−i'). If this syntax has to be defined as well as used, then the analytic description begins to incorporate the skeletal examples of the other method.

Modification (*b*) ('**else**' and '**end**') requires new terms to plug into the editing part of the description. These two words find themselves at the end of the line because they do not start new structures, and the new line rule specifies only

'Each structure starts on a new line'.

If it specified

'Each structure ends on its own line',

then '**else**' and '**end**' would start their lines, but so would ';', and if it specified

'Each structure starts and ends on its own line',

then '**else**', '**end**', and ';' would all go on lines by themselves. Therefore no general rule of this kind will put ';' at the end and '**else**' and '**end**' at the beginning (and yet another rule is needed to align '**end**' with its '**begin**'). When all these cases are accounted for, the number of rules approaches the number of structures, and 'skeletal examples' re-appear.

The conclusion is that there is more variability in each format described than there are terms for describing them. Is there then any merit to hankering for an analytic description of each format if it means establishing new terms for structure and editing? The argument for merit is as follows.

Consider the whole range of textual material that humans process: natural language text divided into chapters, sections, paragraphs; flowcharts and organisational charts; high-level source text; indices, summaries, and tables of contents; interactive text for programmed learning; . . . Consider listing the major structures underlying them (trees, lists, conditional expressions, go-to's, . . .), and the major editing conventions used to express them (subheads, section numbers, parentheses, indentation, . . .). It may be that both lists are quite short in relation to the amount of material to which they apply (even though long in relation to each particular application).

In that case the search for an adequate vocabulary for analytic description of text formats has paid off with a contribution to the handling of human-readable text in general. This may be a significant pay-off if the major problem in the current human-machine relationship is communicating with the humans.

Acknowledgement

Thanks are due to Karen Mitchell, David Parnas, and Aimé Sanders for aid and encouragement, and to the Computer Services Network of Philco-Ford for computing facilities.

References

CONROW, K., and SMITH, R. G. (1970). NEATER2: a PL/I source statement reformatter, *CACM*, Vol. 13, No. 11, pp. 669-675.

MCKEEMAN, W. M. (1965). Algorithm 268: ALGOL 60 reference language editor, *CACM*, Vol. 8, No. 11, pp. 667-668.

NAUR, P. (ed.) (1965). Revised report on the algorithmic language ALGOL 60, *CACM*, Vol. 6, No. 1, pp. 1-17.

SCOWEN, R. S., ALLIN, D., HILLMAN, A. L., and SHIMELL, M. (1971). SOAP—a program which documents and edits ALGOL 60 programs, *The Computer Journal*, Vol. 14, No. 2, pp. 133-135.

Book review

Simula Begin, by G. M. Birtwistle, O.-J. Dahl, B. Myhrhaug, K. Nygaard, 1973 (*Auerbach Publishers Inc.*, $19.95)

Simula is an extension of ALGOL 60 to provide a more powerful programming language and a simulation language.

The first chapter (72 pages) is an overall introduction to the main concepts of programming in general and Simula in particular.

The second chapter (30 pages) is on blocks and the third (20 pages) is on procedures. The fourth chapter (40 pages) introduces the concept of class which is a new and difficult concept, it includes facilities for defining various types of data structures, including queues, as well as various levels of program structures, including concepts such as prefix blocks and remote accessing of attributes of various entities. Chapter 5 (10 pages) is on recursion and chapter 6 (35 pages) on powerful and extensive facilities for text handling and input/output. Chapter 7 (45 pages) illustrates the power of the facilities introduced so far by applying them to various non-numeric applications such as games, tree structures, queues and symbolic differentiation. The last two chapters (60 pages) introduce simulation facilities of the language such as processors and their scheduling (activate, passivate, terminate, etc.)

The book contains a large number of programs, examples, solutions and appendices.

The production of ALGOL-type programs in print is always difficult, and this book is difficult to read because programs and diagrams were reproduced by photocopying, often from some sort of computer terminal printout.

Simula is a powerful general language, but as a simulation language it lacks several facilities, e.g. for collection of simulation and queues statistics. This is the only book on the language; up to now one had to look up various papers and manuals. The language is powerful enough to provide the advanced programming and data structures concepts needed for the additional facilities, but simulation languages are expected to provide them as standard facilities.

To sum up, the language, like the book, is powerful but difficult to use.

I. M. KHABAZA (London)