STYLES OF THINKING: FROM ALGEBRA WORD PROBLEMS TO PROGRAMMING VIA PROCEDURALITY[1]

Kate Ehrlich
Elliot Soloway
Valerie Abbott

Department of Computer Science
Yale University
P.O. Box 2158
New Haven, Connecticut 06520

1. ABSTRACT

Algebra word problems are often surprisingly hard for college students to solve. However, more students are able to solve these problems correctly when asked to write a computer program, than when asked to write an equation. We have also found that programmers, with the same level of math experience as non-programmers, do consistently better on the algebra word problems, after only one semester of an introductory programming class. We argue that some of the difficulty associated with the algebra word problems can be traced to misconceptions about what the algebraic expression represents. Students often appear to use an algebraic expression as if it were a static description rather than as denoting an active operation being performed by one number to get another number. Although programmers may be equally prone to such misconceptions, it seems that experience with programming helps them to overcome these misconceptions, by encouraging them to develop a more active, procedural view of the problem.

2. INTRODUCTION

In recent work, Clement and Lockhead [Clement, Lochhead, and Monk, 1980] have demonstrated that there is a class of apparently simple algebra word problems that students find very difficult to solve correctly. A typical problem is shown as Example 1 in Table 2-1.

When Clement et al gave this problem to a group of engineering students they found that only 63% of the group gave the correct response of S = 6P. The most common wrong answer was: 6S = P, the reversal of the correct answer. In another problem, also shown in Table 2-1, in which there are two integrals, only 27% of the class were able to produce a correct answer. These findings are very robust and have been replicated in a number of studies (e.g. Soloway et al, 1982; Clement et al, 1980; , Kaput, 1979).

Clement and his colleagues [Soloway et al., 1982, Clement, Lochhead, and Monk, 1980]. carried out videotaped interviews with some of these students to try and find the source of the errors They identified two principal kinds of strategies that students were using to solve the problems. Some students used a syntactic, word order matching strategy, in which the order of key words, such as "student" and "professor" and the numbers from the problem description, were mapped directly onto the order of symbols appearing in the equation. Paige and Simon [Paige and Simon, 1966] have also argued for the weaknesses inherent in this kind of direct translation strategy.

Another strategy that students adopted can be characterized as "static comparison". For instance, one student described the equation in the following manner:

> There's six times as many students, which means it's six students to one professor and this (points to 6S) is six times as many students as there are professors (points to 1P).

What's wrong with these strategies? Their main problem is that students seem to have a static, descriptive view of algebra. For instance, students who espouse the strategy denoted as "static comparison" seem to want the algebraic expression to represent directly the relative sizes of the objects in the problem. In so doing, they treat the variables such as S and P as standing for "students" or "professors" rather than for the *number* of students or the *number* of professors. But, algebra does not function as a description in the same way as English provides descriptions. The correct equation, S = 6P does not describe the sizes of the groups, rather it denotes an equivalence relation that would obtain if one of the groups, the professors, were made six times larger. In this way, the algebraic expression represents an active operation that is performed on one number to obtain another number.

3. IMPACT OF PROGRAMMING: PROCEDURALITY

If the correct conception of algebra is an active, procedural one, then putting students in an environment that encourages them to adopt a more procedural approach should help them to generate correct solutions to the algebra word problems. Programming is such an environment. Indeed, Papert [Papert, 1971] has claimed for some time that learning to program can enhance problem solving skills.

In previous research [Clement et. al., 1980, Soloway et al., 1982], we found that significantly more students could solve the problems correctly when the problem was presented in the context of writing a computer program than in the context of an equation. We have also conducted videotaped interviews with some of the students who were unable to write the equation [Soloway et al., 1982]. In several cases, we found that the same student was able to solve the problem in the context of a computer program but not in the context of an equation, even when there were only a few minutes separating the two solutions. These results support the claim that it is easier to write a program to solve a certain class of problems that to write an equation.

4. PROGRAMS VS EQUATIONS: THE CONTRIBUTION OF PROCEDURAL WORDING

In the study reported in [Soloway et al., 1982], the instructions for the two versions of the problem are worded a little differently. In particular, the instructions for the program version are themselves more procedural than the instructions for the equation version. Thus, it may be that the critical factor in the study was the wording of the instructions rather than any difference between writing an equation or a program. If the wording was the critical factor, however, there should be a difference between the two kinds of wording for non-programmers as well as programmers.

In the new study we used three versions of the algebra word problem; these are shown in Table 4-1. The equation and the program version are the ones used in the previous study; the

[1]This work was supported by the National Science Foundation, under NSF Grant IST-81-14840.

function version is new. We ran this study with students who had no programming experience as well as with students who had taken at least one programming course. The programmers received all three versions, while the non-programmers were given the equation and the function versions of the problem. Each student saw only one version of the problem.

The data, which are shown in Table 4-2, show that the procedural wording of the instructions had no effect on accuracy for the non-programmers. The programmers, on the other hand, did write more correct equations when given the procedural instructions than when given the original, equation version. As in the previous study, there was also a significant improvement for writing programs over writing equations with non-procedural instructions. The results show that the procedural wording of the instructions only improves performance if students have had programming experience. One implication of these results is that procedural wording alone is not sufficient to induce people to adopt a more active view of algebra; people need experience in a procedural domain such as programming.

5. TRANSFER EFFECTS FROM PROGRAMMING TO ALGEBRA

The results of the previous study suggest that it is experience with programming rather than the procedurality of the instructions that is critical. In the next study we examined more directly whether programmers do better on the algebra word problems than non-programmers, when the problems are presented in the standard non-procedural context.

We constructed a large diagnostic test containing 17 algebra word problems as well as filler items. The test was administered to 28 people with no programming experience and 32 people who had just completed a semester of an introductory programming course. The groups were equated for level of math experience and in many other respects had similar academic backgrounds. Many of the people taking the programming course had majors in non-scientific subjects such as History or English, while some of the non-programmers had majors in fields such as psychology which includes some math experience in the form of statistics.

EXAMPLE 1

Given the following statement:

"There are six times as many students as professors at this University"

Write an equation to represent the above statement. Use S for the number of students and P for the number of professors.

- Result: 63% correct
- Typical wrong answer: 6S = P

EXAMPLE 2

Given the following statement:

"At Mindy's restaurant, for every four people who order cheesecake, there are five people who order strudel."

Write an equation to represent the above statement. Use C for the number of cheesecakes ordered and S for the number of strudels ordered.

- Result: 27% correct
- Typical wrong answer: 4C = 5S

Table 2-1:
EXAMPLES OF ALGEBRA WORD PROBLEMS

All the problems were presented in a non-programming context and none of the problems had procedural wording. Over the 17 problems, the non-programmers got an average of 64.5% of the problems correct while the programmers got an average of 75% of the problems correct. This difference between the groups was significant ($t = 4.7$, $p < 0.0005$). Although the average performance between the two groups differed by only 10%, the significance of the difference reflects a small but consistent improvement over all the problems for the programming group.

It may be argued, that although we controlled for level of math experience, and academic background, the programming group as a whole were smarter than the non-programmers. If this is the case, we should expect to find a fairly constant rate of improvement over all the problems. However, the data do not support that argument. Some sense of the kind of advantage conferred by programming experience can be illustrated by examining one set of problems that were included in the test.

There are three main forms in which the solution equation can be expressed. It can be expressed as a multiple , e.g. 5C = 4S; as a ratio, e.g. C/S = 4/5; or with a single variable on one side, e.g. C = 4/5 S. The wrong solutions are most often expressed in the form of a multiple, the ratio is the form students seem most familiar with, and the third form is the one appropriate to the equation written in a computer program. We included in the test, a set of problems in which people were given solution fragments in each of these three forms.

The percent correct completions for the two groups of subjects are shown in Table 5-1. When we compared performance on each version of the problem across the two groups, we found that there was no reliable difference between

PROBLEM

"At Mindy's restaurant, for every four people who order cheesecake there are five people who order strudel."

1. EQUATION

Write a mathematical equation to represent the above statement.

2. PROGRAM

Write a computer program which can be used to calculate the number of cheesecakes ordered when supplied with the number of strudels ordered.

3. FUNCTION

Write a mathematical function which can be used to calculate the number of cheesecakes ordered when supplied with the number of strudels ordered.

Table 4-1:
EXAMPLES OF WORDING

NON-PROGRAMMERS

	FUNCTION	EQUATION
CORRECT	82	73
INCORRECT	40	49

Equation vs Function: N.S.

PROGRAMMERS

	FUNCTION	EQUATION	PROGRAM
CORRECT	71	48	77
INCORRECT	32	53	22

Equation vs Function: $p < 0.01$
Equation vs Program: $p < 0.001$
Function vs Program: N.S.

Table 4-2:
The number of people given each problem type who produced a correct and incorrect solution

	NON-PROG	PROG
(multiple) ? C = ? S	36%	50%
(ratio) ? C = - - ? S	68%	78%
(single letter) C = ? - S ?	36%	59%

Table 5-1: EQUATION FRAGMENT:
Percent correct responses for each solution type

the programmers and the non-programmers except on the fragment that had the form of a single letter on the left hand side ($\chi^2 = 3.35$, $p < .10$). These data mitigate against claims that the programmers may have done better because they were smarter. Moreover, the data suggest that experience with programming confers quite specific problem solving skills to other domains such as algebra word problems.

6. CONCLUSIONS

There are a number of reasons why programming may enhance certain problem solving abilities. These reasons range from the explicitness required by the syntax of programming languages, through to the practice of "debugging" and number checking that is encouraged in programming. However, perhaps the main benefit of programming is that it provides the student with a model of an active input/output transformation which functions as a metaphor of change. It seems clear that people should be encouraged to develop skills that help them to construct these kinds of models. The results of the studies we reported, suggest that these skills are best developed in the context of learning to program.

References

Clement, J., Lochhead, J., and Soloway, E. . ***Positive Effects of Computer Programming On Students' Understanding of Variables and Equations.*** Proceedings of the National ACM Conference, Nashville, Tenn., 1980.

Clement, J., Lochhead, J., and Monk, G. Translation Difficulties in Learning Mathematics. *American Mathematical Monthly*, 1980, *88(4)*, 26-40.

Kaput, J. Mathematics and Learning: Roots of Epistemological Status. In J.Lochhead & J. Clement (Eds.), *Cognitive Process Instruction*, : Franklin Institute Press, 1979.

Paige, J. and Simon H. Cognitive Processes in Solving Algebra Word Problems. In *Problem Solving Research, Method and Theory*, : John Wiley and Sons, New York, 1966.

Papert, S. ***Teaching Children to be Mathematicians Versus Teaching About Mathematics.*** Technical Report 249, MIT AI Lab, 1971.

Soloway, E., Lochhead, J., Clement, J. Does Computer Programming Enhance Problem Solving Ability? Some Positive Evidence on Algebra Word Problems. In R. Seidel, R. Anderson, B. Hunter (Eds.), *Computer Literacy*, New York, NY: Academic Press, 1982.