one-hour sessions. The instruction included explanations of the different types of quantities in word problems and the conventions for constructing semantic networks. Five instructional units were developed, each concerned with one of the following types of one-step problem: additive combination, additive comparison, multiplication of an extensive and an intensive quantity, multiplicative comparison, and multiplication of two intensive quantities. Each unit except the last one introduced a new type of quantity, and instruction included a discussion of the new quantity type and the shape used to represent it graphically. A problem using the new kind of quantity was shown as an example. Then the student practiced labelling individual boxes on the screen according to text phrases that specified single quantities.

Instruction was then given using example problems. First, the instructor discussed a problem setting with the student indicating the placement of quantities in the network and

the correspondence between the quantities in the word problem and the quantities in the network. Next, an example was used to explain different interpretations of a triad.

(1) In an additive combination, the triad was described as a whole with two parts and as two parts combining to make a whole.

(2) Additive comparison triads were described in terms of the larger quantity broken into two sets, with one set equal in size to the amount of the smaller quantity and another set for the amount left over. Additive comparison triads were also described in terms of the smaller set augmented by the difference set, creating a set equal in size to the larger quantity. Finally, the difference quantity was described as an abstract relation between the two concrete quantities.

(3) Multiplicative triads with two extensive and one intensive quantities were described as converting or exchanging or cashing in the smaller quantity, with the intensive quantity as a rate of exchange. These triads were also interpreted as sorting the larger quantity into piles represented by the smaller "container" quantity, with an equal number in each pile represented by the intensive quantity. These multiplicative triads were also described as an abstract relation, the intensive quantity, between two concrete quantities.

(4) Descriptions of multiplicative comparison triads were similar to those of multiplicative triads with an intensive quantity, with the main difference involving different dimensions in the extensive-intensive triads, and only one dimension in multiplicative comparisons.

(5) The quantities in multiplicative triads with three intensive quantities were described approximately as follows: "The stuff that there's the most of, the stuff that there's an intermediate amount of, and the stuff that there's the least of. The intensive quantities with round hats refer to possible relations between these. The two relations at the bottom of the network involve the intermediate quantity. The relation at the top of the

network relates the stuff that there's the most of to the stuff that there's the least of, by-pass int the intermediate quantity and relating the smallest and largest quantities directly. All possible relational descriptions were also provided: "The right-hand intensive relates the top intensive to the left-hand intensive," etc.

Next, the student saw the permitted permutations in the network (e.g., an exchange in the positions of the parts) and those that are not permitted (e.g., an exchange of a part and the whole). Unpermitted examples were demonstrated by inverting the display, showing black boxes with white printing. The student gave explanations for the illegality of the unpermitted arrangements.

Next, two examples were used to describe the step-by-step process of representing word problems with values of two quantities given and the third value unknown. Then the student worked a series of problems, including construction of network diagrams. The student worked approximately four examples, or as many as the instructor considered necessary until the student appeared to have learned to solve the kind of problem being used. Thinking-aloud protocols were obtained during the solution of these problems.

The instruction then moved to another kind of problem, involving the same types of quantities but with different types given and unknown. The student was told that the new set of problems had the same types and pattern of quantities, but the order in which the quantities were mentioned was different. For the unit on additive combinations, there were two kinds of problems; for the other units, there were three kinds of problems.

Finally, the student practiced on a set of problems with about one-half of the problems of the kind that the student had just learned, and the rest distributed among the kinds of problems studied previously.

Throughout the instruction, students were prompted with questions such as "Is that right?" and "How do you know?" following their solutions. Students were also asked

frequently about their placement of quantities with questions such as "How do you know where things go?" or "How do you know there are more?" and were encouraged to focus on the words to determine quantity placement rather than the numbers. The instructor also pointed out the consistence between quantity placement and word order conventions in additive (more-than) and multiplicative comparisons and in extensive-intensive multiplicative compositions, in which the name of the relational quantity always refers to the largest quantity first (placed at the top) and the smallest quantity second (placed at the bottom).

All of the instruction with graphical representation involved one-step problems with numerical values of variables (i.e., not with letters). Students were told that the representations could be used for more complex problems; this came up when students wanted to enter the results of arithmetic operations before linking the boxes, and the instructor said that it would be easy to get confused if there were several boxes on the screen for a multi-step problems. Students also were told that the value in the bottom line of a box could be anything, even a variable, and the procedure for setting up the network would be the same.

The total instructional time was about 15 hours. Following the instruction, the student solved a set of review problems using the graphics system, with one problem of each kind from the instruction, presented in random order. Then the student took a posttest that included 120 problems, of the same kinds given in the pretest, and with the same time limits.

Results of this preliminary study are necessarily impressionistic. The most important finding is that students found the task of learning to construct representations a reasonable one. Apparently the human factors issues in the design of the graphics system have been dealt with reasonably well. We also have the impression that learning the graphical representation served its intended purpose of teaching students to recognize a set

of distinctions and patterns that are important forms of information in word problems. This impression is supported by comments in the protocols, such as the following:

> "It's -- there's a lot of different ways of looking at these. You know, usually, what I usually do, I look at them one way. But this helps me to see that the different way that you can look at it, which would be helpful, you know. Because sometimes you will have this as the unknown and that as the unknown, or whatever."

> "The hat, though, tells me that I'm, well, you know, like when you think of, math, you think of addition and subtraction together. And multiplication and division together. And if I look at this hat, I know that that's a difference box and difference is subtraction. So, depending on whether, you, though I know I'm going addition or subtraction right there. And then wherever the unknown is, I work toward that. If I'm working up I add, if it's down at the bottom, I'm working down, I subtract."

> "This tells me which box to pick: the pointy one or the round one. But then I don't do what I'm going to do until I see it all set up."

Table 7

Problems Attempted

Steps, Student Pretest, Number Problems		Posttest, Number Problems	Pretest, Variable Problems	Posttest, Variable Problems
One-Step, S1	28	28	10	28
One-Step, S2	28	28	23	28
One-Step, S3	24	24	12	28
One-Step, S4	28	28	23	24
	—	—	—	—
One-Step, Average	27.0	27.0	17.0	27.0
Two-Step, S1	31	21	11	13
Two-Step, S2	29	32	22	30
Two-Step, S3	27	29	23	25
Two-Step, S4	19	24	25	27
	—	—	—	—
Two-Step, Average	26.5	26.5	20.2	23.7

Table 8

Correct Solutions

Steps, Student Pretest, Number Problems	Posttest, Number Problems	Pretest, Variable Problems	Posttest, Variable Problems
One-Step, S1	28	28	3	25
One-Step, S2	28	28	17	28
One-Step, S3	24	24	10	15
One-Step, S4	28	28	17	23
One-Step, Average	27.0	27.0	11.7	22.7
Two-Step, S1	28	19	2	10
Two-Step, S2	26	29	14	22
Two-Step, S3	26	25	13	16
Two-Step, S4	16	23	19	25
Two-Step, Average	24.0	24.0	14.5	18.2

Tables 7 and 8 show the results from the pretest and posttest. There was no improvement in performance on problems with numerals, although this might be due to a ceiling effect with the college-age students who participated in the study. There was, however, a substantial improvement in performance on the problems with values of quantities denoted as letters, requiring formulas as solutions. Recall that instruction involved only one-step problems with numerical values of quantities, so the problems with variables as values were new (as were the two-step problems). A reasonable interpretation is that the instruction may have provided students with knowledge for representing problem

information more abstractly, so the combinations of quantities denoted by letters were easier to reason about representing problem information more abstractly, so the combinations of quantities denoted by letters were easier to reason about.

3. Basic Skill in a Transformation Task

Analyses of performance in mathematical tasks have contributed significantly to the theory of procedural knowledge and problem solving. One feature of this contribution has been the detailed analysis of strategic knowledge that is acquired through instruction for problem solving in specific domains.

Previous Analyses. Analyses of solving proof exercises (Anderson, 1982; Greeno, 1978) in geometry provided detailed examples of knowledge that organizes problem-solving activity in a task domain in which students receive instruction and practice. Rather than the weak problem-solving methods that characterize problem solving in novel domains (cf. Newell & Simon, 1972), students in geometry appear to use specific strategies that use features of the geometry problem domain. Examples include proving that triangles are congruent as a subgoal for proving that components of the triangles are congruent, and using relations between angles formed by parallel lines to prove that angles are congruent or supplementary.

Analyses of procedural knowledge in elementary arithmetic also have provided theoretical understanding of knowledge for organizing problem-solving activity through the setting of goals. A significant portion of errors in arithmetic tests apparently are caused by underlying procedural flaws, or "bugs," and these can be understood as variations within a general structure of an arithmetic procedure that includes the main components, such as answering in each column (Brown & Burton, 1980). Acquisition of many of the incorrect procedures can be explained by assuming that the general goals of the procedure are

known, but the student's knowledge is incomplete and therefore an impasse is encountered in which the student is unable to achieve a goal with the procedure that he or she knows. The student then is assumed to use an action that satisfies the general goal using general problem-solving heuristics, but this action may not be correct. However, if that action is remembered and used in later problems, the student's performance will show one of the "bugs" (VanLehn, 1983).

In a research project conducted before the one described in this report, an empirical study was conducted of problem-solving performance by beginning students in algebra (Greeno, Magone, Rabinowitz, Ranney, Strauch, & Vitolo, 1985). The intention of that project was to develop cognitive models of students' knowledge in early stages of learning algebra. Our major conclusion was that the beginning students in our study had largely fragmentary and unsystematic knowledge. This contrasts with the conclusions about characteristics of mathematical knowledge in other domains that have been studied, in which the knowledge that students acquire is apparently quite well organized, as we have discussed above.

There might be several reasons for students' acquiring fragments of knowledge, rather than a systematically organized procedural structure. One reason that seems plausible is a lack of prior knowledge of the kind of procedure that is needed in the new domain. When we considered the cognitive requirements of elementary algebra problems, we realized that they have a fundamentally different structure from that of almost all the tasks students learn to perform in arithmetic. In arithmetic, almost all problems involve *evaluating* symbolic expressions, but in algebra, most problems involve *transforming* symbolic expressions into equivalent expressions. The operators and goals for transformation tasks differ significantly from those of evaluation tasks. This means that students who have learned arithmetic have had little or no training in the general structure of procedures that they need to acquire when they begin learning algebra. The algebra domain

also presents significantly new kinds of symbolic data: variables and equations. These also cause difficulty in learning the domain (e.g., Kieran, 1980; Matz, 1980; Wagner, 1981).

A New Task for Learning Procedural Structure. A new task was developed to allow investigation of knowledge and learning of procedures like those of algebra, but without the new symbolic structures of algebra. We obtained significant new results about the acquisition of procedures in a symbolic domain by observing students learning the task. The task is also potentially useful for instruction in preparation for algebra or concurrent with the beginning of algebra instruction, and we investigated some issues in the design of intelligent tutoring systems by developing some components of a computer-based tutor for the task.

The task that we developed is an adaptation of a symbolic transformation task used by Newell and Simon (1972) in a classic cognitive analysis of problem solving. Newell and Simon studied the task of finding proofs in propositional logic. A set of rules is given, for example, $A \vee B \Leftrightarrow B \vee A$, $A \supset B \Leftrightarrow \sim A \vee B$, and $A \supset B$, $A \Rightarrow B$. In the task, one or more expressions are given, and another expression is the goal. The task is to find a set of transformations that work from the given expressions and produce the goal. For example, if the given expressions are $Q \vee \sim P$ and P, and the goal is Q, then the first rule can be used to obtain $\sim P \vee Q$ from $Q \vee \sim P$, the second rule can be used to obtain $P \supset Q$ from $\sim P \vee Q$, and the third rule can be used to obtain Q from $P \supset Q$ and P.

R1 =>	A + B <=> B + A	<= R1
R2 =>	A x B <=> B x A	<= R2
R3 =>	A x (B + C) <=> (A x B) + (A x C)	<= R3
R4 =>	A x (B - C) <=> (A x B) - (A x C)	<= R4
R5 =>	(A + B)/C <=> (A/C) + (B/C)	<= R5
R6 =>	(A - B)/C <=> (A/C) - (B/C)	<= R6
R7 =>	A + (B + C) <=> (A + B) + C	<= R7
R8 =>	A x (B x C) <=> (A x B) x C	<= R8
R9 =>	A - (B - C) <=> (A - B) + C	<= R9
R10 =>	A - (B + C) <=> (A - B) - C	<= R10

Figure 8. Rules for the transformation task for arithmetic espressions, as shown on the computer graphics display.

In the task that we developed, the given expressions and goals are arithmetic expressions. An example problem has the initial expression $3 \times (5 + 4)$, and the goal is $(3 \times 4) + (3 \times 5)$. Rules are given for transforming expressions. Figure 8 shows the set of rules in the form that they are displayed in the computer-based tutor. To solve the example problem, Rule R3 can be used to obtain $(3 \times 5) + (3 \times 4)$ from $3 \times (5 + 4)$, then Rule R1 can be used to obtain $(3 \times 4) + (3 \times 5)$ from $(3 \times 5) + (3 \times 4)$.

The procedural structure of this task is like that of logic exercises, and different from that of ordinary arithmetic. Newell and Simon (1972) characterized the process of solving logic exercises as a form of heuristic search that they called means-ends analysis. The problem is solved by moving through a sequence of states. The initial state is the set of given expressions, and the goal is the expression to be derived. Each step in the solution produces a new state by adding a new expression to those given and previously derived. In the means-ends method, the current state is compared with the goal state, and differences between the current and goal states provided the basis for selecting an operator. Of the differences that are found between the current and goal state, one difference is

chosen to be the focus of activity; a subgoal is set to reduce or remove that difference. An operator is chosen that is known to change the feature that distinguishes the current state from the goal. An attempt is made to apply that operator. Operators have conditions that are required for their application, and the attempt to apply an operator includes a test whether the current state satisfies the operator's conditions. If the operator can be applied, it is, and the result is a new problem state. Then this new state is compared with the goal, and if the goal has not been reached, another difference will become the basis for a new subgoal. If the operator cannot be applied, there may be another subgoal to change the state to satisfy the operator's conditions, or another operator may be selected to try to achieve the same subgoal, or the subgoal may be set aside and a new subgoal may be chosen based on another difference between the current state and the goal.

The difference between the procedures of ordinary arithmetic and the transformation task can be illustrated with the problem mentioned earlier, with the given expression $3 \times (5 + 4)$ and the goal expression $(3 \times 4) + (3 \times 5)$. In the means-ends method, the initial state and the goal state are compared, and differences are noted. The differences depend on the way that expressions are represented. A representation for arithmetic that is like Newell and Simon's (1972) for logic uses a tree structure with operators, subexpressions, and numerals. There are several differences between the initial and goal states; one is that the operators + and × appear at different levels in the two representations, with × being the highest level operator in the initial expression and + being the highest level operator in the goal expression. In a model of successful problem solving that we developed, this difference is chosen as the target of a subgoal. There are four operators that change the level of an operator; these are the distributive rules R3, R4, R5, and R6. R3 is chosen because the operators in R3, + and ×, are the same as those in the expression to be changed. R3 can be applied either from left to right or from right to left; the left-to-right application changes the highest-level operator from × to +, as needed for the subgoal. The left side of R3 is then matched to the current expression, with A bound to 3, B bound to 5,

and C bound to 4. A new expression is constructed according to the right side of R3, with these bindings, producing $(3 \times 5) + (3 \times 4)$, and the addition of this new expression creates a new problem state. The new expression is compared with the goal state, and the difference is that subexpressions are in different orders. Rules R1 and R2 change the order of terms, and in R1 the operator between the terms is +, as in the expression to be changed. The left side of R1 is matched to $(3 \times 5) + (3 \times 4)$, with A bound to (3×5) and B bound to (3×4). Then the right side of R1 is used to construct a new expression with these bindings, producing $(3 \times 4) + (3 \times 5)$. This expression matches the goal, so the problem is solved.

In ordinary arithmetic, students learn to evaluate expressions, rather than to transform them. An expression such as $3 \times (5 + 4)$ is evaluated by performing a series of operations with the numbers in the expression as arguments. The sequence of operations is indicated by parentheses in the expression, so that for $3 \times (5 + 4)$, the first operation evaluates $5 + 4$, giving 3×9, and this is evaluated to obtain the answer, 27.

Evaluation can be considered as a transformation task, but there are some important features of the "proof" task that are not present in evaluation.

First, in evaluation all operators have outputs consisting of single numbers. The operators are the arithmetic "facts," involving functions that correspond to a +, -, ×, or / sign in a subexpresion, the numbers in the subexpression as arguments, and a single number as the value. In the "proof" task, operators transform expressions into other expressions more generally. The operations are formal, in that they transform expressions based on their form, rather than on the specific numerals that are in the expressions. This difference in the operations involved is reflected in the way rules are stated in the two tasks. The operators of evaluation are stated in terms of specific numbers, such as 3×9 is 27. The operators of the "proof" task are stated in terms of variables that can correspond to many different numerals or subexpressions, such as $A \times (B + C) \Leftrightarrow (A \times B) + (A \times C)$.

A second difference is that in evaluation, structural features of the expression are needed only to determine the sequence of operations. A subexpression is found that can be evaluated, and its value is substituted into the expression. Operations do not change structural features of expressions, except in the simple way of taking a subexpression at the bottom of the tree and replacing it by a single number. In the "proof" task, structural features of the expression are used to select operators and operators transform expressions in ways that change their structures. For example, operators change the groupings of terms in the expressions, and change the locations of operators in the trees that represent expressions.

A third difference between the "proof" task and evaluation is the inclusion of specific goals in the "proof" task. In evaluation, the goal is to "find the value," and operators are applied until the expression is changed into a single numeral. In the "proof" task, the goal is a specific expression to be derived. The choice of an operator is then determined by differences between the current expression and the goal, and attention is therefore directed to the goal expression to determine how it differs from the current expression. In evaluation, where the goal is always to derive a single numeral, attention can be focused entirely on features of the current expression, to find a subexpression that can be replaced by its value.

The tasks that students learn to perform in algebra are like the "proof" task in the first two of these features that distinguish it from ordinary arithmetic, but are like ordinary arithmetic in the third feature. The operations that students learn for simplifying expressions and solving equations are transformations of expressions into other expressions, rather than evaluations with numerical values, and they are defined formally by the structural features of expressions rather than by the specific numerals and letters in the expressions. The goals of obtaining simpler expressions or solutions of equations, however, are not in the form of specific expressions that are to be derived, as in the "proof"

task, but are general criteria to be achieved. The process of simplifying expressions seems to involve more attention to the current expression and an attempt to apply an operation to change it, rather than attention to features that distinguish the current expression from a goal. The process of solving equations is more goal-directed, but the goal is like that of evaluation, involving a single form in which the variable is alone on one side of the equation, so the choice of operators is based entirely on features of the current expression rather than on features that distinguish the current expression from a goal that varies in structure from one problem to another. Further analysis and empirical study would be needed to determine whether and how work on this new task might facilitate or impede students' learning of algebra, and we do not propose a strong opinion about this complex question here. Even so, the features of the "proof" task shares with algebra are sufficient to make it seem worthwhile to study acquisition of the task for its potential usefulness in the mathematics curriculum. And in any case, there is considerable theoretical interest in studying the properties of a task that involves several significant extensions of knowledge beyond that required for arithmetic.

Empirical Observations of Learning. A small empirical study was conducted with a few middle-school students and some students from a remedial mathematics class at a community college. Protocols were obtained as the students learned to solve problems in a paper-and-pencil version of the task. The students were given an explanation of the task and a list of the rules. They were shown an introductory example involving a simple application of Rule R1. Then they were given a series of problems to solve, increasing in complexity.

This informal study had two main goals. First, we determined whether students can learn the task of transforming arithmetic expressions in a reasonable time. Second, we obtained a collection of errors that students make while they are learning the task. These errors provide information about knowledge needed for the task that students do not have

when they begin learning. This is important for a theoretical understanding of the process of acquiring skills in symbolic domains, because analysis of a learning process requires a characterization of the initial knowledge state of the learner as well as a characterization of the knowledge that is acquired. It also is important for the design of instruction in the skill, because the diagnosis of student knowledge and tutorial decisions are based on a characterization of the skill as a set of cognitive components in terms of instructional issues.

In order to observe errors that would provide a full picture of the cognitive requirements of learning, instruction was given in a discovery mode. New kinds of problems were presented without instruction in how to solve them or examples of correct solutions. After the student gave a solution, feedback was given to point out any errors that occurred and to explain the correct operations.

We present errors that occurred in three categories: (1) errors of commission that neglect restrictions; (2) errors of omission in which operators are not applied when they could be; and (3) slips in which variables and values are confused.

(1) Errors of Commission. Several errors involved use of an operator incorrectly, where a global property of the operator's condition is satisfied in the current expression, but a more specific requirement is not satisfied. One kind of error involved matching the terms of a rule to the terms of an expression, where the general form of the rule matched the form of the expression, but the terms of the rule did not correspond to the terms of the expression in all their occurrences.

Apply Rule R3 (right-to-left): $A \times (B + C) \Leftrightarrow (A \times B) + (A \times C)$

Current Expression: $(3 \times 2) + (5 \times 3)$

Correspondence: [A B A C].

The student had not learned that a single term that appears more than once in a rule must be matched with the same numeral in all of its occurrences.

A second kind of error involved transformations that agree with global properties but not with relevant details. One example was the following:

Apply Rule R7 (left-to-right): $A + (B + C) \Leftrightarrow (A + B) + C$

Current Expression: $4 + (7 + 5)$

Correspondence: [A B C]

New Expression: $(7 + 5) + 4$

The student said that the rule and the expression had a "thing in parentheses and something sticking out both ends," and the operation reverses the direction in which something sticks out. Another mistake characterizes Rule R7 as reversing the direction of terms, and produces a new expression in which all the elements are reversed, apparently obtained by reading the left side from right to left; for example, for ,

New Expression: $(5 + 7) + 4$.

(2) Errors of Omission. One kind of error involving an incorrect restriction was observed when students did not apply an operator because the variables in the rule already had values in a previous transformation. For example, a problem was begun by one of the students:

Apply Rule R3 (right-to-left): $A \times (B + C) \Leftrightarrow (A \times B) + (A \times C)$

Current Expression: $(9 \times 1) + (9 \times 3)$

Correspondence: [A B A C]

New Expression: $9 \times (1 + 3)$.

The goal of the problem was $9 \times (3 + 1)$, and Rule R1 could be used to change 1+3 to 3+1. The student said that Rule R1 should be used, but it could not be applied because A already had the value 9. The missing knowledge was that after a transformation is completed, the bindings of the variables used in that transformation are no longer in force for the next transformation.

Another kind of error occurred because students had to learn that rules can be used to change subexpressions rather than whole expressions; for example,

Apply Rule R1 (left-to-right): $A + B \Leftrightarrow B + A$

Current Expression: $3 \times (4 + 5)$

Correspondence: [A B].

Students had to learn that the rules are not restricted to situations where the complete expression is matched to the rule. A similar requirement is to learn that variables in a rule can be bound to subexpressions, rather than only to single numerals, as in:

Apply Rule R2 (left-to-right): $A \times B \Leftrightarrow B \times A$

Current Expression: $(2 + 4) \times (7 + 1)$

Correspondence: [A B]

Some students were unwilling to apply a rule in this way, assuming that variable had to be matched to individual numerals until they were instructed that variables could be bound to subexpressions.

(3) Slips between Values and Variables. In the experimental situation, students directed the instructor to write on paper, providing an audible account of the session to be recorded. Errors that occurred frequently in the process of telling the instructor what to write involved substitution of variable names for the values of the variables. For example:

Apply Rule R5 (right-to-left): $(A + B)/C \Leftrightarrow (A/C) + (B/C)$

Current Expression: $(2/5) + (3/5)$

Correspondence: [A C B C]

New Expression: $(A + B)/C$

The slip can also occur in part of the expression, as in the following:

Apply Rule R3 (right-to-left): $A \times (B + C) \Leftrightarrow (A \times B) + (A \times C)$

Current Expression: $(3 \times 4) + (3 \times 5)$

Correspondence: [A B A C]

New Expression: $3 \times (4 + C)$

These slips probably result from execution failures in the process of reading the result of a transformation, which is a translation from an expression in a rule to an expression with numerals, based on a set of bindings of the variables, in which the process of recalling the values is dropped out. These errors may not indicate significant procedures that need to be learned, in the sense of their being unknown by the students, but they indicate that there are distinctions in the symbolic domain that are not handled easily when the information-processing load is heavy.

The errors that we have described here can be characterized as involving the technical skill needed to perform the task, in contrast to the strategic knowledge that is required. Only a few hours of instruction were given, and during this time the students worked on problems requiring understanding of the basic rules, but the problems did not become complex enough to provide much information about their use of the means-ends strategy. There were some instances of learning of strategic knowledge, in which students began to notice differences between the goal expression and expressions that were given or

had been derived, and to use these difference in choosing operators. The most likely conclusion is that strategic knowledge, in the form of processes for performing means-ends analysis, has to be learned when problems that involve nontrivial strategic choices are presented.

The errors that we observed provide detailed information about the components of technical knowledge required for the task, and this involved an aspect of cognitive skill in symbolic domains that had not been considered in detail previously. Newell and Simon's (1972) subjects, working on logic exercises, did not have difficulty with the basic processes of matching rules with expressions, perhaps because they had more advanced training in mathematics. In any case, Newell and Simon's model focused on the strategic aspect of problem solving, and their analyses of protocols focused on the accuracy of simulations provided by GPS for the strategic decisions made by their subjects. Our students had considerable difficulty in processes of matching rules with expressions, and we therefore obtained data that enabled us to characterize the more basic components of skill involving knowledge of how to associate variables in the rules with symbols in the expressions.

Components of an Intelligent Tutoring System.[2] The empirical findings that were described above indicate that development of a computational system to provide instruction for the task of deriving "proofs" for expressions in arithmetic would raise important new questions in the design of intelligent tutoring systems for skills in symbolic domains.

We developed some components of an intelligent tutoring system for the task of deriving arithmetic expressions, using the general framework provided by the WEST tutoring system for an arithmetic game (Burton & Brown, 1982). In the system that

[2]An earlier version of these components was developed by James Rowland

Burton and Brown implemented, the focus of instruction was on strategic knowledge. Technical knowledge in their task consists of basic arithmetic facts of addition, subtraction, multiplication, and division. In the game, a player is given three numbers and has to form an expression using those numbers to determine how far his or her piece will move on the board. The WEST system includes an expert problem solver that generates expressions with the given numbers that are optimal, according to a specified strategy. It also includes a model of student knowledge that is based on an analysis of successful performance, consisting of a set of cognitive issues. A set of filters is included, testing whether each issue is relevant to the solution given by the expert problem solver and a solution given by the student. A differential diagnosis is conducted, comparing the student's solutions over a sequence of trials with the solutions of the expert, and updating the student model with respect to the student's knowledge state regarding the various issues. A tutorial strategy is followed, providing advice to students regarding strategies that could be used differently than the student has been following.

In the tutoring system that we have partially developed, the cognitive issues for the task include both strategic issues and technical issues involving basic skill. Burton and Brown's (1982) system included a provision for issues involving basic skill, but these were not developed explicitly. The extension of the WEST-type tutoring system to include technical as well as strategic issues does not seem to require fundamentally new conceptualizations in the design of the system. On the other hand, there were some significant technical requirements, only some of which we achieved during the time available in this project.

Previous analyses of procedural skill for instruction have characterized knowledge in terms of an ideal model of student performance (Anderson et al., 1984), or in terms of flawed versions of procedures (Brown & Burton, 1980; Sleeman, 1982). The cognitive issues are abstractions from components of correct procedures, and can be considered as

specifications of procedures at what Newell (1982) called the knowledge level. In a procedural characterization, the goals of instruction and diagnoses of performance focus on whether certain components of cognitive procedure have been acquired by students. In the characterization that we give, goals and diagnoses focus on whether the students' cognitive procedures satisfy certain specifications. The two characterizations seem similar in many ways, but whether their differences lead to significant differences in the design of tutoring systems requires further analysis.

Table 9 shows the 11 cognitive issues that were used in the design of a problem generator, and a student model and diagnostic system. Most of the issues were based on the observations of students learning the task, discussed in the previous section.

Table 9

Cognitive Issues of Basic Skill

in the Arithmetic Transformation Task

1. Match the operators of the problem expression to the operators of a rule
2. Bind values of variables consistently from the initial expression to the resulting expression of a rule
3. Bind all occurrences of each variable to a single value
4. Rebind variables in each new problem
5. Rebind variables in each step of a problem
6. Allow the same value for different variables
7. Allow non-isomorphic mappings between an expression and a rule
8. Allow transformations to be applied to subexpressions
9. Allow variables to be bound to subexpressions
10. Bind variables only to syntactically complete subexpressions
11. Use rules either from left to right or from right to left

A problem generator was developed, to enable construction of problems in which a specified set of issues could be involved in the solution. The problem generator constructs pairs of expressions that differ in ways related to the specified issues. The generator produces transformation sites that can be filled either with single numerals or with subexpressions. This permits issues to be included in solutions of problems in a kind of layered structure, with a specification of one transformation at the level of a complete expression, involving one issue, and another transformation involving a subexpression that involves a different issue. The transformations that are specified in the generation of problems provides a solution of the problem; however, there may be other correct

solutions, so it is necessary to provide for solutions that differ from the one that the is specified by the problem generator.

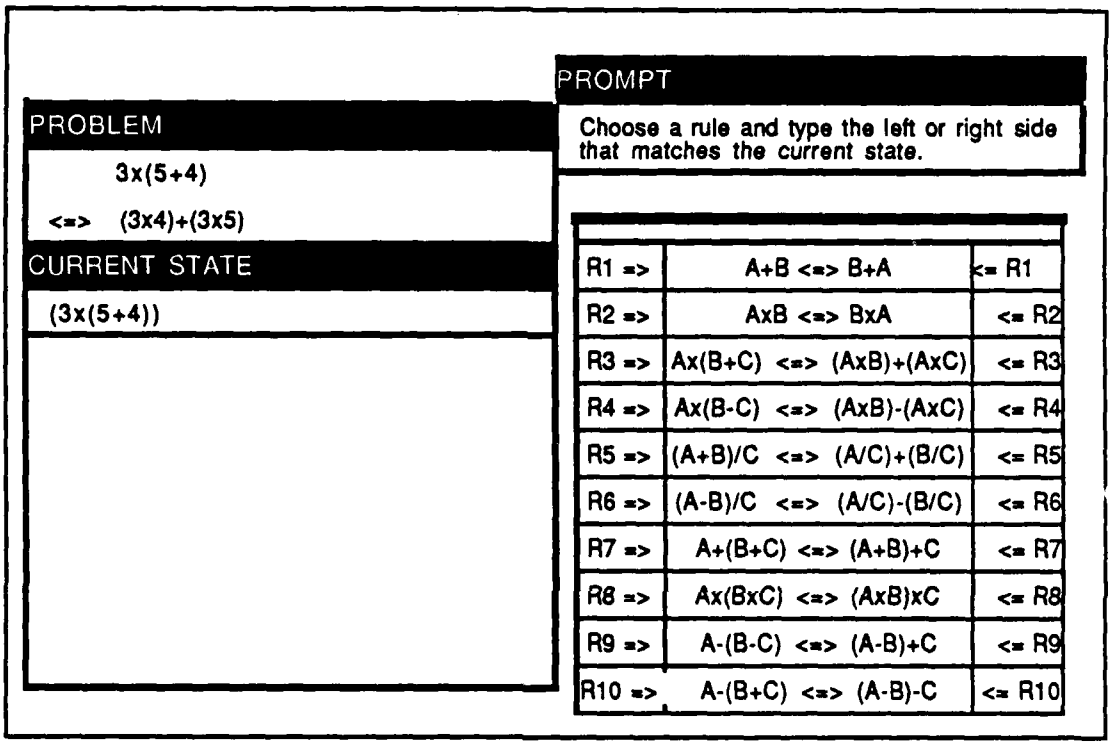


Figure 9. Presentation of a problem.

Figure 9 shows a problem in the form that it is presented to students. Two modes of interaction are available. In one mode, the student types the part of a rule that matches the current state, and the new state that is produced by the application of the rule. If this is correct, the current state changes to the one produced by the student's action, and if the problem is not solved, the student is prompted to choose another rule. In the other mode, the student is required to type the part of a rule that matches the current state and is prompted to type the values that are bound to the variables in the rule in matching it to the expression.

An expert problem solver was developed that solves problems using means-ends analysis. The problem solver is used when a student works on a problem to provide a differential diagnosis of the student's performance with respect to the cognitive issues. When the problem is presented, the expert solver analyzes differences between the given expression and the goal, and chooses a rule that will reduce the most important of the differences, according to an ordering that it uses. When the student chooses a rule, it is compared with the expert's rule. If the student's rule and the expert's rule disagree, information may be obtained about one or more of the issues involving the possibility of using rules in situations where students may not know they can be used. Whatever rule the student uses, information is obtained from the correctness or incorrectness of the rule's use. Filters are used to evaluate the relevance of the issues to the correct application of the rule, and the relevance of the issues to the way in which the student's use of the rule differs from the correct application, if it is incorrect. If the student uses a rule correctly and produces a new expression, the expert uses that new expression as a problem to be solved with means-ends analysis, and work on the problem proceeds.

After a problem has been completed, or a student attempts to use a rule incorrectly, the student model is updated. The student model consists of an inferred level of knowledge for each of the issues, ranging from weak to strong.

Tutoring can be provided, using the student model to choose an issue for which the student's knowledge is judged to be weak. The problem generator can produce a problem in which the issue is relevant, and the expert problem-solver's solution can be shown to the student. An example is in Figure 10. The expert's solution can include explanations of the rules that are chosen using the differences between the current state and goal in means-ends analysis. This full disclosure is shown in Figure 10. An option is to merely show the solution that the expert produced, without the explanations.

We did not implement programs that make tutorial decisions, or that provide tutorial interactions other than student problem solving and presentation of worked examples. Further work on these components would be needed for the tutoring system to function independently of a human tutorial supervisor, and would allow a more complete evaluation of the analysis of the skill as a set of cognitive issues and the other components of the tutoring system that were completed.

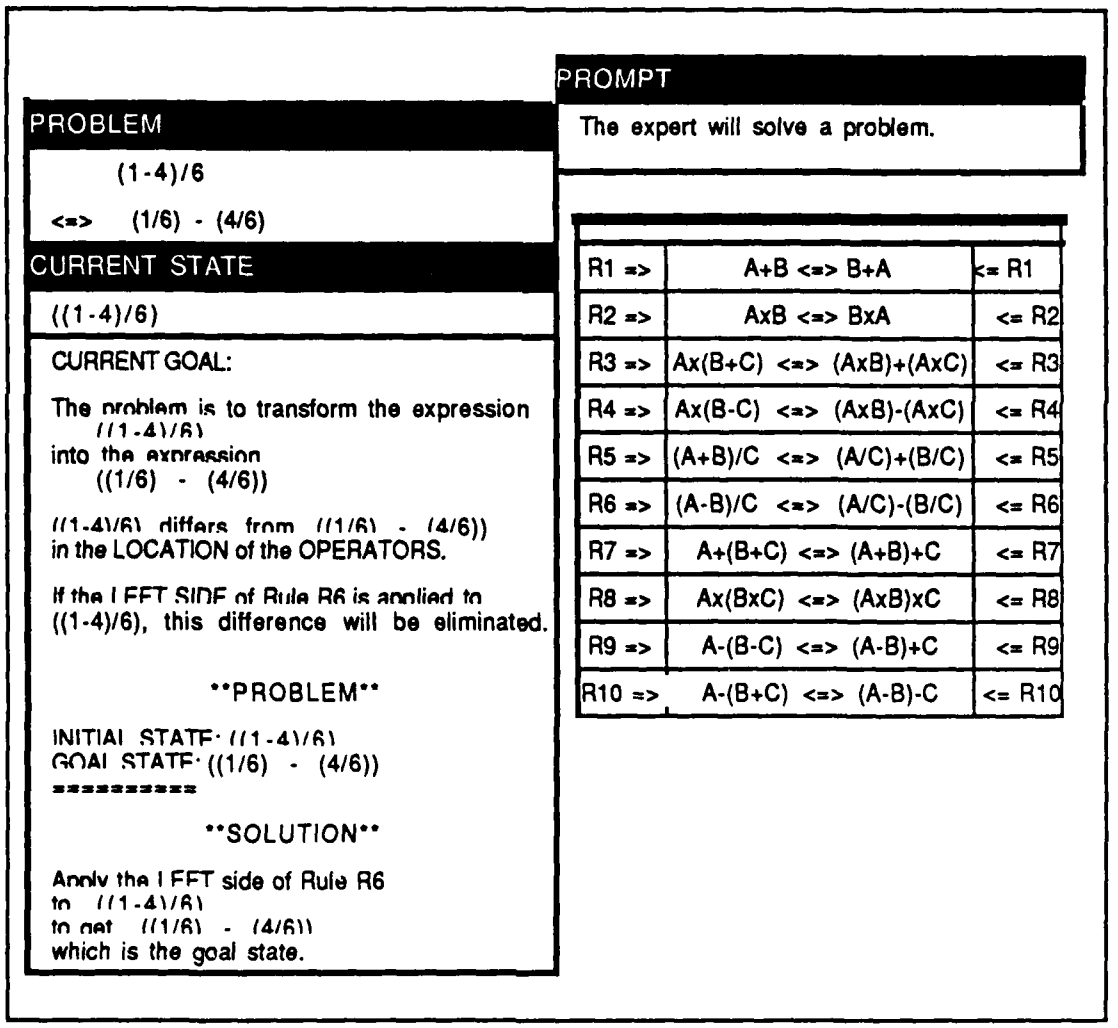


Figure 10. An example problem with means-ends information and a solution.

4. Strategic Knowledge and Reflective Learning

The third project that was included in this contract is concerned with acquiring strategic knowledge in the domain of elementary algebra. By "strategic knowledge" we refer to knowledge for setting goals and adopting plans in work on a problem. Strategic knowledge, like knowledge for representing problems, usually is tacit.

A computational system called Algebraland, developed at Xerox Palo Alto Research Center by Kelly Roach and Carolyn Foss, provides instruction in strategic knowledge for algebra using two pedagogical devices. One device is the use of graphics to present information that is directly relevant to the student's strategic performance, enabling the student to reflect on strategic issues more directly. The second device is to reduce some of the non-strategic demands on the student's problem solving, enabling more attention to be given to performance and learning at the strategic level.

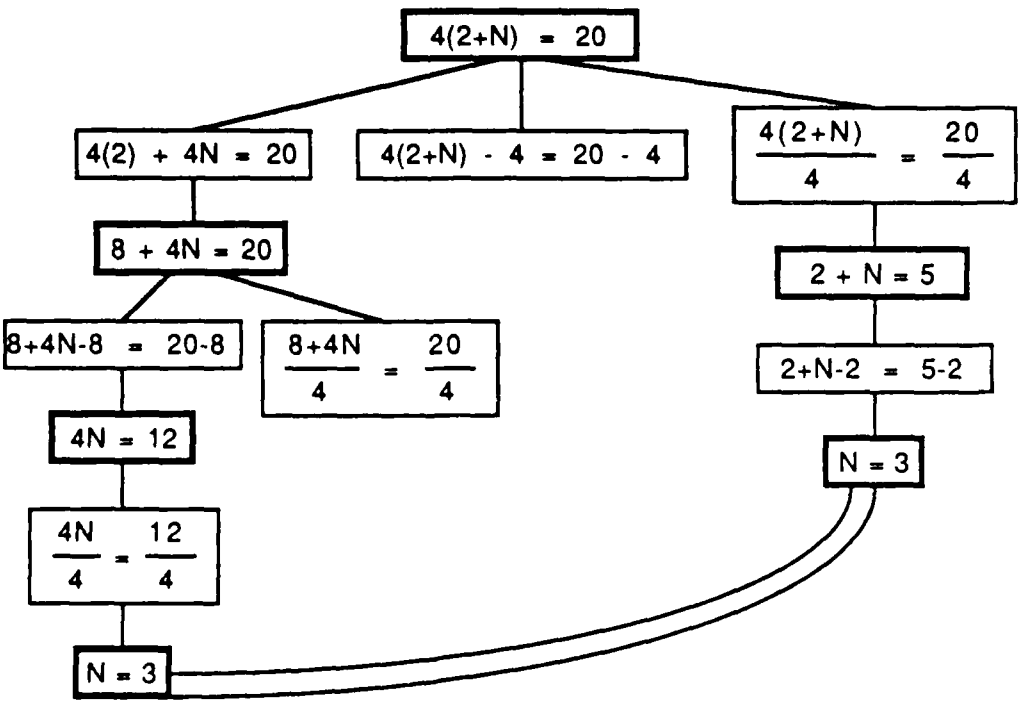


Figure 11. Display of the trace of a student's solution in Algebraland

Figure 11 shows a trace of a student's work on a problem, created by the Algebraland system. The trace shows the expressions that were formed by transformations that the student applied.

The transformations are chosen by the student from a menu of operations, shown in Figure 12. For example, the first operation in the left branch of Figure 7 was Distribute. To produce the expression in the middle branch, the student subtracted 4 from both sides. To reach the first expression in the right branch, the student chose to divide both sides by 4. Do-Arithmetic is the name given to a collection of operators that "clean up" expressions, simplifying them by cancelling numbers in the numerator and denominator of a fraction, or performing simple arithmetic calculations. Expressions that are obtained prior to use of Do-Arithmetic are enclosed in light rectangles, and darker lines are used to enclose expressions after Do-Arithmetic has been used. Double lines connect states in the solution that are identical.

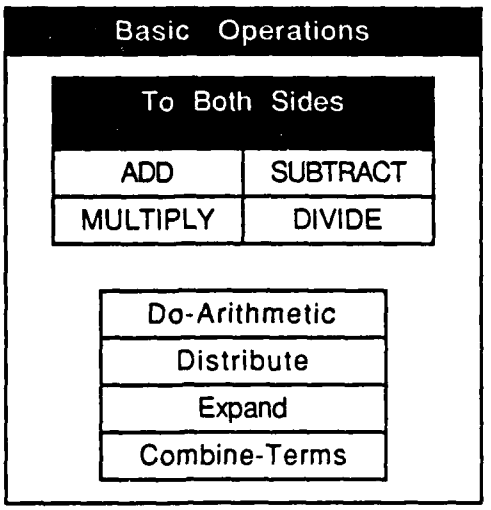


Figure 12. Window in Algebraland showing basic operations.

The display also provides a menu of planning concepts, shown in Figure 13. Concepts like these were suggested by Bundy (1975). The concepts correspond to general goals that can be used to organize problem-solving activity. In the Algebraland system, the plans are available for a student to use in annotating a trace of problem solving activity, either that the student has produced or that is provided for didactic purposes.

The opportunity to examine a trace of problem-solving activity and relate steps in the solution effort to strategic goals provides an unusual situation that encourages reflective learning. For example, a student could examine Figure 11 and consider which planning goal is furthered by each of the first steps that was attempted. The left branch achieved the goal of splitting an expression containing N. The center and right branches involved attempts to isolate the variable by removing a numerical term; the attempt in the center was unsuccessful for a technical reason. In this problem, the strategy of isolating the variable leads to a more efficient solution than splitting the expression, as can be seen from the solution expression requiring two steps (ignoring clen-ups) instead of three. The reason is that in the initial expression, N is a term without a coefficient, and distributing 4 through (2+N) results in a term, 4N, that then has to be reconverted to N. To appreciate this somewhat subtle feature probably requires an opportunity to reflect on the consequences of alternative actions of the kind that Algebraland affords.

PLAN MENU
ISOLATE the variable
COLLECT like terms in a single expression
GROUP together like terms (transpose terms)
SPLIT apart expressions containing the variable
SIMPLIFY the expression

Figure 13. Menu of planning concepts

Several theoretical distinctions have been clarified in the process of developing the Algebraland system. One issue involves the level of detail in characterizing algebraic operations. An operation called "Do-Arithmetic" is poorly specified by formal standards, and its use in this system is justified by its cognitive utility. An early version of Algebraland distinguished between different arithmetic operations such as cancelling a number in the numerator and denominator, replacing a pair of multiplied numbers by their product, and replacing a pair of added numbers by their sum. The effect of these distinctions was a set of operations that was clearly difficult for students to manage. By including an operation that seems poorly specified, a significant simplification of the system was achieved, allowing more successful use by students. Our hypothesis, based on this experience, is that operations like the ones we have included in "Do-Arithmetic" come to be relatively routine for students and do not enter significantly into their deliberate analysis and planning.

A second issue that has become clearer is the way in which goal states and operators can interact in a student's understanding. The representation of operators and strategic concepts in Algebraland was designed with the goal of enabling students to reflect on strategic aspects of problem solving. This required that the representation of operators and the problem-solving trace be at an appropriate level of detail to allow the learner to recall or reconstruct the goal states that occurred during problem solving. The decision to have Do-Arithmetic as a general operator was motivated by this consideration.

Thirdly, we also came to appreciate that efforts to relieve students of low-level cognitive demands may be less simple than they appear. Algebraland provides students with capabilities that allow them to avoid a great deal of symbolic computation. On the other hand, they must learn to manipulate the Algebraland system, which is demanding in itself. Some of the requirements are quite obvious, but some are interesting and involve

issues of representation in the domain, such as whether a student is required to supply arguments for an operation like combining terms.

Results of a Study of Acquisition of Error Management Skills. Algebraland provides a setting for studying processes of reflective learning of strategic concepts, and Foss is currently conducting such a study at Xerox PARC. This study focuses on the acquisition of skills in detecting, recovering from, and learning from errors. These skills are an integral part of acquiring and using a new procedure. They are particulaty important once the student is faced with new problems, removed from the tutorial situation.

Error management refers to skills needed for 1) *recognition* of error states or inefficient solution strategies ("thrashing") and 2) error *recovery*. To recover from an error, students need to apply some sort of *state selection heuristics;* i.e., they need to decide to either back up to some previous problem state in the search tree or to continue applying forward operators (e.g., distribute or combine terms) from the current state. Foss is investigating the possibility that such error management skills can be acquired by appropriate experiences with making, recovering from, and learning from errors in a "safe" situation, such as that produced by Algebraland.

Two aspects of error management were explored in a study with junior high-school and high-school students who solved problems with Algebraland. Foss was interested in both how students recognize error states and how they detect thrashing in the absence of feedback, and what they do in response to such discoveries.

Error states are defined as problem states that are off the optimal solution path. They are detected by determining if the number of operations separating the current and goal state is greater than the number separating than the number separating the previous (or initial) state and the goal. Introductory algebra students usually lack the knowledge needed

for such an evaluation function, however, so we suggest that they rely on heuristics that use simple visual information about the problem's surface features or other cues available in the problem-solving environment to detect errors. Many cues for monitoring progress are present in the problem -solving environment: the surfact appearance of the equation; the shape of the search tree; and the student's cognitive context such as the student's goal stack or the overall certainty the student has regarding a particular solution strategy.

In an initial study, 12 junior high-school and high-school students used Algebraland to solve a set of algebra equations. After they were instructed in the use of the system and were given a set of sample problems to solve with the guidance of an observer, they worked through a set of progressively difficult algebra problems. Afterwards, they were interviewed regarding their solution strategies for two of the problems. While viewing the search trees produced while working the problems, they were asked to describe what they did at each state (i.e., what operator was applied) and what they were trying to accomplish. Special attention was paid to branch points and back-up points -- i.e., they were asked why they decided to back up and why they selected a particular state to return to.

The students seemed to have a repertoire of informal rules for recognizing error states or thrashing. Table 10 presents a categorization of cues in the problem-solving environment that students reported using for detecting error states or thrashing. Points currently under consideration are 1) which cues are most effective (i.e., least likely to produce false alarms or misses); 2) how students' use of these cues changes with practice and expertise (e.g., there is probably a movement from emphasis on surface features to "deep features" such as active goals); and 3) how the cue attended to affects aspects of error recovery such as state selection (i.e., is there a particular type of "repair" or error recovery response associated with the rules used to detect errors?). Note that these cues can also be useful for deciding whether the current solution strategy is on the right track.

Foss suggests that these cues are used not only to detect error states, but that they also can trigger various error recovery strategies: backing up to some previous state, perhaps re-thinking the current strategy, or engaging in some sort of brute-force search for an operation that will move the equation closer to the goal state. One aspect of error recovery is state selection: choosing a problem state to retreat to or deciding to continue from the current state. In related tasks, students usually choose to either back up to the immediately preceding state or to start the problem over, which is not always the optimal thing to do since states closer to the goal may be on other branches. This could be due to some problem-solving set or it could be that it is too difficult to develop some sort of state evaluation function, or to re-establish the problem-solving context corresponding to an intermediate state. But perhaps the same cues that alert the student to errors can be used to aid in selecting a proper strategy for recovering from them.

Table 10

Cues Reported by Students for

Detecting Errors or Thrashing

Cues in the Physical Context

1) Surface Features of Equation: increase in equation's complexitity, e.g., number of terms around the target variable, number of variable's occurrences, placement of variable in the equation.

2) Repeated States: duplicate states and return to previous states (i.e., branch points).

3) Repeated application of inverse operators: e.g., add n, subtract n.

4) Noticing shorter path on other branch in search tree (often when equation gets messier than its initial state).

Cues that Require Cognitive Context:

5) Difficulty in deciding on appropriate operator (when too many seem applicable or none does)

6) Certainty factor drops below threshold (perhaps because the student used too many guesses for current problem).

7) Expectation Violation (when application of operator produced a surprising result).

8) Plan-Action Conflict: the actiions used to satisfy a goal were ineffective.

9) Conflicting Subgoals: earlier accomplished goals are undone as a result of satisfying the current subgoal.

10) Plan Evaluation Difficulties: cannot see ahead or evaluations is difficult for some other reason.

11) Approach seems too messy, too long, or too difficult.

As an example, if a student decides to back up because of conflicting subgoals, then the student knows that his or her goal or overall problem strategy should be revised. In a problem such as 5B + 17 = 7 - 3B, some students tried to solve the problem by setting two subgoals: 1) solve for the B on the left side, and solve for the B on the right side. After dividing both sides by 5 (thereby isolating the left-side B), then dividing both sides by 3/5 (thereby isolating the right-side B), then dividing both sides by 5/3 (thereby isolating the left-side B again), etc., the student realized that he was thrashing: satisfying one subgoal undid the other subgoal. At this point he backed up to the part of the search tree corresponding to the point just before he started isolating the variables individually.

Another alternative can occur if a decision to back up is based on some other cue, such as noticing that the equation is getting more complex. Then the student may simply try alternative ways of satisfying the current goal rather than making major revisions to the overall goal structure of the current solution method. This occurred at a point in a solution process where a student wanted to isolate W in 1/W = 5, but needed three tries before she was successful. After each try she backed up to the problem state corresponding to her current goal (1/W = 5) rather than completely starting over.

Learning to monitor progress in problem solving and to fix errors in the absence of external feedback is an important metacognitive skill that has not been studied or documented. By observing students using Algebraland Foss has been able to observe the development of a set of heuristics that students use to monitor their progress. These heuristics have implications for issues related to how much exploration of the search space is optimal (i.e., what can be learned from visiting error states) and the role of making and correcting errors in acquiring a skill.

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89,* 396-406.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. (1984). *Cognitive principles in the design of computer tutors* (Technical Report #ONR-84-1). Pittsburgh: Advanced Computer Tutoring Project, Carnegie-Mellon University.

Briars, D. J., & Larkin, J. H. (1984). An integrated model of skill in solving elementary word problems. *Cognition and Instruction, 1,* 245-296.

Brown, J. S., & Burton, R. B. (1980). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science, 4,* 379-426.

Bundy, A. (1975). Analysing mathematical proofs (or reading between the lines). In P. H. Winston (Ed.), *Proceedings of the 4th IJCAI,* Georgia, USSR.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 157-184). New York: Academic Press.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 79-98). New York: Academic Press.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5,* 87-119.

diSessa, A. (1982). Unlearning Aristotelian physics: A study of knowledge-based learning. *Cognitive Science, 6,* 37-75.

Greeno, J. G. (1978). A study of problem solving. In R. Glaser (Ed.), *Advances in instructional psychology* (vol. 1, pp. 13-75). Hillsdale NJ: Lawrence Erlbaum Associates.

Greeno, J. G., Magone, M. E., Rabinowitz, M., Ranney, M., Strauch, C., & Vitolo, T. M. (1985). *Investigations of a cognitive skill.* (Report 1985/27) Pittsburgh PA: University of Pittsburgh Learning Research and Development Center.

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

Kieran, C. (1980). The interpretation of the equal sign: Symbol for an equivalent relation vs. an operator symbol. *Proceedings of the Fourth International Conference for the Psychology of Mathematics Education* (pp. 163-169). Berkeley CA: Lawrence Hall of Science.

Kintsch, W., & Greeno, J. G. (1985). Understanding and solving word arithmetic problems. *Psychological Review, 92,* 109-129.

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

Matz, M. (1980). Towards a computational theory of algebraic competence. *Journal of Mathematical Behavior, 3,* 93-166.

Nesher, P., & Katriel, T. A. (1977). A semantic analysis of addition and subtraction word problem in arithmetic. *Educational Studies in Mathematics, 8,* 251-269.

Newell, A. (1982). The knowledge level. *Artificial Intelligence, 18,* 87-127.

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

Resnick, L. B. (1983). A developmental theory of number understanding. In H. P. Ginsberg (Ed.), *The development of mathematical thinking*. New York: Academic Press.

Riley, M. S. (1984). *Structural understanding in performance and learning* (doctoral dissertation). Pittsburgh: University of Pittsburgh.

Riley, M. S., Greeno, J. G., & Heller, J. I. (1983). Development of children's problem-solving ability in arithmetic. In H. P. Ginsburg (Ed.), *The development of mathematical thinking*. New York: Academic Press.

Schoenfeld, A. H. (1979). Explicit heuristic training as a variable in problem-solving performance. *Journal for Research in Mathematics Education, 10,* 173-187.

Schwartz, J. (1976). *Semantic aspects of quantity*. Cambridge MA: MIT Division for Study and Research in Education.

Shalin, V., & Bee, N. V. (1985a). *Analysis of the semantic structure of a domain of word problems*. (Technical Report No. UPITT/LRDC/ONR/APS-20). Pittsburgh PA: University of Pittsburgh Learning Research and Development Center.

Shalin, V., & Bee, N. V. (1985b). *Structural differences between two-step word problems*. (Technical Report No. UPITT/LRDC/ONR/APS-19). Pittsburgh PA: University of Pittsburgh Learning Research and Development Center.

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

VanLehn, K. (1983). *Felicity conditions for human skill acquisition: Validating an AI-based theory* (Report No. CIS-21). Palo Alto: Xerox Palo Alto Research Center.

Wagner, S. (1981). Conservation of equation and function under transformations of variable. *Journal for Research in Mathematics Educatiion, 12*, 107-118.

END

DTIC

8-86