Lisp, Second Edition

Patrick H. Winston and Berthold K. P. Horn; Addison-Wesley, 1984, 434 pp, $25.95

Reviewed by Daniel Weinreb, Symbolics Inc.

Winston and Horn's *Lisp* is a textbook on symbolic computation and the Lisp language. It was originally published in 1981, based on a set of course notes developed for the introductory course in artificial intelligence at MIT.

When the first edition was written, a wide range of Lisp dialects were in use throughout the AI community, and in order to make sure that the examples in the book would work properly, the authors restricted themselves to a "lowest common denominator" subset of Lisp. Unfortunately, this subset was weak and primitive, and the examples were forced to use poor programming practices and style. In the second edition, every program has been rewritten in Common Lisp using modern style. The code is now as crisp and clear as the text.

The book has two parts. The first part introduces the ideas of symbol manipulation and the basics of Lisp programming. The first nine chapters introduce the basic concepts of Lisp: the elementary data and control structures, printing and reading, "destructive" modification, and so on. The book is all-out in support of Common Lisp, and mentions many language features normally associated specifically with Common Lisp, such as lexical scoping, **remove-if**, and optional parameters.

The next three chapters present examples drawn from the realms of binary images, search, and mathematics. Chapter 10 is built around an extended example program that does simple classification of binary images. This chapter introduces Lisp arrays. It also gives the reader a sense of the basics of computer vision software, and demonstrates that Lisp can be used in a style like that of other programming languages.

Chapter 11 illustrates basic search strategies with several small examples. It shows how to model a real-world problem using Lisp's data structures, and goes on to show some sample problems that can be solved using the paradigm of searching. Chapter 12 contains several examples from mathematics, including infix-to-prefix translation, sparse matrix representation, representing complex numbers and finding algebraic roots, and so on. These examples introduce basic Lisp techniques such as tail-recursive style, data abstraction, using lists as push-down stacks, and so on.

The second part of the book gives further demonstrations of how to use Lisp on somewhat larger problems. The authors say that its purpose is to "demonstrate Lisp's muscle and to excite people about what Lisp can do."

Chapter 13 explains the classic "blocks world" problem, and outlines a program to solve it. Chapter 14 uses the blocks world example to demonstrate points about good programming style, and to illustrate the use of debugging tools such as insertion of breakpoints, tracing, and stepping. Chapter 15 develops the blocks world program further, adding a history facility so that the program can explain what it has done. This chapter illustrates procedure-defining procedures, an

important Lisp technique which can only be appreciated in the context of a developed example such as this one.

Chapter 16 introduces the basic ideas of object-centered programming. It shows how to implement some of the rudiments out of plain Common Lisp, and then begins to explore inheritance and method combination. However, it doesn't get very far, and doesn't offer any serious examples that illustrate how inheritance is used. It seems that the authors have again run up against the language barrier. Common Lisp will presumably have a standard language extension for object-centered programming, and I hope the next edition will be able to take advantage of it.

Chapter 17 discusses pattern matching, and works through several examples of progressively more sophisticated matching functions. Chapter 18 builds on the basic idea of pattern matching to build a simple rule-based system, and gives a simple example "expert system" to identify animals. Chapter 19 shows an interpreter for an augmented transition tree, a classic data structure used in parsing natural language, to illustrate how to make an embedded language within Lisp. Chapter 20 goes on to show how to write a compiler for augmented transition trees, showing the power of Lisp's ability to manipulate programs as data. Chapter 21 goes further in that direction, showing a program that wries its own search procedures in Lisp. Chapter 22 shows how to represent frames using Lisp data structures, illustrating defaults, demons, and inheritance. The frame language is a more elaborate embedded language within Lisp, and leads to a discussion of writing Lisp in Lisp.

Every chapter is lavishly supplied with problems, the solutions to which are provided at the end of the book. Each chapter has a summary of the new ideas of the chapter, and annotated references to relevant academic literature. The book design and typesetting are excellent. Program examples are properly indented, well-commented, and written in proper modern Lisp programming style.

For introducing a new programmer to Lisp, Winston and Horn's book is less gentle than Touretzky's *Lisp: A Gentle Introduction to Symbolic Programming*. It goes more quickly, and is generally more demanding. Winston and Horn wrote primary for MIT freshmen, who generally have a strong background in computer programming before the arrive at college. In general, Winston and Horn's book is best suited for computer science majors, or people who are serious about wanting to do real programming in Lisp. Readers should have firm grounding in some programming language, be willing to work hard, or both. However, all the topics are covered, and the text is clear and well-organized, so it should be possible for anyone to use it.

The great strength of the book lies in its examples. For one thing, if you are interested in going on to learn more about artificial intelligence, the examples are drawn from many fields in AI and are a good way to get started. But more important is the number and depth of the examples. Learning Lisp consists not only in learning the rules of the language and what all the functions and special forms do, but understanding how to put them together to solve real problems. The book provides examples that are sufficiently small that you can understand them, but sufficiently large that they illustrate and motivate important techniques of symbolic programming. In this, Winston and Horn's *Lisp* is unsurpassed.