The Structure of Generate-and-Test in Algebra Problem-Solving

Adisack Nhouyvanisvong (adisack@andrew.cmu.edu)
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

Irvin R. Katz (ikatz@gmu.edu)
Human Factors and Applied Cognition Program
George Mason University
Fairfax, VA 22030

Abstract

In this paper, we investigate students' use of the generate-and-test strategy to solve algebra word problems. This strategy involves first choosing an estimate for the answer and then checking whether the estimate satisfies the constraints of the problem. Based on verbal protocol data, we developed a production system model to simulate students' behavior when they apply this informal strategy. The model predicts problem features that should affect the difficulty of the problems. A large-scale experiment tested the predictions of the model. Verbal protocol data provided additional insights into how students use the generate-and-test strategy.

Introduction

What is the weakest way to solve a problem? Take a guess. Generate-and-test is a heuristic that at first blush may seem hopelessly unlikely to lead to solution. Yet in the domain of algebra word problems, generate-and-test is used effectively by many students. A form of this heuristic (called "guess and check") is even taught by some algebra instructors. In several studies of algebra problem solving, Berger & Katz (1995) found that participants (high school students) used generate-and-test on approximately 50% of problems presented by the researchers. Generate-and-test is not necessarily a "fallback" strategy for weaker students, but is used effectively by students with good math skills (Katz, Friedman, Bennett, & Berger, 1996; Tabachneck, Koedinger, & Nathan, 1995). Other researchers have similarly noted the prevalence of generate-and-test and other "informal" strategies (e.g., Hall, Kibler, Wenger, & Truxaw, 1989; Koedinger & Tabachneck, 1994). Yet generate-and-test is underrepresented in the problem solving literature, and practically unmentioned in research on algebra word problem solving. Much of the research on algebra word problem solving has focused on the formal representations and procedures used by students when solving traditional word problems (e.g., Kintsch & Greeno, 1985; Mayer, Larkin, & Kadane, 1984; Paige & Simon, 1966). These problems typically have one unique solution and can be solved using standard algorithmic techniques.

In the generate-and-test strategy, a student chooses a possible answer, then checks whether that answer satisfies the constraints of the problem. If the candidate answer does not fit, the student generates a new one.

The use of an informal strategy such as generate-and-test raises several theoretical issues. For example, how do problem solvers generate their initial estimate of a potential solution? If their first estimate is found to be incorrect, how do they update that estimate to get another (hopefully better) estimate? Another theoretical question is how problem solvers make the initial decision to use the generate-and-test strategy. Why do they choose generate-and-test when they could use the formal algebraic strategy they had been taught?

This paper presents a production system model of the generate-and-test strategy as it appears when solving algebra word problems. This model is consistent with observed solution procedures of students solving traditional algebra word problems. However, an explanatory model alone is not particularly strong evidence for one's account of human behavior. To provide stronger evidence, predictions derived from the model were tested in a large-scale experiment. A second experiment, involving collection of verbal protocols, suggests that generate-and-test may be a more complex skill than is captured in the model.

To investigate the generate-and-test strategy, we use a type of algebra word problem that engenders use of the strategy. Under-determined problems cannot be solved by a purely algebraic approach. Each problem is under-determined in that it cannot be represented as an algebraic equation with a single unknown. Instead, the problem solver poses example answers that fit the constraints presented in the problem. The next section describes under-determined problems in more detail. Following that discussion, we describe our production system model that accounts for solving both under-determined problems and well-determined problems (i.e., traditional word problems).

Under-determined problems

Under-determined problems do not provide all the information necessary to find a unique solution to a problem—i.e., it is impossible to apply a *standard* algebraic solution strategy to determine a correct solution. By algebraic strategy, we mean a strategy that allows the problem solver to isolate and solve for one unknown

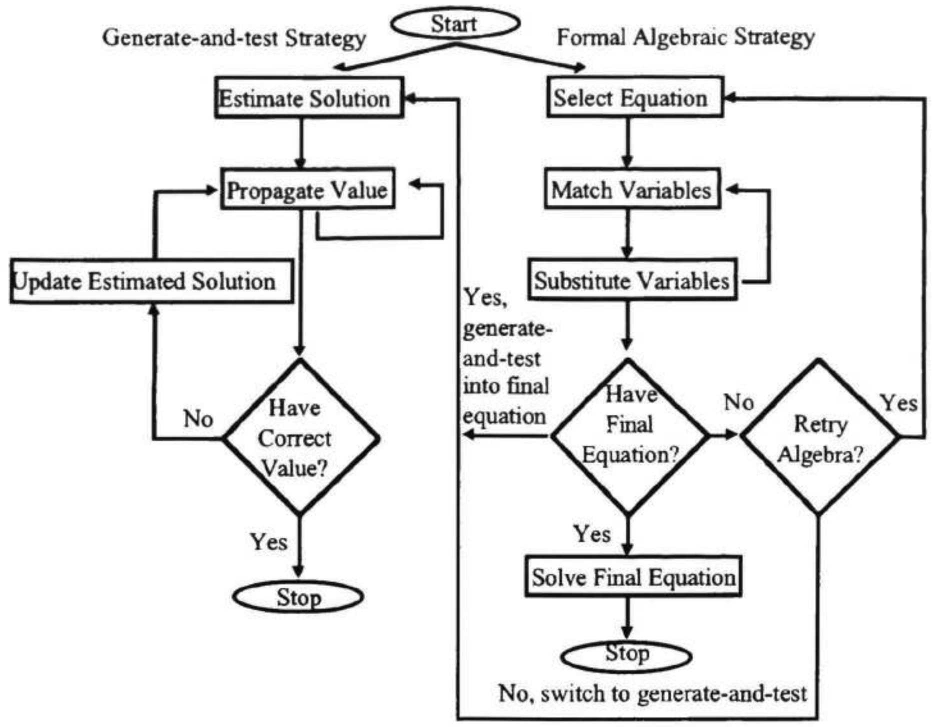

Figure 1. Model of the generate-and-test and formal algebra strategies.

variable (e.g., x = 22). Note some under-determined problems may involve inequalities that allow the problem solver to use algebraic manipulation of inequalities to isolate a variable with an inequality (e.g., x > 20). Note, this final derived inequality does not constitute a solution. The problem solver must still generate a solution based on that inequality. Thus, under-determined problems require problem solvers to generate examples of potential solutions. Below is an example of an under-determined problem:

If some tickets to a play were bought for a total of $50.00 and if tickets cost $1.00 for adults and $0.50 for children, how many children's tickets could have been bought?

The problem solver must produce an example value for the total number of children's tickets that satisfy a variety of explicitly stated constraints:

1. The total amount paid for the tickets is $50.00;
2. The cost of an adult's ticket is $1.00;
3. The cost of a child's ticket $0.50;

These constraints do *not* determine a unique solution to the problem. Several values for the total number of children's tickets satisfy these constraints.

A problem solver might proceed in solving the above problem by first identifying the explicit constraints given in the problem. The problem solver might then infer that, since the total amount paid is $50.00, the number of children's tickets must be even. Through this derivation of implicit constraints, the problem solver greatly reduces the space of possible examples to consider. Ultimately, however, taking into consideration all the constraints that have been identified, both explicit and implicit, the problem solver must propose a candidate solution and evaluate it against the constraints. A successful example is found if the proposed solution does not violate any problem constraint.

It is this reasoning with constraints that makes under-determined problems seemingly different from traditional, well-determined word problems. Unless the student detects, prioritizes, and instantiates the constraints by specifying appropriate values, the problems cannot be solved. Under-determined problems are potentially more realistic than traditional word problems, as most real problems are not well formulated (Frederiksen, 1984).

Computational Model

Our model simulates the behavior of students solving either well-determined or under-determined algebra word problems, using the generate-and-test strategy, an algebraic strategy, or a combination of the two. Our model was written in the ACT-R production system language (Anderson, 1993).

The general structure of the generate-and-test strategy came from protocols of 55 students solving well-determined algebra word problems, similar to those found on the SAT-Mathematics test. For two problems in particular, students tended to use a generate-and-test strategy to find a solution rather than an algebraic approach. This preference cuts across all ability levels, suggesting that generate-and-test is not necessarily a fallback strategy to be used only when one does not know a stronger way to approach a problem.

The model simulates the generate-and-test strategy by first estimating a potential solution. Then it propagates the value through the constraints of the problem. Propagation continues until all of the constraints have been used. The process terminates with success if the proposed solution simultaneously satisfies all of the constraints of the problem. If a discrepancy is found, the estimate is updated and the propagation process is repeated.

Data on the estimation and updating portions of the model were gathered from verbal protocols of seven students as they solved four problems similar to the "tickets" problem presented above (i.e., well-determined, simultaneous equation problems).[1] The number of generate-and-test "cases" was reduced (from 28) because students occasionally solved the problems algebraically, despite instructions. Of 16 cases in which students generated an initial estimate, students' estimates were half of the potential maximum value for the estimated element in nine of the cases. The other two-thirds of cases were not as easily categorized as reflecting a particular estimation strategy. The model implements this "half-of-total" initial estimation strategy.

If an estimate is found to be incorrect, the model simulates an update strategy observed in students. In this strategy, the estimate is changed based on the relation between a derived quantity and the known quantity. For example, a well-determined version of the tickets problem stated that "70 tickets were bought to a play." Through the propagation process, a student might derive a value of 80 for the total number of tickets (by first estimating 60 to the number of children's tickets, then deriving $30 for the cost of children's tickets, and then deriving $20 and 20 tickets for adult's tickets). The student would decrease the estimate of the children's ticket by 10. Alternatively, if the derived value were 60, the student would *increase* the estimate by 10. We call this strategy the "logic change rule". Note following this rule will not necessarily result in the correct choice. The correctness of the rule depends on the relation among the variables in the problem. Nonetheless, students do not realize this caveat and apply the rule indiscriminately. Of the 17 cases in which students updated their estimate at least once, on 11 occasions students' performance was consistent with this update approach. In other words, 63% of the time, students updated their initial estimates in a way consistent with the logic change rule.

The model also solves algebra word problems by applying a formal algebraic strategy. The strategy was derived from the protocols mentioned earlier. The model first selects an equation from the problem situation. It then simulates symbolic manipulation by matching and substituting variables. The model iterates until a single final equation with just one unknown variable is left or until all the equations have been used. Once a final equation is obtained, the answer can then be solved for. On the other hand, when a final equation cannot be obtained because the equation contains more than one unknown variable and there are no more values or equations to substitute in, the model either retries the algebra strategy or switches to the generate-and-test strategy.

Predictions

The model suggests the types of factors that should affect the difficulty of under-determined problems. For example, *requesting a second example* from a solver should be more likely to result in an error because of the additional processing needed. The additional processing provides the solver with more opportunities to commit math errors. Alternatively, a student might have happened upon a correct first example by chance, but does not understand the problem sufficiently to find a second example.

A second prediction is that *additional constraints* might increase difficulty, again because of predicted increased processing. However, the model suggests that only certain types of constraints will increase processing. "Verifier constraints" are used to verify whether a correct example solution has been derived. These constraints often involve a relation between two variables. According to our model, the use of such a constraint involves an extra iteration through the propagation process (see Figure 1). On the other hand, a constraint that just changes the range of possible estimates should not necessarily be more difficult because it affects a small part of the model. Such "generator constraints" involve a single variable that broadly suggests where a solution may be found. In the model, such constraints are used to estimate potential solutions. Thus, adding a generator constraint to a problem should not entail extra iteration, and so might not affect difficulty.

Experiment 1[2]

Method

Participants. Participants were 257 paid volunteers, including advanced undergraduates (54%), first-year graduate students (28%), and individuals not currently registered as students who intended to apply to graduate school (18%). Participants were recruited through 10 institutions of higher education in different regions of the United States.

Design and Materials. Two parallel test forms were created, each containing nine under-determined problems.[3] The first four problems represented a manipulation of the number of constraints provided. The treatment group received problems that contained one additional constraint compared with the problems given to the control group. The control and treatment problems were otherwise identical (see Table 1). In the first three problems, the additional constraint was a generator constraint; the final problem contained an additional verifier constraint. Note either a verifier or a generator constraint was added to each problem, not both. However, either a generator or a verifier constraint could have been added to a problem. For example, in the first problem of Table 1, a generator constraint could have been added by stating that the "company sold more than 1500 paperback books." The last five items represented a

[1] The algebra word problems were given to these participants as warm-up exercises for an unrelated experiment.

[2]This section describes an experiment reported in Bennett, Morley, Quardt, Singley, Katz, & Nhouyvanisvong (1998). We limit the current discussion to those portions of the experiment that test the computational model.

[3] As reported in Bennett et al. (1998), the test forms contained an additional 11 problems, representing two other manipulations.

Table 1. Problems showing the addition of a verifier versus generator constraint. Additional constraints are in bold.

Addition of a Verifier Constraint	
A company makes a profit of \$3.30 on every hardback book it sells and a profit of \$1.20 on every paperback book it sells. If it made a profit of \$3,960 on hardback and paperback books last month, how many books could it have sold last month?	A company makes a profit of \$3.30 on every hardback book it sells and a profit of \$1.20 on every paperback book it sells. Last month **the company sold more than twice as many paperback books as hardback books** and it made a profit of \$3,960 on the books. How many books could the company have sold last month?
Addition of a Generator Constraint	
A certain population of bacteria doubled every *d* hours. At *t*=60 hours the population was 1,024,000. What is a possible value for the initial bacteria population at *t*=0 and the corresponding value of *d*?	A certain population of bacteria doubled every *d* hours, **where d < 20**. At *t*=60 hours the population was 1,024,000. What is a possible value for the initial bacteria population at *t*=0 and the corresponding value of *d*?

manipulation of the number of solutions requested. These five problems were identical in the treatment and control conditions except that the treatment group was asked to provide two examples for each problem whereas the control group was asked to provide just one example.

Procedures. Test forms were randomly assigned to participants, with each participant receiving either the control or treatment problems. The math problems were administered on a computer. Participants were allowed to use scratch paper and a calculator during the 60-minute problem solving session.

Results

To test the effects of the additional constraint on performance, we ran a 2x2 repeated-measures ANOVA, with number of constraints (fewer constraints/more constraints) as a between-subjects factor and constraint type (generator or verifier constraint added) as a within-subjects factor.[4] The main effect of number-of-constraints was significant, $F(1,228)=9.05$, $p<.01$, as was the main effect of constraint type, $F(1,228)=10.32$, $p<.01$. Of primary interest was the significant interaction between number-of-constraints and constraint type, $F(1,228)=16.19$, $p<.01$. Figure 2 shows the interaction. As predicted, adding a generator constraint did not affect problem difficulty, but the addition of a verifier constraint does increase difficulty.

A one-way ANOVA revealed the predicted significant effect of number-of-examples, $F(1,234)=27.77$, $p<.01$. Participants in the one-example group solved more of the problems correctly than did participants in the two-example group, 3.3 and 2.3, respectively. Furthermore, a comparison of only the first solution from the two-example group with the one-example group did *not* reveal a significant difference between the mean number of problems correct, 3.1 and 3.3, respectively. This result suggests that the greater difficulty of the two-example problems comes from deriving the second example.

Discussion

This experiment investigated the effect of two problem features—number of constraints and number of requested examples—on problem difficulty. As predicted by the model, an additional constraint increases difficulty only if that constraint leads to more processing (i.e., verifier constraints). Generating a second example increases difficulty as well, again supposedly due to the greater processing needed to generate a second solution.

In the next experiment, we investigate whether the problem features had the expected effects for the reasons predicted by the model. That is, the model predicts that providing the second example solutions requires more iteration of the generate-and-test strategy and that *verifier* constraints affect difficulty because they affect the propagation process of the model. The second experiment seeks this evidence through an investigation of participants' solution processes as inferred from concurrent verbal protocols (cf. Ericsson & Simon, 1993).

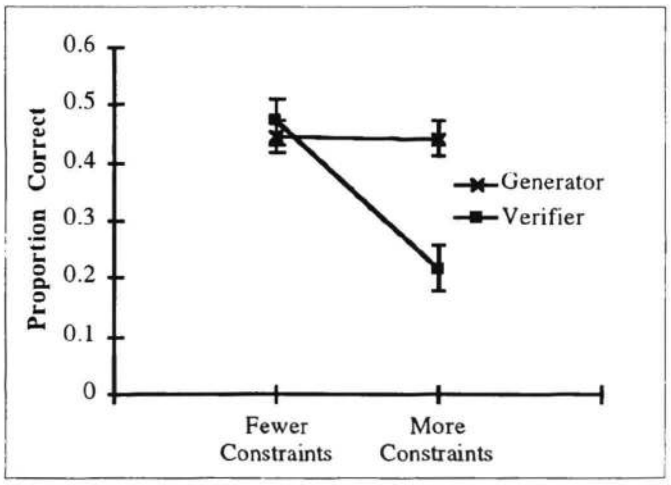

Figure 2. Proportion correct in control (fewer constraints) and treatment (more constraints) conditions.

Experiment 2

Method

Participants. Six college seniors and recent graduates from the Princeton, New Jersey area participated for pay.

Design and Materials. The design and materials were

[4] Analyses excluded cases for which one or more problem scores were missing (because the participant exceeded a time limit).

identical to those of Experiment 1.

Procedure. The volunteers participated individually in sessions lasting from 1-1.5hr. All problems were administered on paper, with a 10min time limit per problem. Participants were asked to provide concurrent verbal protocols as they solved the problems. All written work was videotaped and participant utterances were recorded on the videotape. Participants were provided with a simple calculator (analogous to the calculator provided in Experiment 1).

Results

The same general pattern of means was found as in Experiment 1. Addition of a generator constraint did not affect proportion correct (control: 0.67; generator constraint added: 0.67), whereas addition of a verifier constraint did impact difficulty (control: 0.67; verifier constraint added: 0.33). One-example problems were more difficult than two-example problems (one-example: 0.80; two-example: 0.67).

The analysis of participant solution procedures should address three questions: (1) Can the extra difficulty associated with two-example problems be attributed to participants re-applying the generate-and-test strategy? (2) Does the addition of a verifier constraint increase difficulty because of the extra processing involved in judging whether a potential solution fits all of the problem constraints? (3) Why do generator constraints not add difficulty?

Producing a second example. When asked at the end of the session whether providing two examples was difficult, a few of the participants expressed that "it wasn't difficult" because they "could just double the first answer." That is, for two of the five problems, simply taking a trivial transformation of the first response could generate a correct second example. For instance, to provide a second example, participants could simply double the first example (or take any multiple of it). For one problem, more elaborate (nontrivial) calculations were required.[5] This suggests that the difficulty between the one-example and two-example problems may be solely due to the nontrivial transformation problem.

The analysis did not reveal any difference in the mean proportion correct between the one-example and two-example trivial transformation problems. The comparable difference was .33 for the problem that required more complex calculations. Reanalysis of the data from Experiment 1 support these results. The difference in mean proportion correct between the one-example and two-example trivial transformation and nontrivial transformation problems was .14 and .39, respectively. These results support the claim that the added source of difficulty from providing two examples comes from having to recycle through the generate-and-test strategy. However, the extra processing occurs only for problems that do *not* allow a shortcut strategy (i.e., trivial transformation problems).

[5] The remaining two items could not be classified unambiguously and therefore are omitted from this discussion.

Adding a verifier constraint. Why does an additional verifier constraint increase problem difficulty? Of the six participants, three correctly answered this problem. They provided correct solutions by using the generate-and-test strategy. The three participants who did not provide correct examples used symbolic-like strategies by manipulating the constraints and values of the problem. Thus, it is unclear whether verifier constraints cause more iteration of processes as suggested by our model. Future research can determine whether verifier constraints cause more cycling of a strategy or lead to use of inappropriate strategies.

Adding a generator constraint. Analysis of the protocol data revealed that on 50% of the problems, participants in the control condition generated example solutions that satisfied the additional generator constraint in the experimental condition. In other words, participants' estimates were unaffected by the generator constraint.

Discussion

The analyses of verbal protocols suggests that the account of generate-and-test presented in Figure 1 is incomplete. Generating a second example sometimes involves reasoning that falls outside the generate-and-test or formal algebra strategies. In addition, the difficulty of the verifier constraint may be due to its encouraging an inappropriate solution strategy rather than because it causes more processing in a generate-and-test strategy. Finally, the analysis of participants' estimates suggests that different manipulations of the generator constraints may, in fact, affect difficulty. The current generator constraints suggest solution ranges that overlap with participants' natural propensities. Problems may be made more difficult (or easier) by adding generator constraints that "push" participants into estimation ranges that have a lesser (or greater) density of acceptable solutions.

General Discussion

Our production system model illuminates the cognitive processes involved in solving both under-determined and well-determined algebra word problems. The model simulates the processes involved in performing the generate-and-test strategy for both well-determined and under-determined problems. It also simulates the symbolic manipulation of constraints, an integral skill of the formal algebraic strategy.

In addition to accounting for the solution paths, the model correctly predicted factors that affect the difficulty of under-determined problems. We found that providing to examples is more difficult than one and that adding a constraint that influences the propagation process of the model affects difficulty. The protocols revealed that the source of difficulty can be attributed to elaborate calculations to derive the second example and not to short-circuiting of this process. Although this insightful strategic behavior of short-circuiting the process does not work on all problems, it can be an effective strategy on other problems. Future research should investigate how students decide to use this strategy. Based on that research, we can revise and extend our model to account for this behavior.

Currently, our model also does not account for the strategy choice between generate-and-test and formal algebra. Recall, our research only used under-determined problems. Consequently, the formal algebraic strategy could not be implemented to solve them. To force the model to use generate-and-test, the probability of the processes involved leading to eventual success is set higher for productions involved in the generate-and-test strategy. For example, the model will start to generate and test if we set the probability of the production that estimates a potential solution to be higher than the probability of the production that selects an equation. Conversely, if we reversed the probabilities, the model will start the formal algebraic strategy by selecting an equation, instead of estimating a potential solution. Thus, in our simulations of under-determined problems, we set the probability of estimating a solution higher than selecting an equation (see Figure 1).

This probability dependent decision process is analogous to a low ability student always choosing one strategy over the other strategies irrespective of the problem type. The student might not be mathematically sophisticated and consequently lacks the skills (the production rules) to perform the alternative strategies. On the other hand, a high ability math student will possess the skills to solve a problem successfully through numerous means. Consequently, the student must decide which strategy to implement. This decision to select one strategy over the others might be dependent on problem features. For example, factors such as the number of variables, equations, and type of equations in a problem may affect this decision process. Future research comparing under-determined problems to their well-determined counterparts should provide us with a better understanding of how the choice between using generate-and-test and formal algebra is made.

In addition to developing the strategy selection component, we plan to build ability-referenced models and devise metrics of problem difficulty. Through the inclusion or omission of certain productions as well as certain buggy productions, we can produce models that simulate different mathematical ability levels. As alluded to earlier, a low ability student may lack the skills to match and substitute variables. Analogously, a low ability model may lack the necessary production rules to perform these procedures. Another variant of a low ability model may be to include buggy productions that manipulate the problem information (declarative memory elements) incorrectly or apply them at inappropriate times. High ability mathematics students may possess more math knowledge and skills than low ability students. Thus, a high ability model may include more production rules to reflect the advanced knowledge and skills of a high ability student.

The model simulation output can provide us with metrics of problem difficulty. In the simplest case, the number of production cycles (i.e., the number of times the productions fired) may be predictive of problem difficulty. Another predictor may be the number and kind of productions that fired. This metric not only reveals the skills that are necessary, it also discloses the skills that students predominantly use to solve the problem. Knowing the skills more likely to be employed for a certain problem may lead us to better predict the difficulty of the problem.

Acknowledgments

This work was funded by the GRE Board and Educational Testing Service. We thank Erik Altmann and Audrey Lipps for their comments on previous drafts of this paper.

References

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Bennett, R., Morley, M., Quardt, D., Singley, M., Katz, I. & Nhouyvanisvong, A. (1998). *Generating examples: A new response type for measuring quantitative reasoning* Manuscript submitted for publication.

Berger, A. E., & Katz, I. R. (1995, April). *Solving mathematics word problems: Problem features affect strategy choice.* Paper presented at the Annual meeting of the American Educational Research Association, San Francisco, CA.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data, 2nd edition.* Cambridge, MA: MIT Press.

Frederiksen, N. (1984). The real test bias: Influences of testing on teaching and learning. *American Psychologist, 39*, 193-202.

Hall, R., Kibler, D., Wenger, E., & Truxaw, C. (1989). Exploring the episodic structure of algebra story problem solving. *Cognition and Instruction*, 6(3), 223-283.

Hinsley, D., Hayes, J. R., & Simon, H. A. (1977). From words to equations: Meaning and representation in algebra word problems. In P. Carpenter & M. Just (Eds.), *Cognitive processes in comprehension.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Katz, I. R., Friedman, D. F., Bennett, R. E., & Berger, A. E. (1996). *Differences in strategies used to solve stem-equivalent constructed response and multiple choice SAT mathematics items.* (ETS Research Rep. No. RR: 96-20). Princeton, NJ: Educational Testing Service.

Koedinger, K. R., & Tabachneck, H. J. M. (1994, April). *Two strategies are better than one: Multiple strategies use in word problem solving.* Paper presented at the Annual meeting of the American Educational Research Association, New Orleans, LA.

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

Mayer, R. E., Larkin, J., & Kadane, J. (1984). A cognitive analysis of mathematical problem-solving ability. In R. Sternberg (Ed.). *Advances in the psychology of human intelligence.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Paige, J. M., & Simon, H. A. (1966). Cognitive processes in solving algebra word problems. In B. Kleinmuntz, (Ed.), *Problem solving: Research, method, and theory.* New York: Wiley.

Tabachneck, H. J. M., Koedinger, K. R., & Nathan, M. J. (1995). A cognitive analysis of the task demands of early algebra. Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society (pp. 397-402). Hillsdale, NJ: Lawrence Erlbaum Associates.