Bibliography

This short list of books and papers is not intended to be exhaustive, but to point in the direction of further reading about occam, parallel programming, and some of the subjects touched upon in the course of this book.

About occam 2

The authoritative reference on the occam 2 language is

1. INMOS Limited (1988) occam 2 *Reference Manual*, Prentice-Hall International, Hemel Hempstead,

and it is the document against which we measured our material for this book for its accuracy. In addition to the language definition, the *Reference Manual* contains small illustrative examples of each feature of the language, and details of the functions and procedures to be provided by standard libraries in an occam programming system.

2. Dick Pountain, David May (1987) *A Tutorial Introduction to* occam *Programming*, Blackwell Scientific Publications, Oxford.

A very gentle introduction to parallel programming is oddly bound in this book with an extremely terse (only twenty-two page) language definition by the designer of occam 2.

There are a couple of small historical discrepancies between the language described in this book and the *Reference Manual*, apparently to be removed in a forthcoming edition. The same sorts of slight variations from the reference language are also present in

3. Alan Burns (1988) *Programming in* occam 2, Addison-Wesley Publishing Company, Wokingham,

which is a broader survey. As well as the kind of introduction to occam 2 in the first few chapters of the present book, Burns touches on the implementation of occam on transputers, and relates it to other languages.

About occam

The revolutionary program structures possible in the present occam language were first introduced in a simpler language, sometimes called 'proto-occam', or occam 1. Indeed the principal differences between that and occam 2 are to do with data typing, and other matters which are just as important in sequential programs.

The original reference on occam is

4. INMOS Limited (1984) occam *Programming Manual*, Prentice-Hall International, Hemel Hempstead,

which bears a family resemblance to the language summary in [2].

A more discursive approach will be found in

5. Geraint Jones (1987) *Programming in* occam, Prentice-Hall International, Hemel Hempstead,

which was the prototype for the present book. It contains treatments in proto-occam of most of the same examples as the present book, together with a summary of the language.

Another collection of examples of occam programs will be found in

6. Jon Kerridge (1987) occam *programming: a practical approach*, Blackwell Scientific Publications, Oxford.

About reasoning about occam

Parallel programming is potentially much more complicated than sequential programming, yet occam is more tractable than many a sequential language, and occam programs can be more elegant than programs in other languages. Much of this is attributable to a sound mathematical foundation.

The outline of the occam language is very close to the surface in

7. C.A.R. Hoare (1978) Communicating sequential processes, *Commun ACM*, **21**(8) 666–677,

which is now one of the classic papers about writing and reasoning about parallel programs.

The theory of CSP has developed a great deal in the intervening decade, and a programmer who seeks a deeper insight into the theory of concurrency and synchronization should certainly look at

8. C.A.R. Hoare (1985) *Communicating Sequential Processes*, Prentice-Hall International, Hemel Hempstead,

which sets out the theory of concurrency as a course suitable for mathematically competent programmers, as all programmers should surely be.

A mathematician taking an interest in programming might be happier to start with a different, more abstract and algebraic approach of the calculus of communicating systems (CCS)

9. Robin Milner (1989) Operational and algebraic semantics of concurrent processes, in *Handbook of Theoretical Computer Science* (ed Jan van Leeuwen), North Holland, Amsterdam (to appear),

which yet seems on first sight to be much closer to the reality of occam than is that of CSP.

For our purpose here, the principal virtue of these underlying theories is that they give us a way of understanding and reasoning about the programs which we want to write. As

10. A.W. Roscoe, C.A.R. Hoare (1986) *The Laws of* occam *Programming*, Programming Research Group technical monograph PRG-53, Oxford,

notes, one of the beauties of occam is the large number of memorable algebraic laws which relate programs written in it. This paper sets out a semantics of occam which is based on these laws, and has been the basis for work on mechanical manipulation of occam programs.

One of the characteristic signs of the insecure parallel programmer is a morbid fear of deadlock, and a shining success of the theory is that it has started to tame this dragon. A short but important paper about analysing networks of processes

11. A.W. Roscoe, Naiem Dathi (1987) The pursuit of deadlock freedom, *Information and Computation*, **75**(3) 289–327,

presents some techniques for showing that networks do not become deadlocked, in essentially the same sorts of ways that one would show that a sequential program terminated.

About the transputer

Although we have been concerned with parallel programming, rather than with the programming of parallel computers, we should acknowledge that a driving force behind the development of occam has been the transputer and the availability of parallel computers built of transputers.

The engineer's reference manual

12. INMOS Limited (1988) *Transputer Reference Manual*, Prentice-Hall International, Hemel Hempstead,

explains the architecture, which supports the duality of many processes executed by one processor and many processors executing one program. It also describes the INMOS link mechanism which implements synchronization between processes executing on distinct transputers.

About floating-point arithmetic

One of the substantial differences between occam 2 and earlier dialects of occam is the integration of approximate real arithmetic into the language. It was a design commitment that the arithmetic operations should conform to the accepted standard for floating-point arithmetic, which is

13. *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std 754-1985, New York, August 1985.

This is a lengthy document which attempts, by a combination of formality and natural language, to specify precisely the performance to be expected from an implementation of floating-point arithmetic.

Our understanding of the Standard is based on a formal presentation of its requirements,

14. Geoff Barrett (1987) *Formal Methods Applied to a Floating Point Number System*, Programming Research Group technical monograph PRG-58, Oxford,

which sets out formally the arithmetic operations, and shows the first steps in a rigorous development of implementations from these specifications. This work was taken further within INMOS, to transform the occam programs into the microcode which controls the implementation of floating-point arithmetic in the IMST800 transputer.

Some of the design decisions represented in Chapter 10 of the present book are documented in

15. Jerome T. Coonen (1984) Contributions to a proposed standard for binary floating-point arithmetic, *Ph.D. Thesis*, University of California, Berkeley,

which has a chapter on conversion between binary and decimal representations of floating-point numbers.

About Life

The game of Life became widely known through a number of articles in Martin Gardner's column in the Scientific American. A good general introduction is to be found in, in particular,

16. M. Gardner (1970) Mathematical games, *Scientific American*, October 1970, 120–123,

and

17. M. Gardner (1971) Mathematical games, *Scientific American*, May 1971, 112–117,

but the definitive collection of pretty colonies is to be found in pages 817–850 of the second volume of the two volume

18. E.R. Berlekamp, J.H. Conway, R.K. Guy (1982) *Winning Ways for your Mathematical Plays*, Academic Press, London.

This is a mathematically taxing, but entirely beautiful work on the theory of games. Amongst other things, the section on Life demonstrates that every sufficiently well-stated mathematical problem can be reduced to a question about Life. It outlines the construction of a universal machine as a Life colony which performs computations with streams of gliders.

About Huffman coding

Huffman coding lies deep at the foundations of information theory. The original paper which described the idea

19. David A. Huffman (1952) A method for the construction of minimum-redundancy codes, *Proc I.R.E.*, **40**(9), 1098-1101,

is interesting both because of its status as a classic, and because it shows how much easier it has become to describe algorithms after thirty years of progress in computing.

We take our algorithm for adaptive coding, and some insights into Huffman's original algorithm, from

20. Robert G. Gallager (1978) Variations on a theme by Huffman, *IEEE Trans Information Theory*, IT-**24**(6), 668-674,

which was only prevented by publishing delays from being a twenty-fifth anniversary celebration of Huffman's work.