Positive Effects of Computer Programming On Students' Understanding of Variables and Equations

John Clement *
Jack Lochhead *
Elliot Soloway **

* Cognitive Development Project
Department of
Physics and Astronomy
University of Massachusetts
Amherst, Mass., 01002

** Department of
Computer and Information Science
Unversity of Massachusetts
Amherst, Mass., 01002

ABSTRACT

There is a common intuition among those in computer science that programming helps to develop good problem solving skills. Our work has attempted to isolate the specific factors in programming which enhance mathematical problem solving ability. We have found that a surprising number of college students have difficulty with very simple algebra word problems. However, significantly more students are able to solve these word problems correctly in the context of writing computer programs, than in the context of simply writing an algebraic equation. We obtained similar results in comparing the reading of algebraic equations within computer programs and the reading of algebraic equations by themselves. Computer programming apparently puts an emphasis precisely on the active, procedural semantics of equations that many students lack.

I. Introduction

There is a common intuition among those in computer science education that computer programming encourages the development of good problem solving skills. Papert [1971] and the LOGO project were early proponents of this view; they developed a method to teach geometry by way of computer programming. Underlying their view is a recognition of the importance of "doing," of activity, and of procedure. Educators from Dewey to Piaget have emphasized that in order to understand a concept students need to take an active role.

This pedagogical intuition needs to be investigated empirically so that it can be articulated more precisely. A step in that direction has been made recently by Howe, O'Shea, and Plane [1979] in a series of experiments based on the following paradigm: a course in mathematics is taught in the standard way without incorporating computer programming, and simultaneously, the same course is taught with computer programming. Students' mastery of the subject matter is then compared across the two groups. In such experiments there seems to be a consistent effect in favor of incorporating computer programming

The above work might be characterized as experiments on the "macro" level; in contrast, the work reported here has focussed on the "micro" level. That is, we have attempted to develop <u>tools</u> which would enable us to isolate specific, critical factors

This research was supported in part by NSF Grant SED78-22043 in the Joint National Institute of Education - National Science Foundation Program of Research on Cognitive Processes and the Structure of Knowledge in Science and Mathematics.

This research was also supported in part by a grant from the U.S. Army Research Institute for the Behavioral and Social Sciences, ARI Grant No. DAHC-19-77-G-0012.

Any opinions, findings, conclusions or recommendations expressed in this report are those of the authors, and do not necessarily reflect the views of the U.S. Government.

contributing to the above results. Thus, rather than studying an entire course, we have focussed on single problems. We shall present results (section II) concerning the surprisingly poor performance of college students on two ostensibly simple algebra word problems. These results suggest several hypotheses. One is that the errors resulted from the students' failure to give a procedural interpretation to the algebraic equation. A second set of experimental results (section IV) provides significant, new support for this hypothesis. Namely, students do signficantly better on certain algebra word problems when they occur in a programming context, than when the same problems occur in a traditional, algebraic (non-programming) context. We go on to suggest several aspects of programming which could account for the way in which this activity fosters a more active interpretation of algebra by students. We conclude with a description of future research which we hope will further explicate the benefits of programming.

II. Experiments with Word Problems in a Traditional Algebraic Setting

In a previous study, Clement, Lochhead, and Monk [1979] uncovered two ostensibly simple problems with which students had great difficulty. In Table 1 we list the two problems and the performance results gathered from administering them to a group of 150 freshman students at a major state university. Fully 37% missed the first problem while 73% missed the second! Even more disturbing is the fact that all the students in this sample were engineering majors. Difficulty with algebraic manipulation did not seem responsible for these results; almost all students answered correctly problems which tested for this skill [Clement, Lochhead, Soloway 1979]. Nor do we feel that the students' difficulty could be explained by saying that the problem contained "tricky wording." Evidence against this view stems from the results obtained on problems such as 3 in Table 1 (also given to engineering majors). Here students are given a picture description of the problem and asked to write an algebraic equation; this "non-language" problem was missed by 68% of the students. Finally, we note that colleagues at two other colleges and universities have tested similar groups and obtained comparable results [Kaput 1979a, Monk 1979].

The errors made on problems 1 and 2 were largely of one kind; in both cases 68% of the errors were "reversals": 6S = P instead of S = 6P and 4C = 5S instead of 5C = 4S. The consistency of these error patterns argues against the idea that such errors were caused simply by carelessness. This idea is also discounted by the fact that roughly half the subjects were given the following hint with both problems.

> "Be careful: some students put a number in the wrong place in the equation."

This hint did not have a significant effect; it was associated with an increase in the percentage of correct solutions by only 3% and 5% respectively.

III. Interpretation of Algebra Experiments

How is it possible for students with such weaknesses to survive high school and college science courses? It appears that these students have developed special purpose translation algorithms which work for many textbook problems, but which do not involve anything that could reasonably be called a semantic understanding of algebra. Many word problems are constructed so that they can be solved through a trivial word-to-symbol matching algorithm. Others, such as physics problems, are given in a highly restricted context, where there are only two or three pretaught equations to choose between. This choice can be made either by picking the one equation which contains all of the given variables or through units analysis. While these techniques may be partially successful in many classroom situations, they are too primitive and unreliable to be trusted in any but the most routine applications.

In order to pursue the source of these errors, we conducted audio and video-taped interviews with 20 students who were asked to think out loud as they worked these and other related problems. On the "Students and Professors " problem we were able to identify two strategies which led to the reversal error. In the first, the student simply assumed that the order or contiguity of key words in the English language problem statement mapped directly into the order of symbols appearing in the algebraic equation. For example, one student wrote 6S = P and explained:

> "Well, the problem states it right off: '6 times students.' So it will be six times S is equal to professors."

In this type of strategy, the student appears to be using the syntax of the English problem statement --- and not an understanding of the problem itself --- on which to base his/her translation process. Weaknesses in this type of direct translation strategy have previously been analyzed by Paige and Simon[1966].

On the other hand, in a second incorrect strategy, students acted as if they did utilize an accurate representation of the meaning of the problem. However, reversal errors appeared to arise because of confusion about the semantics of the algebraic equation. For example, one subject wrote '6S = 1P' and explained:

> "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)."

When asked to draw a picture to illustrate his equation, the student drew from right to left one circle with a 'P' in it, an equal sign, and six circles with "S's" in them. Subjects such as the

Problem 1:

Write an equation using the variables S and P to represent the following statement: "There are six times as many students as professors at this University." Use S for the number of students and P for the number of professors.

Sample Size	% Correct	% Incorrect
150	63	37

Problem 2:

Write an equation using the variables C and S to represent the following statement: "At Mindy's restaurant, for every four people who order cheesecake, there are five people who ordered strudel." Let C represent the number of cheesecakes and S represent the number of strudels.

Sample Size	% Correct	% Incorrect
150	27	73

Problem 3:

Spies fly over the Norun Airplane Manufacturers and return with an aerial photograph of the new planes in the yard.

They are fairly certain that they have photographed a representative sample of one week's production. Write an equation using the letters R and B that describes the relationship between the number of red airplanes and the number of blue planes produced. The equation should allow you to calculate the number of blue planes produced in a month if you know the number of red planes produced in a month.

Sample Size	% Correct	% Incorrect
34	32	68

Table 1

above seem to use an accurate model of the practical situation, but they still fail to symbolize that understanding with the correct equation.

Apparently such subjects interpret the reversed equation, '6S = P', as stating that a large group of students are associated with a small group of professors. To these students the letter "P" stands for "a professor" rather than "the number of professors" and the equal sign expresses a comparison or association rather than an equivalence. The fact that the "S" side of the equation has a "6" on it indicates that it is larger than the "P" side which has no modifier. Thus, there appear to be more S's than there are P's. Thus the student attempts to write the algebraic equation '6S = P' as a "figurative" statement, describing a passive picture in which relative sizes of the entities are represented.

This contrasts to the correct equation 'S = 6P', which needs to be viewed as expressing an active operation being performed on one number (the number of professors) in order to obtain another number (the number of students). The correct equation, S = 6P, does not describe sizes of the groups in a literal or direct manner. Rather, it describes an equivalence relation that would occur if one were to make the group of professors six times larger. In other words, the equation S = 6P is not a direct description of the actual situtation, but rather, it represents the hypothetical state of affairs which would result after performing the operation of multiplying the current number of professors by 6. While some students find the correct equation through trial and error by writing the reversed equation first and then plugging in numbers as a check, our analysis of protocols from successful solutions indicates that the key to fully understanding the correct translation lies in viewing the number six as an operator which transforms the number of professors into the number of students. One subject who correctly wrote S = 6P said:

> "If you want to even out the number of students to the number of professors, you'd have to have six times as many professors."

The equation is thus interpreted in a procedural manner as an instruction to act.

In the above analysis, a comparison was made between two ways of viewing equations. While the distinction may be subtle, it is nonetheless critical. We stress this issue, since, as mathematically literate adults, it is difficult to imagine not viewing an equation as specifying operations on variable quantities. Nonetheless, our interview data suggest that this viewpoint is abstract and elusive for many students.

IV. Computer Programs vs. Algebraic Equations: Experimental Results

On the basis of the foregoing analysis, we developed the following hypothesis: if students were placed in an environment which could induce them to take a more active, procedural view of equations, then the error rate on these problems should go down. One clear candidate for such an environment is that of computer programming. That is, a computer program is a definite prescription for action; it is a set of commands which produces some result. Below, we present empirical tests of this hypothesis; in the next section we shall present our analysis of these results.

Experiment 1

In this experiment, our subjects were 17 professional engineers, with 10 to 30 years experience, who were taking a one week intensive course on the BASIC programming language. At the beginning of the first day of the course, before any instruction had begun on BASIC, they were asked to solve problem 1 in Table 2. We were suprised to find that 47% of these practicing engineers missed this problem! On the second day of the course, after the students had written and run programs using assignment statments, conditional statements, and for-next loops, and without any discussion of the answers to the above questions, the students were asked to solve problem 2 in Table 2. All subjects answered this question correctly using the statement LET B = (11*H)/6 (or some variant) in their program. Note that the form of this statement is equivalent to that of the correct answer to the first equation. Although this result could conceivably have been due to a "practice effect" from having done the previous problem, we strongly suspect that such an effect alone could not be responsible for so large a jump in performance.

Experiment 2

In this experiment, our subjects were primarily freshmen and sophomores in a course on machine and assembly language programming. This time, however, half the class was given problem 1 in Table 3, while the other half was simultaneously given problem 2 in Table 3. The only difference in the questions is that the latter asks for a computer program while the former asks for an algebraic equation. As indicated in Table 3, significantly more students could solve problem 1 than could solve problem 2. Probability of these results on the assumption that errors on each problem were equally likely is $p < .05$.

Experiment 3

The above 2 experiments explored the writing of computer programs or equations. However, in the study mentioned earlier, Clement, Lochhead and Monk [1979] observed that reading equations also gave students a great deal of trouble. That is, many students failed to write a correct explanation of the relationship expressed by the equation. Following the hypothesis outlined above, we wanted to compare the results of students reading and explaining an equation, which was embedded in a computer program, with students reading and explaining an equation, which stood alone. The two questions in Table 4 were given as part of an 11 question test to 87, mostly freshman, engineering students. The difference between the groups which answered one correctly but the other incorrectly is quite interesting. Namely, the group of students who answered the computer problem correctly (problem 2, Table 4), but the equation problem incorrectly (problem 1, Table 4) was more than 3 times as large as the group who answered the equation problem correctly, but missed the computer problem. This difference is significant at the .005 level. Here again, we see that the programming environment facilitated the students' understanding.

Problem 1:

Given the following statement:

"At the last football game, for every 4 people who bought sandwiches, there were 5 who bought hamburgers."

Write an equation which represents the above statement. Use S for the number of people who bought sandwiches, and H for number of people who bought hamburgers

Sample Size	% Correct	% Incorrect
17	53	47

Problem 2:

Given the following statement:

"At the last company cocktail party, for every 6 people who drank hard liquour, there were 11 people who drank beer."

Write a computer program in BASIC which will output the number of beer drinkers when supplied (via user input at the terminal) with the number of hard liquour drinkers. Use H for the number of people who drank hard liquour, and B for the number of people who drank beer.

Sample Size	% Correct	% Incorrect
17	100	0

Table 2

Problem 1:

Given the following statement:

"At the last company cocktail party, for every 6 people who drank hard liquour, there were 11 people who drank beer."

Write a computer program in BASIC which will output the number of beer drinkers when supplied (via user input at the terminal) with the number of hard liquour drinkers. Use H for the number of people who drank hard liquour, and B for the number of people who drank beer.

Sample Size	% Correct	% Incorrect
52	69	31

Problem 2:

Given the following statement:

"At the last company cocktail party, for every 6 people who drank hard liquour, there were 11 people who drank beer."

Write an equation which represents the above statement. Use H for the number of people who drank hard liquour, and B for the number of people who drank beer.

Sample Size	% Correct	% Incorrect
51	45	55

Probability of these results on the assumption that errors on each problem were equally likely is $p < .05$

Table 3

Problem 1:

Write a sentence in English that gives the same information as the following equation:

$$A = 7S$$

A is the number of assemblers in a factory.
S is the number of solderers in a factory.

Problem 2:

```
Program Kayak
Input I
K = I * 2
Print K
End
```

For the above computer program describe in English the mathematical relationship which exists between I, the number of Igloos, and K, the number of Kayaks.

Comparison of Problem 1 and Problem 2

a. Number of people who got 1 correct, but 2 incorrect	5
b. Number of people who got 2 correct, but 1 incorrect	18

Probability of these results on the assumption that case a and b were equally likely is $< .005$

Table 4

V. Why a Programming Context Decreases Reversal Errors: Some Hypotheses

The range of experiments we have carried out has provided us with compelling evidence as to the positive contribution of a programming environment to certain types of problem solving. These results are even more striking when one realizes that, a priori, one would think that writing a computer program would be more difficult than writing an equation. We have formed several hypotheses, any number of which could explain why students could solve the problems better in a programming environment:

1. Unambiguous semantics of programming language constructions. While various mathematical symbols (e.g., the equals-sign) are often open to a variety of interpretations in mathematics (see [Kaput 1979b]), programming languages require that only one interpretation be associated with each symbol. This fact is usually emphasized in programming language instruction. For example, the meaning of '=' in 'I = I + 1' is explicitly defined as an act of replacement, i.e., the value of the right side of the equation becomes the new value of the variable on the left. Also, the interpretation of variables is clear, i.e., they stand for numbers which are acted on by operators.

2. Explicitness required by the syntax of programming languages. The fact that one must write '6*S' rather than simply '6S' might serve to prompt one to view that expression operatively as meaning "six times the number of students" rather than falling into the error of viewing it descriptively as "six students".

3. Viewing an "equation" in a programming language as an active input/output transformation. That is, the right hand side of the equation (the input) is operated on to produce a value for the left hand side (the output).

4. The practice of debugging programs. While students may not be encouraged to "run their equations" in typical mathematics courses, this concept of actual number testing is an integral part of programming and programming education.

5. The practice of decomposing a problem into explicit steps. A number of students solved the computer program problem by writing down a two step sequence of operations,

   X = B/6
   B = 11*X

   One interpretation for this phemenon might be that students "saw" partial results "produced" on the way to the solution.

VI. Concluding Remarks

The empirical evidence described herein is in general agreement with that obtained by the "macro" study cited earlier. However, our research method has allowed us to develop specific hypotheses concerning factors in a programing language which contribute to improved problem solving. Currently, we are continuing to video-tape students as they solve problems, and hope to establish exactly which aspects of programming are most important to overcoming the reversal error. Our preliminary clinical results in this area point to factors 1, 2 and 3 above as the most important, but further clinical data is required to confirm this observation.

In summary, results from written tests and clinical interviews have shown that many science oriented college students have serious difficulty with the semantics of algebraic notation --- a difficulty in learning to view equations as active operations on variable quantities rather than as statements which describe a static scene. Perhaps this is not so surpising, considering the strong emphasis in secondary schools on equation manipulation in word problems. Symbol manipulation rules can theoretically be learned in school as "legal" patterns of letter movements without any semantic underpinning. Computer programming, however, puts a natural emphasis precisely on the active, procedural semantics of equations that so many students apparently lack. Thus, while our current results must be viewed as preliminary, they directly suggest that it would be beneficial to incorporate computer programming into high school algebra courses, and, we suspect, into other mathematics courses as well.

References

Clement, J., Lochhead, J., and Monk, G. (1979) "Translation Difficulties in Learning Mathematics," Technical Report, Cognitive Development Project, Department of Physics and Astronomy, University of Massachusetts, Amherst.

Clement, J., Lochhead, J. and Soloway, E. (1979) "Translating Between Symbol Systems: Isolating Common Difficulty in Solving Algebra Word Problems," COINS Technical Report 79-19, Department of Computer and Information Science, University of Massachusetts, Amherst.

Howe, J.A.M., O'Shea, T. and Plane, J. (1979) "Teaching Mathematics Through Logo Programming," DAI Research Paper 115, Department of Artificial Intelligence, University of Edinburgh.

Kaput, J. (1979a) Personal communication.

Kaput, J. (1979b) "Mathematics and Learning: Roots of Epistemological Status," <u>Cognitive Process Instruction</u> (J. Clement and J. Lochhead, Eds.), Franklin Institute Press, Philadelphia.

Monk, G.S. (1979) Personal communication.

Paige, J. and Simon, H. (1966) "Cognitive Processes in Solving Algebra Word Problems," <u>Problem Solving Research, Method and Theory</u> (B. Kleinmutz, Ed.), John Wiley and Sons, New York.

Papert, S. (1971) "Teaching Children to be Mathematicians versus Teaching about Mathematics," MIT AI Lab Memo 249, Cambridge.