Section 1 of the book (400 pages) provides descriptions of 18 systems, starting at 1975 with Pygmalion and ending with systems still under development in 1992. The complete list, with dates and authors, is

> Pygmalion (1975, Smith), Tinker (1979, Lieberman), A Predictive Calculator (1982, Witten), Rehearsal World (1984, Finzer and Gould), SmallStar (1984, Halbert), Peridot (1987, Myers), Metamouse (1988, Maulsby and Witten), TELS (1988–9, Witten and Mo), Eager (1990, Cypher), Garnet (1988 onwards, Myers), The Turvey Experience (1990, Maulsby), Chimera (1990, Kurlander), The Geometer's Sketchpad (1991, Jackiw and Finzer), Tourmaline (1990–2, Myers), A History-based Macro by Example System (1991, Kurlander and Feiner), Mondrian (1991, Lieberman), Triggers (1991, Potter) and The AIDE Project (1992, Piernot and Yvon).

As a reader with little previous experience in the field, I found the descriptions generally interesting and occasionally fascinating. To give an idea of the range of systems included: Rehearsal World allows users to create applications by manipulating 'actors' that respond to 'cues', Eager attempts to predict users' intentions and asks to take over when it thinks it can help, Peridot is a tool to help non-programmers create graphical interactive interfaces, and Chimera is an editing system that provides global search-and-replace facilities for graphics.

Sections 2 and 3 (100 pages) provide nine short Chapters that look in more detail at specific issues and implementation techniques. The book is rounded off by three Appendices (a short chronology of developments, a test suite and a glossary), a bibliography, a list of contributors and an index.

Overall, this is a useful and readable collection of information about Programming by Demonstration. Unlike many books arising from conferences or workshops, the quality is uniformly good and the style is consistent. I became more enthusiastic as I read through it.

H. BROWN
Kent University

JEFFREY D. ULLMAN
Elements of ML Programming. Prentice-Hall, 1994, £19.95, 320 pp. softbound, ISBN 0-13-288788-6

From humble beginnings, the programming language ML has grown greatly in popularity to its position today, where there are a variety of implementations of the language and many adherents. This book aims to teach the rudiments of functional programming in ML, to programmers proficient in a conventional imperative language such as Pascal or C.

The book is divided into three sections. The first section is an introduction to ML, with the emphasis on the understanding of concepts rather than complete detail. Thus in this section we are introduced to ML's type system and the binding of identifiers to values. The latter is treated thoroughly, with a discussion of the difference between assigning a value to an identifier in ML, and assignment in an imperative language. This section also introduces lists, functions (including recursion) and patterns. Throughout this section, new concepts are introduced by careful example, with occasional analogies drawn from imperative programming.

The second section introduces more advanced features of ML such as polymorphism, higher-order functions, type definitions, references and ML's module system. The power of ML is illustrated by the use of non-trivial examples, such as binary search trees and hash tables. Again, concepts are presented with many details postponed until later.

The third and final section fills in the gaps left by the preceding sections. In this section we find chapters on matches, exceptions, curried functions and also chapters on practical issues such as creating executable files. The book is rounded off with a complete summary of ML syntax.

The book contains numerous examples ranging from simple declarations to an ML implementation of a numeric integration algorithm. The ML code for all the examples is available by anonymous FTP, so that readers may experiment with them. Each chapter also includes several exercises of varying difficulty, and selected answers are included.

The book's greatest drawback is that the concept of curried functions is postponed to Section 3. Setting aside the importance of currying itself, this means that the treatment of higher-order functions becomes needlessly complicated, and the full power of functions such as *map* and *filter* cannot be demonstrated.

This aside, the book is a well-written introduction to ML. The author succeeds in making concepts, which at first appear complex, quite simple. The treatment of type definitions and ML's module system, is particularly impressive, and was a pleasure to read. For programmers experienced in imperative programming who desire a practical introduction to the language ML, this would be an appropriate text.

P. MUKHERJEE
Birmingham University

BRUCE HILLAM
Introduction to Abstract Data Types using Ada. Prentice-Hall, 1994, £19.95, 662 pp. softbound, ISBN 0-13-124215-6

Producing a textbook requires substantial care and dedication. To produce an Ada textbook of 662 pages is a major undertaking. Unfortunately, the high standard that the reviewer thinks one should expect from a major publisher has not been attained. Virtually every page contains some misprint or small error. To publish the book in this form is a disservice to the student purchasers and surely cannot be in the long-term interest of the publishing industry.

Since every lecturer has at least half a textbook in his/her course notes, the temptation to publish is irresist-