

PROGRAMMING-LANGUAGES AS A CONCEPTUAL FRAMEWORK
FOR TEACHING MATHEMATICS

By W. Feurzeig, S. Papert, M. Bloom, R. Grant, C. Solomon. Taken from the Final Report on the first fifteen months of the LOGO Project, November 30, 1969. Submitted to the National Science Foundation on Contract NSF-C 558. Copies will be available from ERIC.

This is a report of research and teaching directed toward the development of a new mathematics curriculum in which presentation depends fundamentally on the use of computers and programming. The work was centered mainly on a mathematics teaching experiment with seventh grade children utilizing a programming language, LOGO, specifically designed for the teaching of mathematics. We also conducted an investigation of the use of LOGO in teaching much younger children -- a group of second and third graders.

After a brief exposition of the LOGO language, the two teaching activities are described in some detail, including many examples of the classroom and laboratory materials used. The report begins with a discussion of the reasons why the learning and teaching of mathematics are so difficult, and states the underlying issues that have dictated the kind of approach undertaken here. Following the descriptive material on the teaching experiments is a discussion of the results including some evaluations of the year's work and of the project. A detailed description of the LOGO programming language and system is appended.

The seventh grade class was taught by Mrs. Marjorie Bloom from September 1968 through December 1968, and jointly by Miss Cynthia Solomon and Dr. Seymour Papert from January 1969 through June 1969. Dr. Papert, Professor of Applied Mathematics at Massachusetts Institute of Technology, was a consultant to Bolt Beranek and Newman on this project. During the latter period, Mrs. Bloom taught the group of second and third grade children.

We did not begin the teaching with a large body of previously developed classroom materials. These had to be created concurrently with the teaching as the

courses progressed. The dynamic aspects of this day-to-day work helped assure that the content and presentation were adapted to the current needs of the children and were responsive to their difficulties, some of which we had not anticipated.

* * * *

There is an old saying among mathematicians that there is no known theorem which cannot be made transparently clear to a high school student of average intelligence in a reasonable period of time (hours or months, not lifetimes). Yet few high school students acquire an understanding of even the simplest theorems and, for most students, the formal methods of mathematics remain forever mysterious, artificial, poorly motivated, and very obscurely related to intuitive thinking.

The relation of school children to mathematics remains deeply puzzling after more than a decade of wide-scale experiment in the classroom and in the cognitive laboratory. The extent of the puzzle is often obscured by popular prejudices about mathematics and about children.

* * * *

Appropriate teaching with a suitable programming language can contribute to mathematics education in several ways.

(1) Programming facilitates the acquisition of rigorous thinking and expression. Children impose the need for precise statement on themselves through attempting to make the computer understand and perform their algorithms.

(2) Programming can be used to give students very specific insights into a number of key concepts. Ideas such as variable and function remain, to say the least, obscure for many high school students. Indeed, college students often have trouble with the many roles of the "x" in algebra: sometimes it appears to be a number, sometimes a subtly different kind of object called a variable, and on other occasions it is to be treated as a function. We contend that the difficulty stems less from the intrinsic intellectual subtlety or complexity of these distinctions than from their ethereal relation to anything in the real

and familiar world. Moreover, it is possible to fumble one's way through an algebra course without ever facing these issues squarely. In programming, the distinctions arise concretely; they must be faced; and the physical nature of the machine provides a more earthy reference than can any abstract work. These ideas should be easier in this context and our experience is that they are.

(3) Programming provides highly motivated models for all the principal heuristic concepts, for example:

It lends itself perfectly to discussion of the relation of formal procedures to intuitive understanding of problems. It provides a wealth of examples for heuristic precepts such as "formulate a plan", "separate the difficulties", "find a related problem", etc. Thus, it provides a natural context to concretize the approach to teaching associated with the name of George Polya.

It provides a sense of completely formal methods and what their purpose is. It gives the child a chance to learn to distinguish situations where complete formal rigor is necessary from those where looser thinking is appropriate.

In particular, it provides models for the contrast between the global planning of an attack on a problem and the formal detail of an elaborated solution. In the context of programming, the concept of sub-problem or sub-goal emerges crisply. It is at least highly plausible that pupils who have acquired very early the habit of organizing their approach to a mathematical problem will be better able to develop systematic habits of thought in the more murky areas of problem-solving they will have to meet later, in school and elsewhere.

The concrete form of the program and the interactive aspect of the machine allow "debugging" of errors to be identified as a definite, constructive, and plannable activity. The programming concept of a "bug" as a definite, concrete, existent entity to be hunted, caught, and tamed or killed is a valuable heuristic idea.

(4) By enlarging the scope of applications, it allows every problem to be embedded in a large population of related problems of all degrees of difficulty, for example:

Through programming, mathematical induction can be presented and generalized by its relation to recursion. An example of this kind of presentation is shown in Section 2.2. The examples given in Section 4.3 show how we have learned to present recursion itself as related to the general heuristics of planning.

The extension of an operation to a larger domain becomes an everyday activity. The newer mathematics texts do emphasize the extension of addition, for example, to successively more general kinds of numbers (integers $\rightarrow$ rationals $\rightarrow$ reals). But the phenomenon is obscured for children by its isolation and by the fact that children already know how to add real numbers.

Generalizing this, generalization becomes an activity undertaken routinely by the children.

Functions become familiar things one invents oneself to serve real purposes. We have seen children invent as many new functions in a week as they would otherwise learn (by rote!) in their whole career. More importantly, they use these functions as building blocks for constructing more complex functions which often are elements of still larger constructs -- very much in the way mathematicians use propositions to prove theorems and use these theorems to prove more complex theorems.

(5) The use of computers and programming languages is also relevant to what is perhaps the most difficult aspect of mathematics for a teacher: helping the student strive for self-consciousness and literacy about the process of solving problems. High school students can seldom say anything about how they worked towards the solution of a problem. They lack the habit of discussing such things and they lack the language necessary to do so. A programming language provides a vocabulary and a set of experiences for discussing mathematical concepts and problems. Programs are more discussable than traditional mathematical activities: one can talk about their structure, one can talk about their development, their relation to one another, and to the original problem.

(6) A related point is that the computer can be used as a mathematical laboratory to foster an experimental approach toward solving problems. Programming could, in principle, be taught as an abstract mathematical topic without using or, indeed, even mentioning computers. Presented in that spirit, the material

would retain some of the pedagogical virtues that motivate our interest in it. But an essential aspect would be lost. The use of a computer has the major merit of turning a programming language into an active instrument to control an outside reality. The most immediate effect of using a computer is that explicit and precise statement is no longer imposed by the arbitrary edict of a teacher but by the obvious necessities of making the computer do one's bidding. Since students learn to write programs by experience and experiment, it is appropriate to use the term mathematical laboratory for the practical phases of the instruction.

The reason that a laboratory is not traditionally used in mathematical study is not that it would be less valuable there than in biology, chemistry, or physics; rather, the idea of a mathematical experiment was, until recently, unrealizable, and barely conceivable, except in very special or superficial senses. How could a person set in motion a sequence of mathematical events or a mathematical process, and then see its effects unfold? Using a computer with an appropriate programming language adds this extra dimension to mathematical experience; the important contribution of the computer is a new and powerful operational universe for mathematical experiments.

(7) Finally, the richness of non-numerical examples open to programming can be exploited to enlarge the cultural base of the mathematics course by bringing it into contact with physical and biological science, language study, geography, economics, and other subjects.

Thus, our interest is not to teach programming as an auxiliary topic, but to explore means of using it as a foundation for an integrated course in mathematics. This concept of programming is distinct from the already familiar and valuable ones of teaching computer programming as a practical skill in its own right or for use in special courses in numerical applications, applied mathematics, computational methods, and the like.