the effect of future applications of the **INP** production. It is possible that text to be processed after an application of one of these productions may not necessarily start with the symbol **INP**, although this will generally be the case.

References

Brown, P. J. (1967). The ML/1 Macroprocessor, *CACM*, Vol. 10, pp. 618-623.

Brown, P. J. (1971). A Survey of Macroprocessors, *Ann. Rev. Automatic Programming*, Vol. 6, pp. 37-88, Pergamon Press, Oxford.

Mooers, C. N. (1966). TRAC, Procedure-describing Language for the Reactive Type-writer, *CACM*, Vol. 9, pp. 215-219.

Nudds, D. (1974). The Simulation of a Digital Computer and its Languages on another Computer, Ph.D. Thesis, University of Bradford.

Strachey, C. (1965). A General-Purpose Macrogenerator, *The Computer Journal*, Vol. 8, pp. 225-241.

Waite, W. M. (1967). A Language Independent Macro Processor, *CACM*, Vol. 10, pp. 433-440.

Waite, W. M. (1970). The Mobile Programming Programming System: STAGE 2, *CACM*, Vol. 12, pp. 507-510.

Book reviews

Formal Languages and Programming, edited by R. Aguilar, 1976; 129 pages. (*North-Holland*, US$15.00)

One might think to use formal language theory to describe exponential growth in colonies of cells or simple animals; symbols are attached to strings one at a time by production systems, whereas all the animals reproduce themselves simultaneously, however. What is needed instead is a variant theory called Lindenmayer systems. Readers who wish to explore this tempting byway in formal language theory should certainly join the tour led by Arto Salomaa at this symposium, for he has a delightful way of describing recent theoretical work to newcomers. Thus we learn about the animals in IL systems who discuss their reproduction problems together, and those in OL systems who do not. It seems some animals are more context-free than others.

Recent work in transporting compilers across 'families' of computers, in discovering equivalent grammars from which faster compilers might be written, and in describing 'extensible' languages (i.e. families of languages) such as ALGOL 68, appears to entail the study of 'families of grammars'. These are sets of related grammars whose common features can be abstracted by grammar-generating functions called 'grammar forms'. The prophet Seymour Ginsburg came down from the mountain for just long enough to present to the symposium the tablets on which he had briefly though clearly inscribed the main results obtained by his colleagues in this new work.

Doubts are being voiced increasingly about the way in which compound data types have to be declared in the current crop of very high level languages. It is being suggested that what we ought to be offered instead is provision to declare sets or 'clusters' of functions to access the data. The data themselves should remain hidden behind the functions. Melkanoff explains how this proposal might serve to relate linguistic mechanisms for data type declaration to hardware mechanism for memory protection, and how formal language theory and proofs of program correctness might then be used to design both mechanisms more neatly.

Many of us are still arguing over the best way for pairs of co-routines to share variables in order to pass information 'horizontally' between shared storage areas. Adin Falkoff explains how files are shared in time-sharing-APL to the satisfaction of its users apparently, although his solution seems to me far less general than the better-known ones of Dijkstra and Per Brinch Hansen.

The paper by Kupka is flatulent and the one by Indermark is marred by incomplete definition of terms. Intending readers of the three minor papers should bear in mind that their French authors work in an algebraic tradition that refers to contextfree and regular languages as 'algebraic' and 'rational' languages.

R. Edwards (Egham)

Revised report on the algorithmic language ALGOL 68, edited by A. van Wijngaarden et al, 1976; 236 pages. (*Springer-Verlag*, DM 24) or (*Stichting MC Tract*, 50, Dfl. 25)

This report is the final definition of the programming language ALGOL 68. It is a substantial revision of the original report (Mathematisch Centrum Amsterdam, MR 101 February 1969) both in style and content, and was first published as a supplement to *ALGOL Bulletin* 36 in March 1974. The draft was subjected to meticulous scrutiny, and numerous corrections were made before the first publication of the revised report in *Acta Informatica* Volume 5, parts 1, 2 and 3 (December 1975).

The two volumes which are the subject of this review are identical reprints of the *Acta Informatica* text. For the first time, the report has been able to be printed in several typefaces, which as readers of the draft will confirm adds greatly to its readability. In addition, the review of the draft by so many people seems to have been worthwhile; by August 1976 only one minor error was known to the editor of *ALGOL Bulletin*.

The revised report is a very specialised document, for an audience of specialists. It is a formal and rigorous definition of a complex language, and as such of interest to limited classes of people. Theoretical linguists and some computer scientists may find it interesting. University lecturers with the time and inclination to study the metalanguage may find it reassuring to have on their bookshelf, as the final authority for settling arguments. Compiler writers will find it essential reading, although arduous. It is most definitely *not* suitable as an introduction to ALGOL 68 for programmers—even experienced programmers—since the formal descriptive technique is not easy to follow, even with practice.

Anyone who wishes to learn ALGOL 68 (which I would strongly recommend) should read one of the introductory texts which are increasingly available. Anyone who needs the report, particularly anyone who is still working from the draft version, should hasten to order one of these reprints. The choice between them would seem to be a matter of convenience, or the current state of the currency markets.