Preface

This is a book about certain sorts of parallel program. It is certainly not about parallel computers, nor is it really about the *occam* parallel programming language. Hardly any of it is about writing programs to run faster by being executed in parallel.

Long before programmers had made any significant progress with the complexities of programming single processor sequential computers, they knew that the job was only made more complex by trying to get the computer to do two things at once. The parts of an operating system responsible for keeping as many as possible of the peripheral devices in concurrent activity are always the hardest to understand, and the trickiest to get right.

The problems needing solution are rapidly becoming more numerous and difficult. It used to be that the management of concurrency was undertaken on the large scale: creation of new processes, and synchronization and scheduling of their actions were necessarily expensive in comparison to the real work done. These are conditions that seem to persist in the construction of operating systems and of continent-wide communication networks, where processes are consequently long-lived and interact as infrequently as can be contrived. Such are not the conditions experienced by programmers of multi-processor computers, where it becomes feasible to create large numbers of ephemeral processes, and desirable that they communicate frequently.

A great many mechanisms have been suggested for the taming of concurrency, such as semaphores, data monitors, condition queues, critical regions, remote procedure calls and rendezvous, even the disciplined use of shared store. In each case, the aim is to get several processes to co-operate, rather than interfere with each other. The mechanism is to reduce the amount of concurrency in the program by forbidding certain sequences of execution. The essence distilled from these experiments by a number of theoreticians is that the only way to make processes co-operate, usefully to communicate information, is to arrange that their actions be synchronized when they need to interact.

The *occam* language inherits the tradition of theoretical study, being more than reminiscent of recent work on the mathematics of synchronization. It is intended by its devisers as the 'assembly code' of the INMOS transputer, a microprocessor designed to be used in relatively large numbers to make a single machine, yet capable of managing a large number of concurrent tasks within the one processor. The programmer is therefore encouraged to think of process creation, and of synchronization and scheduling operations, as being as cheap as any other 'primitive' actions. Even on a conventional processor, neither process creation nor scheduling need be any more expensive than, say, procedure invocation. This startling scale of costs gives the programmer much greater freedom of expression, and leads to an unaccustomed programming style. If you think that this book is about anything, I would prefer that you think it is about this style.

The book begins with an introduction to *occam* sufficient to make the rest of the work intelligible to anyone who is already able to write and understand sequential programs. Some *occam* idioms are exercised to help to give you an understanding of the way the language can be used. The bulk of the text is devoted to a number of sizeable examples; these are programs written to give me material to discuss in teaching the use of *occam*.

At the back of the book you will find a concise summary of the *occam* language, drawn from material in the *Programming Manual*, published in this same series. Finally, there are reproduced the complete codes of the larger examples. I do not expect that anyone might want to read hundreds of lines of code, but their presence may make it easier to settle ambiguities in the presentation in the text.

* * *

Several people will be content to believe that they encouraged me to put this material between covers, some of them did, for which I am duly grateful, but I would not dream of burdening anyone else with a suspicion of the responsibility for such inadequacies as remain.