6. *VDM Specification Language – Proto-Standard.* Technical Report, British Standards Institution, March 1991. BSI IST/5/19.
7. J. Dawes, *The VDM-SL Reference Guide.* Pitman (1991).
8. I. Hayes (ed.), *Specification Case Studies.* Prentice-Hall International, Englewood Cliffs, New Jersey (1987).
9. I. J. Hayes and C. B. Jones, Specifications are not (necessarily) executable. *Software Engineering Journal*, pp. 330–338 (November 1989).
10. *Information Processing – Text and Office Systems – Office Document Architecture (ODA) and Interchange Format.* Draft International Standard ISO/DIS 8613/1–6. Volume parts 1–6, ISO (1988).
11. *Information Processing Systems – Open Systems Interconnection – LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.* ISO8807. ISO (1989).
12. *Information Processing Systems – Open Systems Interconnection – Estelle – A Formal Description Technique Based on an Extended State Transition Model.* ISO9074. ISO (1987).
13. C. B. Jones, *Systematic Software Development Using VDM* (second edition). Prentice-Hall International, Englewood Cliffs, New Jersey (1990).
14. P. G. Larsen, *The Dynamic Semantics of the BSI/VDM Specification Language.* Technical Report, the Institute of Applied Computer Science (February 1992).
15. N. Plat and W. J. Toetenel, *Tool Support for VDM.* Technical Report 89–81, Delft University of Technology (November 1989).
16. N. Plat and P. G. Larsen, An overview of the ISO/VDM-SL standard. *Sigplan Notices* **28** (8) (1992).
17. N. Plat and W. J. Toetenel, *A Formal Transformation from the BSI/VDM-SL Concrete Syntax to the Core Abstract Syntax.* Technical Report 92–07, Delft University (March 1992).
18. C. L. N. Ruggles (ed.), *Formal Methods in Standards: A Report from the BCS Working Group.* Springer-Verlag, Heidelberg (1990).
19. *Recommendation Z.100. CCITT Specification and Description Language SDL.* CCITT (1988).
20. M. Spivey, *The Z Notation – A Reference Manual* (second edition). Prentice-Hall International, Englewood Cliffs, New Jersey (1992).
21. S. Prehn and W. J. Toetenel (eds), *VDM '91: Formal Software Development Methods.* Lecture Notes in Computer Science, vols 551 and 552. Springer-Verlag, Heidelberg (1991).

Book Reviews

PAUL COCKSHOTT, *A Compiler Writer's Toolbox*, Ellis Horwood, Chichester, 1990. £18.95. ISBN 0-13-173782-1

There are two sorts of compiler courses: the one presents and compares alternative techniques for each phase of compilation; the other takes a single technique from each phase and deals with it in detail. The comparative course aims to give the student the knowledge needed to choose the correct set of techniques for a given language and application, and of necessity will involve small examples and exercises to illustrate each point. The single-model course is geared more towards the student acquiring the skills and experience needed to implement a complete compiler, special cases and all. One could say that the first course gives the science, and the second the engineering, and that both are necessary and complementary. With compiler writing-taking being edged out of the curriculum these days, it is unlikely that there will be room for two. The choice of which course to go for is therefore important.

Paul Cockshott's book is firmly in the second camp. It presents an integrated set of techniques for compiling languages in the Algol/Pascal tradition into abstract machine code which is then assembled on a PC, the whole system using Turbo Pascal and a special Toolbox written by the author. One can scarcely get more specific, and so one wonders what the value is of sharing these tools and techniques outside the university department where they were developed.

The value is that the book treats compiler writing **seriously**, addressing the issues which make or break a production compiler, but which are usually ignored in textbook examples. Specifically, the book considers object-oriented heaps, raster graphics, the integrated editing environment, persistent data, classes, dynamic linking and modifying memory allocation, in addition to the usual data types, control structures and procedures of a block-structured language.

The vehicle for discussion is the language Persistent S-algol, which is introduced in Chapter 2. As its name suggests, PS-Algol supports dynamic data declaration, implicit storage management, recursion and powerful input–output facilities. As such, it provides a greater challenge for the compiler writer than Pascal.

Chapter 3 looks at overall compiler strategy, and describes how the compiler will be built up from a one-pass PS-Algol to abstract machine translator followed by an assembler, and stresses how the extra step aids portability. Chapter 4 gives a brief (but modern) overview of the theory behind grammars, leading into Chapter 5, which deals with lexical analysis via regular grammars and a finite state machine. Here we encounter for the first time the Compiler Writer's Toolbox, with different programs and units being brought into play for efficient state-driven lexical analysis. The lexemes produced as a result are then classified and stored in a symbol table as discussed in Chapter 6.

Syntax analysis begins in Chapter 7, and the method employed is recursive descent based on BNF. Analysis of the basic control structures and expressions is covered, before Chapter 8 returns to the symbol table. The Toolbox once again provides the framework for the more sophisticated types, class hierarchies and graphics in PS-Algol. The author discusses how some of these features (for example, first-class procedures) had to be eliminated from current versions of the language in order that the compiler could run on a small PC. On the other hand, colour screens were deemed sufficiently ubiquitous to cause the raster graphics to be retained.

Chapter 9 introduces the S-Algol abstract machine and goes through all the steps necessary to generate code for S-Algol's rich data areas: stack, volatile heap and persistent heap. Chapter 10 follows this up with a discussion of the assembler, which is in itself a very useful stand-alone section. Chapter 11 goes into more detail on heap management in a dynamic declaration environment (unlike Pascal's), and Chapter 12 discusses the integrated editor, which provides for interactive compiling and good error reporting. Finally, Chapter 13 looks at linking a program for execution and loading it into memory, once again using the Toolbox to good effect. Some of the issues discussed here are very real – interfacing to the DOS exec unit, use of COMMAND.COM and getting back exec error codes – and are usually simplified out of compiler courses. Their inclusion adds greatly to the value of the book.

The comprehensive set of appendices rounds off the book, and includes details of how to acquire the software from the author. There are several minor errors in the diagrams and text, but overall the book is well written and presented. Only the early chapters have exercises, and there are no answers. Some of the exercises relate to using the Toolbox to write a Pascal compiler. Since S-Algol, which is the example in the book, is in a sense a superset of Pascal, it would probably be better for students to tackle a more advanced language, for example Ada or C++.

I would certainly recommend this book for compiler engineering courses. The strangeness of S-Algol is adequately compensated by the opportunity it gives to examine more difficult compilation issues. In addition, the Toolbox together with the book could be a useful adjunct to a software practitioner's library.

J. BISHOP
Pretoria, South Africa