How do you think academic circles will accept such new techniques?
Slowly. They didn't like Cobol either. You see when academics first started teaching about computers in universities and colleges all the mathematics students started electing to do numerical analysis and that sort of thing rather than do the mathematics courses.

This meant that the budget of the math department fell and that of the other department rose. This worried the math professors and they began to accuse the computer people of being vocational and not academic. So the computer people decided that they had to be academic so that they could justify their budgets and invented higher level studies. They weren't teaching people how to run computers. They were computer scientists. They leaned over backwards to be highly scientific and highly technical but now we are beginning to see a swing back.

Some universities have two departments — computer science which is very esoteric, and information sciences. The term information can cover many things which includes data processing but you don't say data processing in a university. A few state colleges now are teaching data processing but of course the Ivy League wouldn't even look at a piece of data (and they wouldn't know where to put a decimal point either) — they are mathematicians.
But in spite of expected hindrances you are looking forward to the future?
I just think the most exciting work is still to be done. Speed is going to be the premium.

We will have increased population, increased growth, increased needs for transportation and increased shortages which means you will have to buy exactly what you need instead of what you think you might need.

The distribution of things will have to be managed much more closely than ever before. People will have to have better information faster to make decisions faster.
You don't find threats of shortages and energy crises daunting?
When I came to England this time the one place I wanted to go to again was Stonehenge to see what men could do when they had nothing to do it with.

Those stones are big and bulky but they are all perfectly lined up to a magnificent plan. If, that long ago, men could think of such a concept and complete it with virtually no tools, and no sophisticated mechanical assistance, then we can manage pollution and oil crises and anything else if we want to.

Thomas I. M. Ho
Purdue University
West Lafayette, Indiana

REVIEW

Schwartz, J. T. On programming: an interim report on the SETL Project, Installment II: the SETL language and examples of its use. Computer Science Department, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York 10012 (October 1973).

Mullish, M. and Goldstein, M. A SETLB primer. Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York (1973).

The current work (installment II) is the second in a series of three reports on SETL, a new programming language derived from concepts of the mathematical theory of sets. Installment I, dated February 1973, outlines the general approach of the SETL effort. Installment III will be titled "Extension and Optimization". Installment II describes the SETL language facilities, presents a variety of algorithms illustrating the use of SETL, and provides details of the run-time routines implementing the SETL primitive operators. SETLB is a subset of SETL currently implemented on the CDC 6600 at the Courant Institute.

The variety of SETL-coded examples seems to substantiage the claims by SETL advocates of the language's capability to specify complex algorithms. Being a language of very high level, SETL permits the manipulation of complex data structures without requiring specification of the physical structures ordinarily expected of the user of lower level languages.

The value of these advantages must be weighed against the generally inefficient nature of SETL programs. In particular, the use of certain powerful SETL primitive functions that are readily available makes it all too easy to generate very inefficient programs. For example, the SETL operation pow(a) which forms the set of all subsets of the set a can cause very inefficient algorithms. A less important criticism relates to the awkwardness of the SETLB character set required by the adaptation of the familiar symbols of set notation due to the unavailability of those symbols on the keypunch and the computer.

SETL should be of interest to advocates of structured programming. Dijkstra [1] advocates the postponement of decisions regarding particular data representations in stepwise program composition. SETL enables the postponement of the selection of physical structures for an efficient implementation of an algorithm until after the algorithm has been first abstractly specified in SETL.

Also, practitioners of modular programming should find SETL of interest. Parnas [2] describes a technique to provide program module specifications sufficiently precise and complete so that other modules can be written to interact with the specified module without additional information. Schwartz [3] proposes that SETL might also serve well as a module specification language.

Further, proponents of proofs of program correctness may find that the mathematical foundations of SETL make the language especially suitable for the coding of algorithms whose correctness must be rigorously determined.

Finally, teachers of computer science may want to examine SETL's pedagogic potential in the study of abstract algorithmic processes and of concrete data structures.

Of course, students of programming and of programming languages should find this work interesting for its contribution to mathematical formalization of programming principles and language semantics.

References

1. Dijkstra, E. W. Notes on structured programming. T.H.-Report 70-WSK-03, Department of Mathematics, Technological University, Eindhoven, The Netherlands (April 1970).
2. Parnas, D. L. A technique for software module specification with examples. Comm. ACM 15, 5 (May 1972), 330-336.
3. Schwartz, J. T. Principles of specification language design with some observations concerning the utility of specification languages. in Rustin, R., ed., Algorithm Specification, Englewood Cliffs, NJ: Prentice-Hall (1972), 1-37.