specification and the resulting action or imposing a severe block upon the type of problem that can be solved.

Szanser (1972) states that the Machine Translation project of the National Physical Laboratory constantly ran up against the problem of coping with errors in the input, 'even if the error was trivial by human standards (such as to pass unnoticed —obvious evidence of the existence of 'automatic error correction' in the brain)'.

Again we would emphasise that our position is not to let programmers 'run riot' with the system. A well designed tolerant system will naturally provide incentives to encourage precise programming in that 'correct', that is expected program statements, will be processed faster. It will exploit the redundancy in program specification to increase the chances of recovering from inaccuracy while it will not penalise the correct programs.

References

BLOOM, B. H. (1970). Space/time trade-offs in hash coding with allowable errors, *CACM*, Vol. 13, No. 7, pp. 422-426.

CONWAY, R. W., and WILCOX, T. R. (1971). *Design and implementation of a diagnostic compiler for PL/I*, Research Report 71-107, Department of Computer Science, Cornell University, September 1971.

DAMERAU, F. J. (1964). A technique for computer detection and correction of spelling errors, *CACM*, Vol. 7, No. 3, pp. 171-176.

IRONS, E. T. (1963). An error-correcting parse algorithm. *CACM*, Vol. 6, No. 11, pp. 669-673.

JAMES, E. B., and PARTRIDGE, D. P. (1973). Adaptive correction of program statements, *CACM*, Vol. 16, No. 1, pp. 27-37.

LAFRANCE, J. (1971). *Syntax-directed error recovery for compilers*, Ph.D. thesis, University of Illinois at Urbana-Champaign.

MORGAN, H. L. (1970). Spelling Correction in systems programs, *CACM*, Vol. 13, No. 2, pp. 90-94.

PARTRIDGE, D. P. (1972). *Heuristic methods in the analysis of program statements*, Ph.D. thesis, Department of Computing and Control, Imperial College, University of London.

SZANSER, A. J. (1972). *Automatic error correction in natural texts*, National Physical Laboratory, Part II COM 52 and Supplement COM 63.

Book reviews

FORTRAN to PL/I Dictionary, PL/I to FORTRAN Dictionary by Gary De Ward Brown, 1975. (*John Wiley & Sons, New York*, £5·10).

This book explains FORTRAN and PL/I in terms of each other. The first section consists of alphabetic lists in the style of English-French and French-English dictionaries but the major part of the book is taken with explaining the concepts of the languages from basic statements through data storage and control statements to input/output and debugging aids with where there are differences, as there usually are, FORTRAN to the left and PL/I to the right of the page. There is a chapter on features which have no parallel in FORTRAN, such as multitasking and recursion, and there are appendices on PL/I character sets and abbreviations, interlanguage communication on the IBM 360/370 and answers to the exercises which terminate each chapter. The chapters are ordered as for a reference, not an initial teaching work. There is a good index.

It is assumed that the reader is familiar with the basic concepts of programming, punched cards, and so on and it is visualised that the main use of the book would be to teach one language to a person who knows the other, or to act as a reference for such a person. It is also suggested that the book could be used as a text for a course in which languages are learned in sequence or as a text for a course in comparative languages, or as a language reference.

Of these multifarious targets the one missed by the greatest distance is the function of the single language reference. The FORTRAN covered is described as all of ANS plus 'most of the widely-used non-ANS FORTRAN IV features' including 'those of WATFOR and WATFIV'. What is actually covered is all of ANS plus most of the IBM 360/370 FORTRAN extensions plus a few references to language fragments from other systems, particularly CDC 6000. The earlier chapters are concerned mainly with IBM facilities but later there is more acknowledgement of other systems. The reader, inevitably, must look elsewhere to decide if a particular statement is acceptable to a particular language processor. Similarly the PL/I is unequivocally stated to be IBM version 5 level F with additional features of the IBM checkout and optimising compilers but by Chapter 7 there is an exception indicating a difference between IBM 360/370 and the rest. The book therefore attempts to reference a phantasm.

Further, one must demand accuracy of a reference work and it is not acceptable for example in FORTRAN to use a variable name for both a real and a logical variable in the same statement (p. 48) or to describe the skip/format code as Xw with examples X6 and X10 (p. 127). These are possibly due to printing errors; the general standard of presentation is high, especially given the split-page format, though there are occasional lapses such as using the wrong type face for a word and there is the common error of sometimes continuing a program statement across lines as if it were a natural language sentence.

These minor infelicities will not trouble the experienced programmer who will probably not even notice that page 1 contains an elementary error, a near-illegible example and a spelling mistake. In its basic function of setting out the two languages side by side and line by line with abundant examples the book is excellent and one can imagine it being of great utility to the programmer who is familiar with one of the languages and is rusty at the other and who is prepared to crosscheck with the manual of the processor he is using. *A fortiori*, for comparative language study the book can be used very easily to give an overview of the facilities offered by the two languages.

Those who have read thus far will have gathered that the third of the five aims of the book, to teach the languages in sequence, is not really fulfilled as there are better ways for those who must learn FORTRAN and PL/I from scratch. The author uses the analogy of an English-French dictionary: to give this book to an inexperienced programmer would be to invite translations of the level of 'lettre Française'.

D. T. MUXWORTHY (Edinburgh)

Elementary Algol by A. Brundritt, 1976; 80 pages. (*Macdonald & Evans*, £1·25).

This book may be ideally suited to students on Alan Brundritt's ALGOL courses. I would not recommend it to anyone else. The order in which the language is taught is completely at variance with modern ideas of programming: **goto** is taught early and used heavily throughout the book; and although the safer control statements are introduced, they are badly illustrated (**for** $j := 1, j + 1$ **while** $j > 0$ **do** . . . . . with a **goto** to leave the loop, in one example!). **Comment** is included as an afterthought in the last chapter—it might have helped to explain examples like the one above.

Both input/output and Job Control Language are described without mention of their ICL 1900 dependencies, an omission which is made more grave by the promise in the Introduction that such dependencies would be 'made clear'.

The style is uncomfortably chatty—acceptable in a lecture, but irritating and obtrusive in print.

'There may not be many cheaper books than this, but there are better ones'. Day (1976).

A. C. Day, reviewing Computer Programming/FORTRAN in *Computer Bulletin*, series 2 no. 7 March 1976, page 40.

ANNE ROGERS (Bath)