AI Magazine Volume 15 Number 4 (1994) (© AAAI)

Book Reviews

Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp

Robert P. Futrelle

Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp, Peter Norvig, Morgan Kaufmann Publishers, San Mateo, California, 1992, 946 pp., $49.95, ISBN 1-55860191-0.

Norvig's *Paradigms of Artificial Intelligence Programming* is a splendid work that skillfully weaves together three threads: (1) AI, (2) programming skills in general, and (3) programming skills in Common Lisp in particular. It is written with great intelligence and insight and can benefit a wide audience from advanced undergraduates to seasoned researchers. It is a book that should be in the permanent collection of every AI aficionado because it is such a rich source of ideas and examples. It is not a full-blown AI text; it does not depend on the reader having any previous knowledge of AI but does assume some basic knowledge of Lisp. I have used this book with great success as a supplement in an introductory graduate AI course, the text in a graduate AI course focusing on techniques, and a resource in my research group.

The sheer amount of material is impressive, including symbolic mathematics, logic programming, natural language, expert systems, games, and more. Perhaps even more important, however, are the numerous paradigms introduced, in keeping with the book's title. These paradigms include data-driven programming, search and heuristics, pattern matching, rule-based translation, efficiency-related caching and compiling, the creation of object-based facilities, compilation to an abstract machine, and troubleshooting and trouble-avoidance strategies. The most notable omission from the book is any discussion of machine learning, a topic of growing importance in all areas of AI.

The book is nearly 1,000 pages long with a historical end note in each chapter, extensive references, and a thorough index. The 10,000 lines of source code discussed in detail in the book are available online. Each of the 25 chapters includes approximately 10 exercises of widely varying difficulty or time to complete. Each major section of each chapter includes a g*lossary,* a table that lists the macros, top-level functions, special variables, data types, functions, selected Common Lisp functions, and previously defined functions that are discussed and used in the section. The book includes pointers to other major AI software sources, although these pointers are mostly superseded by the large collection in /pub/ai at ftp.cs.cmu.edu (also available on CD-ROM). The frequently asked questions (FAQs) in the comp.lang.lisp newsgroup are another useful adjunct to the text.

The book starts with a refresher on Lisp. Norvig states that one of the best ways to learn how to program well is to read lots of good code. The code in this book is well written and edifying and pleasurable to read, although it is sometimes rather challenging. Later chapters deal with advanced features of Lisp and troubleshooting. He argues cogently for the use of Lisp because it is open ended and has uniform data structures and the ability to manipulate and generate code through macros and other mechanisms. CLOS is not used in the book, but a limited form of object-oriented programming is developed using closures. Modern Lisps can also be efficient, and the book is replete with examples of strategies to produce fast code.

The second chapter, "A Simple Lisp Program," illustrates the author's strategy of successive refinement. The example is a program to generate English sentences, starting with a set of nested function calls. This program has obvious problems, so a second, more modular, data-driven solution is developed that separates the grammar from the interpreter. Then, a tree printer is integrated into the generator. These themes of successive refinement and data-driven programming are pursued with great success throughout the book.

Much of the first part of the book is devoted to exploring early AI paradigms by recreating historically important systems. The first is Newell and Simon's GENERAL PROBLEM SOLVER (GPS). The treatment of GPS is good, presenting many useful ideas and pedagogical strategies in the clear and interesting style that pervades the book. A systematic strategy is introduced for system development, going from description to specification to implementation to testing and, finally, to debugging and analysis. Norvig discusses five problems encountered with the simple STRIPS-based GPS implementation, followed by the implementation of a version with a somewhat different architecture that solves the problems raised. Instead of generating a new set of data to satisfy

the newly required format, he writes a function that converts the old data format to the new. Elegant and sensible touches such as this one occur throughout the book. The enhanced GPS is applied to mazes, monkey and bananas, and the blocks world. The blocks world application points out the fundamental limitations of the GPS paradigm, and the chapter ends with some sobering reflections on why GPS cannot handle a wide range of important problems or at least cannot do so without unacceptable computational costs.

ELIZA is treated briefly in a similar vein. Tools for pattern matching are developed. A general-purpose rule-based translator, mapping from user input to action, is constructed using the pattern matcher and is applied to ELIZA and GPS. The book takes pains to point out just how brittle the ELIZA paradigm is.

This book can lead to a deep appreciation of the unique strengths of Lisp and help the reader understand why Lisp is still in active use nearly 40 years after it was first developed

Search is introduced, and in only 20 pages, an integrated set of search tools is built and demonstrated. All the standard methods of search are covered, including A* and iterative deepening. GPS is reformulated as a search problem, so that even the Sussman anomaly can be solved. This point is well taken: The solution of many AI problems involves some form of search; so, search is a foundational topic. A minor point: Although the handling of tail recursion is discussed later in the book, no mention is made that the tree-search routines here should be compiled by a compiler that handles tail recursion (to avoid blowing the stack on large searches).

A treatment of Bobrow's 1964 STUDENT system introduces a number of new ideas, including analyzing English input, using a rule-based translator to generate symbolic equations, and solving the equations to get the unknown values. This task seems daunting, but it's all accomplished in 16 pages of lucid code and text. The book points out that some of the approaches used in the STUDENT system have not survived, especially its handling of natural language. It further points out that symbolic integration was once an outstanding AI problem but is now accomplished by deterministic algorithms and is available in packages for PCs. There are chapters later in the book that deal with natural language and symbolic mathematics in a more sophisticated and up-to-date manner and include case studies in performance enhancement. The most advanced chapters on symbolic mathematics deal with rather complicated material and will not be easy for some readers.

Two chapters are devoted to efficiency issues and include material that is hard to find elsewhere. Most notably, *memoization* is implemented, a transparent method of caching results to avoid reevaluation. There is also a discussion of compiling one language into another, turning interpreters into compiled code. All these ideas are applied to a symbolic mathematics program, getting a speedup of 130. The techniques are used later in extensive studies of Prolog and SCHEME.

A significant portion of the book discusses Prolog, develops it in Lisp and then uses it, primarily for natural language analysis. An interpreted version of Prolog is built first, then refined with destructive unification and compiled into Lisp so that it runs 20 to 200 times faster. Some readers might find the chapter on compiling Prolog a bit daunting because various features are added, such as call, bagof, and cut.

Issues of knowledge representation are discussed with an approach that is refreshingly different from the many standard AI textbook treatments. The limitations of the predicate calculus and Prolog are discussed, and partial workarounds for them are developed by altering the Prolog system. The problems treated in this way include efficiency of indexing and completeness. The expressiveness problem is dealt with by building some facilities for higher-order predications, frames, possible worlds, and negation and disjunction.

Many AI students are eager to learn about expert systems. The book rewards them with a fine chapter devoted to a reconstruction of EMYCIN, with medical rules added to build MYCIN. The development is entirely within Lisp, but the relations to Prolog are carefully explained as consisting of Prolog plus uncertainty, explanations, and other features.

Constraint satisfaction is discussed by implementing the Waltz line-labeling algorithm. Unfortunately, the treatment is not related to the current literature on constraint satisfaction; nor is any of the terminology of the field used; nor, in the chapters on Prolog, is the related field of constraint logic programming mentioned. (Perhaps the latter development is just too recent.) In spite of some ingenious ideas and code, the general nature of constraints in problem solving is not explored deeply enough in this chapter. To pursue the topic further, the reader is advised to start with a 1992 special issue of the *AI Journal*, 58(1–3), or the book by E. Tsang, *Foundations of Constraint Satisfaction* (Academic Press, 1993).

Games are always of great interest to students of AI. Norvig devotes a satisfying chapter to search and the game of OTHELLO. He develops search heuristics and systematically improves them, culminating in a system of tournament-level quality that uses techniques from the high-end programs IAGO and BILL.

One of this book's many strengths is its treatment of natural language processing. The topic is introduced with an exhilarating tour of the basics of natural language parsing by building a simple parser for a phrase-structure grammar, then generating

multiple parses for ambiguous sentences, dealing with unknown words, integrating compositional semantics, and integrating preference-based parsing using the same strategy. A speedup of 10 to 20 is obtained by memoizing the parser. Elsewhere, Norvig has shown that careful memoization of a simple parser can match the performance of the more complex chart parsers (1991, Techniques for Automatic Memoization with Applications to Context-Free Parsing, *Computational Linguistics* 17(1): 91–98).

Unification grammars for natural language, a thoroughly modern approach, are discussed at length. Norvig uses Prolog in this discussion, which might seem an odd choice for a book built around Lisp; however, the fact that in Prolog, the grammar **is** the code, as it were, gives the entire presentation great clarity. To go beyond this simple initial treatment, definite clause grammars (DCGs) are introduced. The economy of his treatment leaves room for a discussion of interesting issues, such as quantifiers and their scope, ambiguity, and long-distance dependencies. An additional chapter is taken up entirely with presenting an extensive DCG grammar of English, making this textbook possibly the only one that includes both a working parser and an extensive grammar. Studying such a large grammar is informative, but the reader should have been warned that such large grammars, applied to real English (with 40+ word long sentences such as this one), have to deal with serious problems of performance that are mostly driven by lexical and structural ambiguity problems.

This book can lead to a deep appreciation of the unique strengths of Lisp and help the reader understand why Lisp is still in active use nearly 40 years after it was first developed. The ideas underlying Lisp are powerful and pervasive, so they will be with us for a long time (even if the language is reborn as DYLAN or as some future language yet unknown). The paradigms of AI programming so artfully presented in this book will remain valuable well into the future.