Modula-2 which share some of Ada's philosophy and design goals. In other words, those who might learn most from the book are the least likely to read it.

FRANK BOTT
University of Wales, Aberystwyth

BRUCE P. LESTER
The Art of Parallel Programming. Prentice Hall International Editions, 1993, 375 pp., softbound, ISBN 0 13 074980 X

THOMAS BRÄUNL
Parallel Programming, An Introduction. Prentice Hall International Editions, 1993, 270 pp., softbound, ISBN 0 13 336827 0

It is generally accepted that the speed of computers organized according to the von Neumann model is reaching its physical limit. In order to overcome this problem, efforts have been focused on the design of *parallel* machines as well as languages to program them. Especially over the last decade the significance of parallel computing has become increasingly clear. Various architectures have been suggested and algorithms have been developed to perform efficiently on them. For this reason, most universities with curricula in computer science are beginning to offer courses in this area.

Parallel programming is not an easy subject to master. It contains subtleties and complexities and it brings together elements from many different areas. In particular, writing a parallel program involves the design of a parallel algorithm, its implementation in an appropriate parallel language, debugging and performance evaluation, and at the same time a thorough knowledge of parallel computer architectures.

Many books have been written in this field, most of which are suited as reference books or for intensive graduate courses. In addition, the majority are mostly concerned with only one of the areas involved, such as parallel algorithm design or parallel architectures.

The Art of Parallel Programming by Lester is intended for a wider range of students and provides a general introduction and overview of parallel programming. With only a few prerequisites, it can be used at both the advanced undergraduate level or at the beginning of a graduate course.

Most of the existing books on parallel programming advocate the concept of abstract programming that hides the underlying architecture by presenting programs which are independent of particular machines. However, Lester stresses the importance of considering the target architecture while writing a program and throughout the book there is a continual interplay between parallel architectures, languages, algorithms and performance evaluation. The text is clearly written, well structured and organized according to major programming techniques. The material is covered thoroughly, leaving hardly any questions open and chapters are complete with summaries, numerous examples of parallel algorithms studied in depth as well as exercises and programming projects.

The feature that makes this book unique as an introduction to parallel programming is the software accompanying it. This software consists of an interactive and user-friendly system which is written in Pascal. It allows students to write their own programs in the parallel programming language Multi-Pascal, the language used in all examples in the text. Programs can be compiled and debugged with the aid of a powerful debugger and then their execution can be simulated. It is possible to specify characteristics of the required target machine such as the number of processors, whether the machine has shared or distributed memory and its topology. In this way the performance of the program can be compared for different architectures. Multi-Pascal was developed by the author and it consists of a simple set of parallel abstractions powerful to represent parallel algorithms for both multiprocessors and multicomputers.

Thus Lester's book makes an interesting and readable introduction to parallel programming and supplements reading and understanding issues concerning parallel algorithms with the experience of actually writing programs and seeing how they perform.

Bräunl's book *Parallel Programming an Introduction* is directed to a similar range of students. It pays particular attention to system architectures and how they can influence the development of parallel programs. However, whereas Lester encourages understanding of parallel programming through experience and examples, Bräunl concentrates on presenting and discussing issues of parallel programming. A number of interesting examples are included in the text but they are not discussed into much depth.

A notable flaw of the book is the following: even though it is a successful translation from German to English, its style makes it difficult to read continuously and the coverage is somewhat patchy. Often it is the case that not enough information is given to clarify figures and program listings and occasionally the tone becomes slightly instructional. Also, at points questions are left open but usually references are given to sources for further discussion. Still there are various features to recommend this book. Most of the book is comprehensible, and very informative. Problems of synchronous and asynchronous programming and possible solutions are covered clearly and at a well-judged level of detail. In addition, discussions concerning automatic parallelisation bring interesting insights to the subject. Unlike Lester's book, it contains parts on specific programming languages with information concerning their parallel constructs. A chapter of the book is dedicated to the introduction to Petri nets. Their significance as a tool for the definition of asynchronously parallel tasks is stressed but one may criticise the fact that hardly any

mention of them is made throughout the rest of the book. In all, the book would be undoubtedly useful as a reference book.

Examples of parallel algorithms are presented in Modula-P and Parallaxis for synchronous and asynchronous programming respectively. The interested student can obtain a free copy of the compilers and simulation systems as well as a Petri net simulation tool via ftp to experiment with. The book also includes exercises and a supplementary booklet with solutions is available from the publisher.

A. PHILIPPOU
University of Warwick

KATHY SPURR, PAUL LAYZELL, LESLIE JENNISON and NEIL RICHARDS (Eds)
Software Assistance for Business Re-Engineering. John Wiley, 1993, £26.95, 224 pp., hardbound, ISBN 0 471 94240 5

This book comprises the papers presented at the British Computer Society CASE specialist group seminar of 29th June 1993. Like all conference proceedings, it is a curate's egg. Business (Process) (Re-)engineering—its practitioners cannot agree on its syntax, let alone its semantics—takes up the baton of corporate data base and enterprise analysis. Not that B(P)(R)E is confined to IT, of course: it encompasses the entire enterprise and subsumes total quality management. It is nothing, many of the contributors tell us, if not holistic (so there's goodness for you).

Between three papers of introduction to the subject and two that address how the approach can be made to work, come six on proprietary methods and tools. However, the best introduction is probably the final paper, by Linda Hickman. The six central papers are much what we would expect: generally well written, with enough information to tempt and tease, but not enough to allow the reader any real understanding. They are, in a word, exercises in selling, some more blatant than others. One contributor lists seven selection criteria for process modelling tools and concludes: *Few process modelling tools in present use match up to these demanding requirements. The remainder of this paper discusses one tool ... which scores highly under every one of the issues discussed above.* Surprise surprise. The products range from mundane, but probably useful, tools for recording process analyses, to a process simulator that is object oriented (of course) and AI-based: just the sort of tool you would trust in a fundamental remodelling of your entire business.

The first of the selling exercises, by Julian Watts, briefly mentions the option of piecemeal process engineering—which cautious souls might favour—and gives a good account of the analysis of value streams (value-adding sequences of processes). Chris Haynes' introductory paper, the third and shortest, describes a somewhat different and rather promising value-based approach. But such nuggets are small, and few, and far between; the subject is new, its notions vague, and some of its verbiage makes one think that re-engineering has been applied to our mother tongue. Skip straight to Haynes' paper to avoid: *in the onrushing face of change ... Business Process Redesign as a concept is contained within the overall domain of change ... All this adds up to mean that the change is transformational ... New ways of working mean a change to the status quo ... rollout is the formal implementation and institutionalization of the innovation within the organizational mainstream.*

ADRIAN LARNER
De Montfort University

NADA LAVRAČ and SAŠO DŽEROSKI
Inductive Logic Programming. Ellis Horwood, 1994, £39.95, 293 pp., hardbound, ISBN 0 13 45870 8

Inductive Logic Programming (ILP) combines techniques drawn from inductive machine learning and logic programming with the principle aim of designing programs that generate (or induce) a set of general rules (clauses) from a given set of observations (facts). ILP also offers a rigorous and well-understood theoretical foundation and encouraging practical results when applied to real-world domain problems. Examples of applications include, automated knowledge acquisition inductive program synthesis and knowledge discovery in databases.

ILP splits into empirical and interactive approaches. Empirical ILP is used where numerous examples of a single concept and comprehensive background knowledge are available to the user; in contrast, interactive ILP covers cases where multiple concepts need to be acquired and where little background knowledge is available. This book concentrates on the former approach. In use empirical ILP assumes (i) a set of stipulated facts covering both positive and negative cases involving some unknown predicate in question, (ii) a description language which specifies syntactic restrictions on the target predicate, and (iii) background knowledge defining other predicates used to describe the domain and which are related to the target predicate. The goal is to find a set of clauses yielding a definition for the target predicate which accounts for the input stipulated facts and background knowledge.

The book is divided into four well-balanced parts: Part 1 introduces ILP; Part 2 is a detailed description of empirical ILP techniques and systems used; Part 3 considers the problem posed when using imperfect data; and Part 4 is an overview of ILP applications.

The authors draw upon years of research into theoretical and practical work on machine learning and qualitative modelling in an ILP framework. This integration works well in a book of this nature. As with any relatively new subject, material normally found in confer-