obtaining OCCAM programs from CSP and case studies. Sensibly enough, the equational theory of CSP is explained intuitively, without frightening the reader with mathematics. All important concepts are well illustrated by a number of small solved examples.

However, the authors seem to have got carried away by the lucidity of their own diagrams and explanations—so much so that they have also given in-line solutions to all their exercises. Where, then, is the distinction between an example and an exercise? The exercises and review questions are unfortunately not challenging enough for an 'intelligent' reader. (The review questions, by the way, are those for which the reader will *have* to go to the end of the part before getting at the answer.)

The major design examples in the book are a telephone network and an image-processing example. For some obscure reason the Towers of Hanoi problem has been studied—perhaps as an afterthought.

S. ARUN-KUMAR
Indian Institute of Technology
New Delhi

DANIEL PIERRE BOVET AND PIERLUIGI CRESCENZI
Introduction to the Theory of Complexity. Prentice-Hall. 1994. ISBN 0-13-915380-2. £29.95. 282 pp. hardbound.

This book by Bovet and Crescenzi is among the first dedicated to the theory of computational complexity. The main goal of this theory is the introduction of natural complexity classes composed of problems that have similar computational complexities and the study of the relations among these complexity classes. Computational complexity is closely related to the theory of algorithms. Less emphasis is given, however, to the exact complexity of specific computational problems and much more to the relations between various problems.

The book is composed of four main parts. The first part (Chapters 1–3) includes the necessary mathematical preliminaries. The second part (Chapters 4–8) introduces and studies the most significant complexity classes, most notably the classes P, NP, LOGSPACE and PSPACE. The third part (Chapters 9 and 10) deals with probabilistic algorithms and their corresponding complexity classes. The last part (Chapters 11 and 12) deals with parallel algorithms and parallel complexity classes. The table of contents of the book can be obtained, using the World Wide Web, from `http://dsi.uniroma1.it/~piluc/comp_book.html`.

The book is mainly suited for advanced undergraduates, and a course on computational complexity can be easily taught using it. Some of the material can also be covered in graduate courses.

The first two parts of the book roughly cover the material contained in the classical book *Computers and Intractability, A Guide to the Theory of NP-Completeness* by Garey and Johnson. The third and fourth parts contain much more recent material, most of which was not in existence when the book by Garey and Johnson was written. It is especially nice to find sections on the fascinating and recently introduced subjects of interactive proof systems and probabilistic checkable proofs.

The approach used by the authors in the book is, as they say, 'partly algorithmic and partly structuralistic'. Therefore the book nicely complements the books *Structural Complexity I* and *Structural Complexity II* by Balcázar, Díaz and Gabarró. Another recently published book on computational complexity theory is the voluminous *Computational Complexity* by Papadimitriou, which covers more material than the book under review.

The proofs given in the book are usually easy to follow. The text is written in a pleasant informal way, with remarks like 'yes, all this is very similar to alternating Turing machines' and 'Whoever at the beginning of this chapter thought 'interactive proofs are just another strange model of computation' should now be convinced that, as usual, science is full of surprises'.

A nice feature of the book is the abundance of exercises at the end of each chapter. Beware, however, of Exercise 4.10! A glossary of terms and notations would have been a useful addition.

URI ZWICK
Tel Aviv University

ALI MILI, JULES DESHARNAIS AND FATMA MILI
Computer Program Construction. Oxford University Press. 1994. ISBN 0-19-509236-8. £42.50. 379 pp. hardbound.

A binary, homogeneous relation is a set of ordered pairs whose components are drawn from the same set S. This book explains how such a relation can be used to specify a computer program. Because the elements of S can themselves be highly complicated data structures, this allows a very large class of systems to be specified.

The authors also show how this kind of specification can be refined into a program written in a high-level, imperative programming language. This is a task that cannot be fully automated—a fact that the authors are well aware of. They do, however, present a number of heuristics that are often useful in practice. These heuristics do not depend on any specific properties of the elements of S. They only require the relation in question to have certain general properties. The method of constructing programs that the book describes is, thus, elegant and based on a well-known mathematical theory.

It is unlikely, however, that in its present form the method described could be used on real-world problems. There are several reasons for this:

1. The specification method only deals with state-transformations and no account is given of how to handle input-output features.