array [1:2], and the list operations carried out on these arrays. On most machines this approach would increase the storage requirements of a program by a factor of two over assembly language implementations.

As the authors point out, the primary purpose of their scheme is the suppression of detail. Some of the burden of coding is shifted to the machine, allowing more efficient utilization of programmers. So long as the operators and data structures proposed do not go beyond the capabilities of the "ALGOL machine," this scheme is reasonable. The paper's first detailed example is a set of definitions for matrix manipulation, a task for which the "ALGOL machine" is quite well suited. The last example, list processing, is unnatural for the "ALGOL machine," and hence this implementation suffers.

In summary, the method proposed in this paper would find its primary application in the creation of a library of abbreviations useful to users of ALGOL. The casual user would probably be unaware of the details of these abbreviations. Machine code methods are rejected in order to ensure full compatibility with all ALGOL implementations. This proposal does not allow for efficient extension of ALGOL. Because definitions are restricted to the operators and data types of ALGOL, it is not possible to produce programs which are structurally different from normal ALGOL programs although they may appear considerably more compact.

The paper itself is apparently intended as a complete description of the proposed system. It is in no way a simple introduction to the topic, and only the first section is accessible to most ALGOL users. A tremendous amount of information is packed into its 19 pages, perhaps enough to implement the scheme. The text bristles with syntax and is quite short on explanations and examples. Those which can be found demonstrate the scope of the processor but are not calculated to appear transparent to anyone but the expert. The title bills this as "a proposal," but little attention is given to convincing arguments that this approach is useful. Only those people who are engaged in the implementation of a similar facility will find the courage to press on to complete mastery of this paper.

William M. Waite
University of Colorado
Boulder, Colo. 80302

R68-25 An On-Line Editor—L. P. Deutsch and B. W. Lampson

With the growing abundance of time-sharing systems we begin to hear more and more about text editors. These programs were originally designed to satisfy the common need for creating and modifying source language programs for the various programming language processors found in a time-sharing system. Soon it became obvious that editors could be used in many other situations, for example, in data preparation, report generation, manuscript editing, and in some cases even as the programming language itself. This paper describes a text editor, QED, which has been in use in the Berkeley time-sharing system for the SDS-930 and which has benefited from two and a half years of user experience.

QED is a line-oriented editor designed to be used from a type-writer or Teletype over full duplex lines. Its interesting features include a simple and mnemonic command language, content addressing, a line-edit mode, and temporary storage for frequently used strings of either text or commands.

The QED command language is quite simple with most commands preceded by either one or two addresses (from, to) and terminated by a `;'. The commands INSERT (before), APPEND (after), PRINT, DELETE, and CHANGE (delete and insert) provide the casual user with a basic editing capability. Additional commands permit reading from and writing to files on mass storage, software tab settings, and string substitution. Since the system in which QED resides permits full duplex operation most commands can be abbreviated by their first character followed by a `;'. and QED provides the remaining letters of the command on the user's terminal. When text is to be input, as in the INSERT or APPEND commands, control characters are used to delete the last character, word, or line and to terminate input.

Addresses in QED are of three types: line numbers which refer to the position of the text line in the text buffer, label addressing which causes a search for a string of characters occurring at the beginning of a line, and a more general content addressing which searches for the string anywhere in the line. Any of these addresses may be followed by either positive or negative displacements. The authors motivate the use of content addressing by comparing it with user- or system-generated sequence numbers. In my opinion some motivation is necessary for the use of line numbers (instead of sequence numbers) for they change with almost every editing operation and therefore have very little association with the text.

The line-edit mode of QED permits the user to modify a single record on a character-by-character basis. The user describes, through the use of control characters, those characters which are to be copied, deleted, replaced, or inserted in the construction of the new record. The power of the line edit is shown in several examples given in the paper. The authors have a warning, however, to those who consider implementing such a feature. "This power has been purchased at the cost of considerable complexity; although the basic idea of the line edit is simple, there is a profusion of commands to speed up the handling of special cases which is somewhat bewildering to the new user." (Fig. 4 lists 18 single-character line-edit commands.)

The concept of string buffers or temporary storage is shown to be an extremely useful tool. The buffers can be loaded from either the text or the terminal and when they are referenced in command mode QED behaves as though the characters in the buffer were being typed in from the user's terminal. This permits the user to refer to a frequently used text phrase by its buffer number; to easily reorganize sections of his text; to call on command procedures; and to undo certain editing operations since deleted text is placed into a buffer by QED. This would also seem to be a simple way to bootstrap other commands into the system although this is not mentioned specifically in the paper. The authors considered enhancing this programming capability by adding a conditional facility and pattern matching but they decided to leave programming to SNOBOL or other string-processing languages. However, since most time-sharing systems lack a conversational SNOBOL, these additions might be welcomed.

As a user of time-sharing systems I was impressed with the details of QED which were aimed at user convenience. (A single-character line feed causes the next line to be printed; the string delimiter in the substitute command is dynamically specified by the user.) As a designer of time-sharing systems I would have been interested in learning more of the system environment including file structure, number of users, expected response time for completion of abbreviations, and line editing. I was particularly impressed with the uses made of the full duplex capability of the system. As a reviewer I was impressed with the clear and well organized presentation.

H. A. Freedman
RCA Laboratories
Princeton, N. J.

D. ANALOG COMPUTATION


Howe has included a variety of valuable information in this paper, the unifying thread being the improvement of precision in analog devices, a topic in which he has long been a leader and innovator. This paper describes 1) precise means of implementing with hardware nonlinear mathematical models involving function generation, multiplication, and division, and 2) techniques of scaling the mathematical