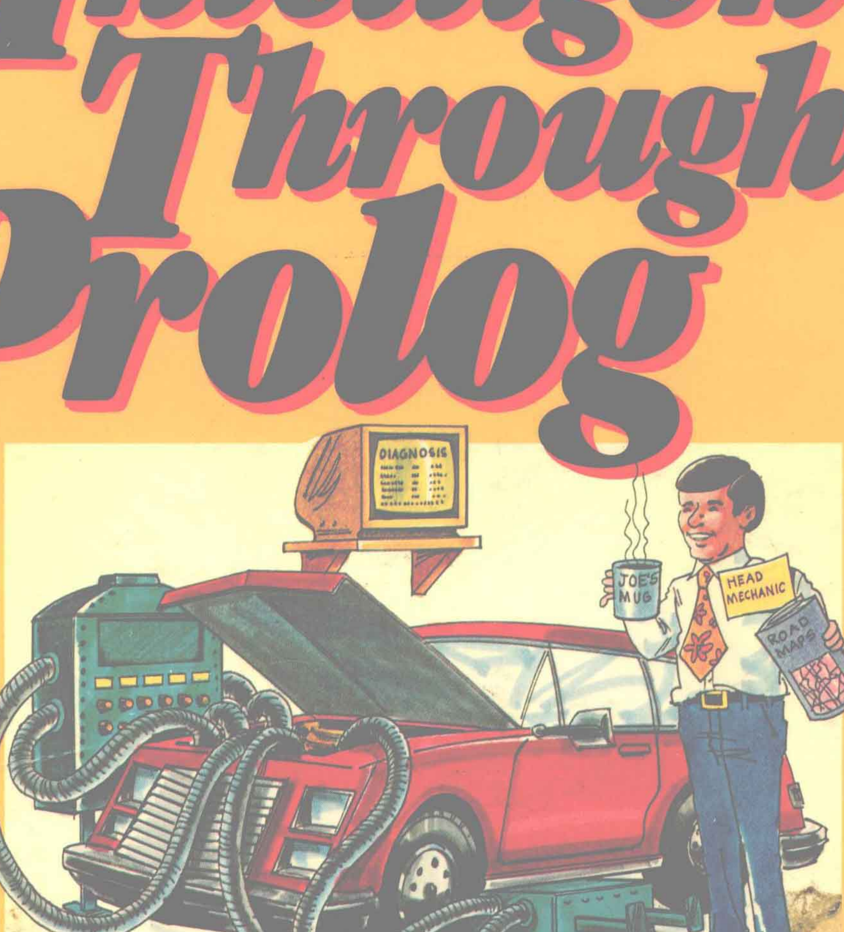
Artificial Intelligence Through Prolog



Neil C. Rowe

ARTIFICIAL INTELLIGENCE THROUGH PROLOG

NEIL C. ROWE

U.S. Naval Postgraduate School

PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Rowe, Neil C.
Artificial intelligence through Prolog / Neil C. Rowe.
p. cm.
Bibliography: p.
Includes index.
ISBN 0-13-048679-5
1. Prolog (Computer program language) 2. Artificial intelligence.
I. Title.
QA76.73.P76R69 1988
005. 13'3—dc19 87-26219
CIP

Editorial/production supervision: Claudia Citarella
Cover design: Lundgren Graphics, Ltd.
Manufacturing buyer: S. Gordon Osbourne

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-048679-5 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

PREFACE

Artificial intelligence is a hard subject to learn. I have written a book to make it easier. I explain difficult concepts in a simple, concrete way. I have organized the material in a new and (I feel) clearer way, a way in which the chapters are in a logical sequence and not just unrelated topics. I believe that with this book, readers can learn the key concepts of artificial intelligence faster and better than with other books. This book is intended for all first courses in artificial intelligence at the undergraduate or graduate level, requiring background of only a few computer science courses. It can also be used on one's own. No prior knowledge of the language Prolog is assumed.

Students often complain that while they understand the terminology of artifical intelligence, they don't have a gut feeling for what's going on or how you apply the concepts to a situation. One cause is the complexity of artificial intelligence. Another is the unnecessary baggage, like overly formal logical calculi, that some books and teachers saddle students with. But an equally important cause is the often poor connection made between abstract concepts and their use. So I considered it essential to integrate practical programming examples into this book, in the style of programming language and data structures books. (I stress *practical*, not missionaries and cannibals, definitions of "grandfather," or rules for identifying animals in zoos—at least rarely.) This book has about 500 chunks of code. Clear, concrete formalization of artificial-intelligence ideas by programs and program fragments is all the more critical today with commercialization and media discovery of the field, which has caused a good deal of throwing around of artificial-intelligence terms by people who don't understand them.

But artificial intelligence is a tool for complex problems, and its program examples can easily be forbiddingly complicated. Books attempting to explain artificial intelligence with examples from the programming language Lisp have repeatedly demonstrated this. But I have come to see that the fault lies more with Lisp than with artificial intelligence. Lisp has been the primary language of artificial intelligence for many years, but it is a low-level language, too low for most students. Designed in the early 1960s, Lisp reflects the then-primitive understanding of good programming, and requires the programmer to worry considerably about actual memory references (pointers). Furthermore, Lisp has a weird, hard-to-read syntax unlike that of any other programming language. To make matters worse, the widespread adoption of Common Lisp as a de facto standard has discouraged research on improved Lisps.

Fortunately there is an alternative: Prolog. Developed in Europe in the 1970s, the language Prolog has steadily gained enthusiastic converts, bolstered by its surprise choice as the initial language of the Japanese Fifth Generation Computer project. Prolog has three positive features that give it key advantages over Lisp. First, Prolog syntax and semantics are much closer to formal logic, the most common way of representing facts and reasoning methods used in the artificial-intelligence literature. Second, Prolog provides automatic backtracking, a feature making for considerably easier "search," the most central of all artificial-intelligence techniques. Third, Prolog supports multidirectional (or multiuse) reasoning, in which arguments to a procedure can freely be designated inputs and outputs in different ways in different procedure calls, so that the same procedure definition can be used for many different kinds of reasoning. Besides this, new implementation techniques have given current versions of Prolog close in speed to Lisp implementations, so efficiency is no longer a reason to prefer Lisp.

But Prolog also, I believe, makes teaching artificial intelligence easier. This book is a demonstration. This book is an organic whole, not a random collection of chapters on random topics. My chapters form a steady, logical progression, from knowledge representation to inferences on the representation, to rule-based systems codifying classes of inferences, to search as an abstraction of rule-based systems, to extensions of the methodology, and finally to evaluation of systems. Topics hard to understand like search, the cut predicate, relaxation, and resolution are introduced late and only with careful preparation. In each chapter, details of Prolog are integrated with major concepts of artificial intelligence. For instance, Chapter 2 discusses the kinds of facts about the world that one can put into computers as well as the syntax of Prolog's way; Chapter 3 discusses automatic backtracking as well as Prolog querying; Chapter 4 discusses inference and inheritance as well as the definition of procedures in Prolog; Chapter 5 discusses multidirectional reasoning as well as the syntax of Prolog arithmetic; and so on. This constant tying of theory to practice makes artificial intelligence a lot more concrete. Learning is better motivated since one doesn't need to master a lot of mumbo-jumbo to get to the good stuff. I can't take much of the credit myself: the very nature of Prolog, and particularly the advantages of the last paragraph, make it easy.

Despite my integrated approach to the material, I think I have covered nearly all the topics in ACM and IEEE guidelines for a first course in artificial intelligence. Basic concepts mentioned in those guidelines appear toward the beginning of chapters, and applications mentioned in the guidelines appear toward the ends. Beyond the guidelines however, I have had to make tough decisions about what to leave out—a coherent book is better than an incoherent book that covers everything. Since this is a first course, I concentrate on the hard core of artificial intelligence. So I don't discuss much how humans think (that's psychology), or how human language works (that's linguistics), or how sensor interpretation and low-level visual processing are done (that's pattern recognition), or whether computers will ever really think (that's philosophy). I have also cut corners on hard noncentral topics like computer learning and the full formal development of predicate calculus. On the other hand, I emphasize more than other books do the central computer science concepts of procedure calls, variable binding, list processing, tree traversal, analysis of processing efficiency, compilation, caching, and recursion. This is a computer science textbook.

A disadvantage of my integrated approach is that chapters can't so easily be skipped. To partially compensate, I mark some sections within chapters (usually sections toward the end) with asterisks to indicate that they are optional to the main flow of the book. In addition, all of Chapters 7, 10, and 14 can be omitted, and perhaps Chapters 12 and 13 too. (Chapters 7, 10, 13, and 14 provide a good basis for a second course in artificial intelligence, and I have used them that way myself.) Besides this, I cater to the different needs of different readers in the exercises. Exercises are essential to learning the material in a textbook. Unfortunately, there is little consensus about what kind of exercises to give for courses in artificial intelligence. So I have provided a wide variety: short-answer questions for checking basic understanding of material, programming exercises for people who like to program, "play computer" exercises that have the reader simulate techniques described, application questions that have the reader apply methods to new areas (my favorite kind of exercise because it tests real understanding of the material), essay questions, fallacies to analyze, complexity analysis questions, and a few extended projects suitable for teams of students. There are also some miscellaneous questions drawing on the entire book, at the end of Chapter 15. Answers to about one third of the exercises are provided in Appendix G, to offer readers immediate feedback on their understanding, something especially important to those tackling this book on their own.

To make learning the difficult material of this book even easier, I provide other learning aids. I apportion the book into short labeled sections, to make it easier for readers to chunk the material into mind-sized bites. I provide reinforcement of key concepts with some novel graphical and tabular displays. I provide "glass box" computer programs (that is, the opposite of "black box") for readers to study. I mark key terms in italics where they are defined in the text, and then group the most important of these terms into keyword lists at the end of every chapter. I give appendices summarizing the important background material needed for this book, concepts in logic, recursion, and data structures. In other appendices, I summarize the Prolog dialect of the book, make a few comments on Micro-Prolog, and provide a short bibliography (most of the artificial intelligence literature is now either too hard or too easy for readers of this book). The major programs of the book are available on tape or diskette from the publisher for a small fee. Also, I have prepared an instructor's manual.

It's not necessary to have a Prolog interpreter or compiler available to use this book, but it does make learning easier. This book uses a limited subset of the most common dialect of Prolog, the "standard Prolog" of *Programming in Prolog* by Clocksin and Mellish (second edition, Springer-Verlag, 1984). But most exercises do not require programming.

I've tried to doublecheck all examples, programs, and exercises, but some errors may have escaped me. If you find any, please write me in care of the publisher, or send computer mail to rowe@nps-cs.arpa.

ACKNOWLEDGMENTS

Many people contributed ideas to this book. Michael Genesereth first suggested to me the teaching of introductory artificial intelligence in a way based on logic. David H. Warren gradually eroded my skepticism about Prolog. Harold Abelson and Seymour Papert have steered my teaching style toward student activity rather than passivity.

Judy Quesenberry spent many long hours helping me with the typing and correction of this book, and deserves a great deal of thanks, even if she ate an awful lot of my cookies. Robert Richbourg has been helpful in many different ways, in suggesting corrections and improvements and in testing out some of the programs, despite his having to jump through all the hoops Ph.D. students must jump through. Richard Hamming provided valuable advice on book production. Other people who provided valuable comments include Chris Carlson, Daniel Chester, Ernest Davis, Eileen Entin, Robert Grant, Mike Goyden, Simon Hart, Greg Hoppenstand, Kirk Jennings, Grace Mason, Bruce MacLennan, Norman McNeal, Bob McGhee, James Milojkovic, Doug Owen, Jim Peak, Olen Porter, Brian Rodeck, Jean Sando, Derek Sleeman, Amnon Shefi, and Steve Weingart. Mycke Moore made the creative suggestion that I put a lot of sex into this book to boost sales.

Besides those named, I am grateful to all my students over the years at the Massachusetts Institute of Technology, Stanford University, and the Naval Postgraduate School for providing valuable feedback. They deserve a good deal of credit for the quality of this book—but sorry, people, I'm poor and unwilling to share royalties.

TO THE READER

Artificial intelligence draws on many different areas of computer science. It is hard to recommend prerequisites because what you need to know is bits and pieces scattered over many different courses. At least two quarters or semesters of computer programming in a higher-level language like Pascal is strongly recommended, since we will introduce here a programming language several degrees more difficult, Prolog. If you can get programming experience in Prolog, Lisp, or Logo, that's even better. It also helps to have a course in formal logic, though we won't use much of the fancy stuff they usually cover in those courses; see Appendix A for what you do need to know. Artificial intelligence uses sophisticated data structures, so a data structures course helps; see Appendix C for a summary. Finally, you should be familiar with recursion, because Prolog is well suited to this way of writing programs. Recursion is a difficult concept to understand at first, but once you get used to it you will find it easy and natural; Appendix B provides some hints.

Solving problems is the best way to learn artificial intelligence. So there are lots of exercises in this book, at the ends of chapters. Please take these exercises seriously; many of them are hard, but you can really learn from them, much more than by just passively reading the text. Artificial intelligence is difficult to learn, and feedback really helps, especially if your're working on your own. (But don't plan to do all the exercises: there are too many.) Exercises have code letters to indicate their special features:

- R means a particularly good problem recommended for all readers;
- A means a question that has an answer in Appendix G;
- H means a particularly hard problem;
- P means a problem requiring actual programming in Prolog;
- E means an essay question;
- G means a good group project.

In addition to exercises, each chapter has a list of key terms you should know. Think of this list, at the end of the text for each chapter, as a set of "review questions."

The symbol * on a section of a chapter means optional reading. These sections are either significantly harder than the rest of the text or significantly far from the core material.

CONTENTS

8 REPRESENTING UNCERTAINTY IN RULE-BASED SYSTEMS *164*

9 SEARCH *191*