

abduction are qualitatively different in that they are not necessarily a consequence of the given facts. Each deductive inference, by contrast, is a logical consequence of the facts. Thus abductive conclusions are ampliative and not guaranteed.

The book is an edited collection of material covering research in the Laboratory for Artificial Intelligence Research (LAIR) at Ohio State University. It begins with a very good first chapter which characterizes abduction in general and describes its relation to other kinds of inference and reasoning. It also gives several simple examples that explain exactly when abduction is warranted and how it is used in a variety of situations. For anyone wanting to know more about abduction, this is definitely the place to start.

The authors take great pains to distinguish abduction from other forms of inference such as deduction and induction, which they do very well. Abduction, however, is perhaps better regarded as a *process* of constructing explanations which can call upon various other forms of inference as necessary. In this sense, the definition of abduction is distinct from its implementation, for while the task is well-defined in terms of inputs and outputs, there are no prescribed rules for implementing it.

This is excellently demonstrated by the following chapters which describe several successive generations of abduction programs. The second chapter sets out the principles which guide the development of these abduction systems. First, it situates the work in the view of artificial intelligence which is concerned with the causal mechanisms that produce intelligent behaviour. Second, it introduces a *generic task* approach to building intelligent systems in which problem solving is functionally decomposed into small distinct units which are invoked according to the knowledge available and the current activity. This generic task approach is used as a basis for the construction of the systems detailed next.

Chapter 3 describes the first abduction machines developed at LAIR, RED-1 and RED-2, for identifying red-cell antibodies. On the basis of the experience gained through these systems, a programming tool named PEIRCE, described in Chapter 4, was built to enable easy construction of more flexible abductive systems, some examples of which are described in Chapter 5. Most of the remaining chapters are concerned with increasingly more sophisticated abduction machines, some conceptual and some implemented. These span systems for diagnosing mixed clinical diseases and human-gait disorders and, at the end of the book, abductive systems for perception and language understanding.

This succession of machines is punctuated in Chapter 7 by an analysis of computational complexity, with the conclusion that abduction is inherently intractable. In the subsequent chapters, therefore, the notion of abduction as inference to the *best* explanation is refined to one of finding an explanation for as much as can be *confidently and practically explained*. This is a reasonable

restriction to impose and one which offers good results in most situations, in a way similar to human capabilities.

In its approach of presenting what amounts to an historical record of research in abduction at LAIR, the book succeeds in conveying a very real sense of the progress that has been made. It integrates discussion of the broader concerns of abductive reasoning with descriptions of programs designed to explore those issues. Though the initial work was focused in the domain of medical diagnosis, more recent work has investigated the use of abduction in other areas such as speech understanding and perception, demonstrating its applicability in a number of domains.

Overall, the book provides a very good account of abduction, from the excellent introductory chapter through to the detailed descriptions of the programs developed. There are, however, two minor criticisms to be made. First, the book is made up of some previously published papers and some new writing by several different authors, but sewn together into a new whole by the editors. Mostly this works, but there are occasions when the seams show within chapters, causing some impediment to the flow in an otherwise well-tailored piece. Second, while some related work on abduction is explicitly discussed and compared with that reported in this book, other work in different domains is only briefly alluded to, and it would have been illustrative to have had further and more varied examples of abduction systems in other domains. Nevertheless, this book is a valuable work which draws together many diverse threads from different disciplines and shows how the richness of abductive reasoning can be captured effectively in computer programs.

M. LUCK
University of Warwick

PERCY METT, DAVID CROWE AND PETER STRAIN-CLARK
Specification and Design of Concurrent Systems.
McGraw-Hill Book Company. 1994. ISBN 0-07-707966-3. £21.95. 299 pp. softbound.

This book on system specification, written by faculty in the Computing Department of The Open University, takes the reader through the whole process of specification and design through the various steps. Starting with an informal English-like specification of the problem, the authors proceed to a semi-formal notation (accompanied usually by a diagram) from which a formal specification in the language of Communicating Sequential Processes (CSP) is obtained and the system then coded in OCCAM. Naturally all these formal and semi-formal notations need to be accompanied by copious amounts of detailed explanation. The authors have clearly spent a lot of time teaching and documenting this material.

The book is, appropriately enough, divided into four parts. The design method and process analysis constitute the first part. The other parts are devoted to CSP,

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.