effect of the books, and the very high quality of graphic design, layout, printing and production that has gone into them. No library should be without these books, no course on human–computer interaction should ignore them and every aspiring software designer should use them as meditation aids when contemplating how graphics can be used to represent information and interaction, choice and effect.

MATHAI JOSEPH
University of Warwick

ALISTAIR GEORGE AND MARK RICHES
Advanced Motif Programming Techniques. Prentice-Hall. 1994. ISBN 0-13-219965-3. £24.95. 406 pp. softbound.

The X Window System has developed rapidly in the last few years. A fundamental tool which is needed in order to program easily and effectively in X is a *toolkit*—this is a high-level library supporting a set of *widgets*. Several toolkits have been created, but only in the last couple of years has *Motif* begun to emerge as a *de facto* standard.

The concepts on which Motif is based are relatively simple and several good introductions to Motif are available. These all suffer from a common problem: Motif contains a very large (and somewhat complex) collection of routines and even simple GUI design can be effected in a variety of ways. *Advanced Motif Programming Techniques* contains advice and warnings to help a competent Motif programmer to design and write GUIs more effectively and efficiently.

As the title suggests, this is not a book for the novice X programmer. Although the book commences with a short discussion of the concepts underlying X and Motif, this is too brief to serve as a tutorial. It is assumed that the reader can write programs using Motif and has a good working knowledge of C. Discussion of Motif is supplemented with many code fragments which can be obtained either via FTP or on disk (for the latter a nominal sum is charged).

The book is divided into two parts. In the first, the authors discuss the Motif widgets and explain the rationale for them. Down-to-earth advice about good (and bad) techniques for building complex GUIs is given. These are illustrated by numerous examples of commonly-encountered situations. In the second part of the book other relevant topics are looked at, including: drawing using *Xlib*, use of colour, interaction between Motif and window managers, resources, and low-level X event handling. Some of the paragraphs are annotated with warning triangles, indicating traps for the unwary programmer. I liked this approach. The authors' style is very clear and I had no difficulty following their discussions.

Advanced Motif Programming Techniques is not a book to be read from cover to cover in one sitting; nor is it a reference manual. It is a volume to be consulted for good ideas and for common-sense. It is a valuable complement to your favourite Motif reference books.

MIKE JOY
University of Warwick

ZHAOHUI LUO
Computation and Reasoning. Oxford University Press. 1994. ISBN 0-19-853835-9. £30.00. 228 pp. hardbound.

Type theory concerns the formal study of the idea that entities in common use by computer scientists can be organized into collections with uniform properties (called *types*) together with rules for computing members of types. This gives rise to a conceptual framework for computer science. The book *Computation and Reasoning* is a comprehensive account of (the meta-theory and application of) one particular type theory, the *Extended Calculus of Constructions* (ECC), developed by Chaohui Luo in his recent PhD thesis.

The ECC provides a very rich type system together with an associated language of terms (programs). The syntax of the ECC is introduced, along with computational rules for evaluating terms to values (the results of programs). The formal presentation of the syntax is very clear and the informal explanations of the type theory guide the reader through the intended meaning of the ECC. Results about the meta-theoretic properties of the ECC are proved, such as principal typing and subject reduction. The ECC is shown to be strongly normalizing, i.e. every well typed program computes to a value in a finite number of steps. The method of proof is based on a well known technique (Girard–Tait reducibility) which is adapted in an interesting and novel way for the ECC; this proof will be of interest to the experts. The ECC contains a higher order logic, within which one may reason about programs; there is a concise explication of the logic and the proof of its consistency. A set-theoretic model of the ECC is described, which uses ideas from recursion theory. At first sight, the syntax of the ECC will (probably) seem quite complicated and this semantic model will aid the reader's understanding. However, to understand the model will require some knowledge of category theory. Examples of how to express computational theories in the ECC are given, and there is a very abstract account of how to specify and implement programs in the ECC. The book concludes with a lengthy account of a very general type-theoretic framework in which other type theories (such as the ECC) may be presented.

I found this to be a well written, thorough and enjoyable account of the ECC. As well as a detailed exposition of the meta-theory of the ECC, there are some lengthy philosophical discussions about type theory in general and its role in computer science. The author compares and contrasts the ECC with other well known type theories, in particular showing *how* the ECC extends the Calculus of Constructions and why the ECC may be

considered as an extension of Martin Löf's theory with universes by an (impredicative) type of logical propositions. Such philosophical discussion, together with the fact that the ECC is quite a complex type theory and that many of the proofs (of meta-theoretic results) are quite involved, means that this is not a book for the beginner. However, for someone with background knowledge of type theory I recommend *Computation and Reasoning* wholeheartedly.

ROY L. CROLE
Imperial College, London

IAN PRATT
Artificial Intelligence. Macmillan. 1994. ISBN 0-333-59755-9. £16.99. 280 pp. softbound.

There is a continuing and sometimes heated debate in Artificial Intelligence (AI) between the formalists and experimentalists, often characterized as 'neats' versus 'scruffies'. The former argue for the centrality of logic in AI and its use as a base for further research, while the latter claim that effort is better spent on content rather than form, and that logic can obscure and hinder research. Pratt's book lies in the formalist camp, but while it does indeed give a thorough grounding in the underlying theory of many aspects of AI and concentrate on those aspects that lend themselves to such analyses, it also highlights some of the inherent limitations.

The book can be considered in two parts. The first begins with an introductory chapter which sets the scene by stating that the subject of the book is that part of AI that is the 'enterprise of programming computers to make inferences'. It also describes a couple of motivating examples used subsequently to illustrate various techniques, and characterizes inference as involving *belief* and *heuristics*, key ideas which are further explored throughout the book.

The remainder of the first part includes five other chapters which cover search and planning, logic and inference, closed world assumptions, defeasible inference, and reason maintenance. The thread linking the topics of these chapters is developed coherently and convincingly from the initial example in the introduction concerning a robot moving around a house. Chapter 2 expands on the example by introducing technical details necessary for a fuller account of search and planning. Chapter 3 shows how logic can be used to formalize the previous discussion of planning, and in so doing explores the frame problem and possible solutions to it with alternatives to logic. The next two chapters then develop this further by considering topics in defeasible reasoning including circumscription, inheritance hierarchies and default logic. Finally, the first part ends with a chapter detailing aspects of truth maintenance.

By contrast, a second part of the book is much less cohesive. That this is so is less a criticism of the second part than an appreciation of the first. Here the chapters address such diverse topics as memory organization, probabilistic inference, induction and neural networks. In the preface, the author acknowledges the distinction between these two parts as core chapters and independent chapters, and balances the selectivity of material against the depth of those topics covered.

Much of the material in the book is quite demanding, and despite the appendix providing a tutorial introduction to predicate calculus, the claim made on the back cover that the book gives a 'clear and readable' introduction while 'assuming no prior knowledge of AI or logic' is perhaps a little optimistic. It is certainly well written, but covers the material to a greater depth than is usual in introductory texts and does so very quickly. Those without at least some prior knowledge may easily be intimidated.

Throughout the book, each chapter ends with a set of exercises including programming problems and suggestions for further reading. The suggestions are limited, however, and given that the material is treated in depth, it would have been worthwhile to have included a more extensive bibliography.

Overall, Pratt's book is an ambitious effort at a general AI text which covers its material in detail. It is largely successful and will certainly appeal to those who share Pratt's logical approach.

MICHAEL LUCK
University of Warwick

MARTIN SHEPPERD AND DARREL INCE
Derivation and Validation of Software Metrics. Oxford Science Publications. 1993. ISBN 0-19-853842-1. £30.00. 167 pp. hardbound.

In the words of the authors, software metrics is *the application of quantitative methods to software engineering*. In fact, while quantitative methods lie at the heart of traditional engineering disciplines, until 3 years ago software metrics was widely viewed as a peripheral subject within software engineering. There were less than a handful of books which covered any aspect of the subject in any depth at all. However, things are changing rapidly. If we judge the importance of a subject by the volume of books produced, then software metrics must now have truly arrived into the mainstream of software engineering. At least 30 books on software metrics have been published in the last three years. There have been a number of high-profile ESPRIT technology transfer projects and a mushrooming of conferences; it appears that metrics is a boom area.

So the question is: given this explosion of very recent activity, does this book offer anything substantially different? I believe it does, although inevitably some of the ground is covered elsewhere. This book concentrates on metrics which can be determined *early* in the software development life-cycle; specifically metrics that can be computed from designs and specifications. The authors