YOAV SHOHAM

Artificial Intelligence Techniques in Prolog

Artificial Intelligence Techniques in Prolog

Yoav Shoham

Morgan Kaufmann Publishers, Inc.
San Francisco, California

Sponsoring Editor: Michael B. Morgan
Production Manager: Yonie Overton
Assistant Editor: Douglas Sery
Copyeditor: Fran Taylor
Proofreader: Judy Bess
Cover Design: Studio Silicon
Printer: R.R. Donnelley & Sons

Editorial Offices:
Morgan Kaufmann Publishers, Inc.
340 Pine Street, Sixth Floor
San Francisco, CA 94104

Printed in the United States of America

98 97 96 95 94 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Shoham, Yoav.
Artificial intelligence techniques in Prolog / Yoav Shoham.
p. cm.
Includes bibliographical references and index.
ISBN 1-55860-319-0 (cloth)
ISBN 1-55860-167-8 (paper)
1. Artificial intelligence—Data processing. 2. Prolog (Computer program language) I. Title
Q336.S543 1994
006.3—dc20 92-42117
CIP

To Orit and Maia

Preface

The field of artificial intelligence (AI for short) spans a bewildering array of topics. Although usually thought of as part of computer science, AI overlaps with disciplines as diverse as philosophy, linguistics, psychology, electrical engineering, mechanical engineering, and neuroscience. It is also quite young as scientific fields go. One result of this breadth and youth is an atmosphere of creative excitement and pioneering. Indeed, given AI's relatively short history, the number of innovations that have emerged from within AI, and their effects on neighboring disciplines, are striking.

Another outcome of this diversity and dynamism has been a lack of uniformity of research issues and scientific methodology. Although in general this outcome has both positive and negative ramifications, its effect on AI *teaching* has been largely negative; all too often the effect has been to sacrifice either breadth or depth of coverage. Teachers are hardly at fault here; simply too much happens under the umbrella called AI for teachers to adequately cover it all in a one-semester (let alone one-quarter) course. Thus some courses emphasize the cognitive-science component of AI, some concentrate on knowledge representation, some on knowledge-based systems, some on reasoning techniques, and so on. By necessity, the broader the material covered in an introductory course, the shallower the coverage. The best balance for an introductory AI course is still a topic for debate.

This book is not a broad introduction to AI; the book's primary aim is to provide a crisp introduction to the well-established algorithmic techniques in the field. As a result, it is not particularly gentle, but instead plunges rather directly into the details of each technique. Most importantly, the book gives short shrift to conceptual issues, mentioning them briefly only by way of positioning the material within the AI landscape. Questions such as

"What is the nature of intelligence?"; "What does the Turing Test actually measure?"; and "Is symbol manipulation the best framework within which to model natural intelligence or to create an artificial one?" will remain fascinating and outside the scope of this book.

The techniques included in the book cover general areas such as search, rule-based systems, truth maintenance, constraint satisfaction, and uncertainty management, and specific application domains such as temporal reasoning, machine learning, and natural language. These areas are sufficiently diverse that I have had to omit much material. I hope that the following is nonetheless true of the selection:

- The material is self-contained in two ways. First, I include coverage of basic techniques, even those with which many readers are likely to be familiar (this is true especially of the search chapter). Second, I include (brief) summaries of required background material.
- The techniques discussed are completely demystified. Although I deliberately try to keep the presentation informal, the techniques are explained clearly; sufficient details are supplied to remove ambiguity, and details that are not essential to understanding the techniques are omitted. When desirable and possible, I present the techniques in stages, adding functionality or improving efficiency incrementally.
- The material is up-to-date and balanced. Since the material includes basic techniques as well as some more advanced ones, and, since the areas covered are quite diverse, the coverage of all areas is necessarily partial. Nonetheless, the most influential recent techniques in each area are included.

References for further reading, whether to achieve deeper theoretical understanding or to further explore the techniques discussed, are mentioned at the end of each chapter and appear in the bibliography at the end of the book. In addition, many of the exercises at the end of each chapter have been designed to explore issues which are not treated in the text.

Some readers might wonder why I insist on presenting programs, rather than simply explaining the algorithms in language-neutral terms. Indeed, in many places the programs are preceded by high-level pseudo code. However, it is not without reason that AI practitioners have developed a healthy skepticism of unimplemented ideas. Many of the techniques we will discuss

are quite intricate and messy, and, in the past, many reasonable-looking procedures turned out, upon being put to use, to have swept under the rug some of the most important details. Interpreters and compilers help keep one honest; if nothing else, our programs will expose the limitations of the procedures they implement.

In selecting Prolog as the implementation language, I also hope to dispel some misconceptions about the language. Prolog is a fun language, and students take a quick liking to it. This makes it a good choice for pedagogical reasons. For historical reasons, there are those in AI, especially in the United States, who have claimed that Prolog is unsuited for implementation of all but a narrow slice of AI techniques. As we shall see, this claim is quite false.

Prolog grew out of research in logic, and is the best-known representative of *logic programming* languages. I will nevertheless say little about logic in this book. This is particularly ironic, as much of my own research has been concerned with the application of logic in AI. However, perhaps precisely for this reason, I have too much respect for both Prolog and logic to be glib about the complex relationship between them. In this book I use Prolog as a flexible, efficient, and, yes, procedural language. Furthermore, in various places in the book, efficiency and purity were sacrificed for the sake of clarity. I believe that the utility and beauty of Prolog show nonetheless.

I have not included an introduction to Prolog. Excellent textbooks, such as Clocksin and Mellish's *Programming in Prolog* [7] and Sterling and Shapiro's *The Art of Prolog* [77], already exist for this purpose. A rough criterion for the requisite Prolog knowledge is familiarity with the material in Clocksin and Mellish's book. Chapter 1 elaborates on the required Prolog knowledge and introduces additional Prolog material that will be used in the book.

This book grew out of the course notes for a class I have been teaching at Stanford University, titled "AI techniques in Prolog." I have always started the class with a crash course in Prolog; I have found six 75-minute lectures quite adequate, although students are offered an additional laboratory section as an option. The balance of the course covers material in this book. No single course is likely to cover the entire corpus included here; the topics chosen will depend on the background and interests of the audience. I have tended to divide the time roughly as follows: search (2 lectures), meta-interpreters (1–2), forward chaining and production systems (1–2), truth maintenance (2), uncertainty (1), planning and temporal reasoning (2), learning (2), and natural language (1). This selection is appropriate

for students who have had one course in AI, or for those who have had none but are willing to compensate by studying on their own. If less is assumed on the part of the students, some of the advanced material must be omitted. Conversely, students with more experience may need to spend less time on some of the earlier chapters, for example those on search and forward chaining.

Acknowledgements

This book has been written over about four years, long enough a period to benefit from the feedback of a large number of people. I know that after the book is published it will dawn on me that I neglected to acknowledge the invaluable help of some dear friend; I apologize in advance.

Four research assistants helped tremendously. First and foremost, I thank Dominique Snyers. Dominique helped design the book outline, researched the strengths and weaknesses of existing books covering related material, and helped write some of the code. In particular, the natural language chapter would not have been written without Dominique.

The subsequent research assistants were (in chronological order) Anuchit Anuchitanukul, Avrami Tzur, and Robert Kennedy. They each provided crucial help in designing new algorithms or improving existing ones, implementing them, and debugging. For example, Anuchit came up with the meta-interpreter to handle 'cut,' Avrami wrote the first known implementation of Nilsson's RSTRIPS in the western world, and Robert simplified Allen's temporal constraint-satisfaction procedure. They each did much more, and have my deepest admiration and gratitude.

The following colleagues were very generous with their time, either filling in gaps in my knowledge or commenting on early drafts, or both: Eugene Charniak, Keith Clark, Tom Dean, Rina Dechter, Mark Drummond, Markus Fromherz, Herve Gallaire, Robert Goldman, Maria Gini, Steve Hanks, Pentii Hietala, Pekka Ketola, Apostolos Lerios, Jalal Maleki, David McAllester, Judea Pearl, and Udi Shapiro.

I'd be remiss if I did not single out Richard O'Keefe for special thanks. Richard has sent me what must amount to fifty pages of comments on earlier drafts. Most of his pointed suggestions were too good to ignore, and the result is a better if later book. Chapters 1-3 particularly benefitted from

Richard's comments. For example, in Chapter 1 some of the utility predicates (such as call / n) were supplied directly by Richard, and in Chapter 2 the minimax implementation is based on his suggestion.

I am indebted to a number of colleagues at Stanford. The Robotics Laboratory, where I have worked for the past five years, is a stimulating environment. In particular, this book has benefitted from continuous interaction with Jean-Claude Latombe and Nils Nilsson.

I have a lot for which to thank Mike Morgan from Morgan Kaufmann, who was engaged in this project from an early stage; his intelligent advice has been invaluable, and his informal style a real pleasure. I also thank Yonie Overton for a very friendly and astute production management.

Members of my research group, *knowbotics,* have been my primary source of intellectual challenge and satisfaction. Over the past few years they have included Ronen Brafman, Sylvie Cazalens, Kave Eshghi, Nita Goyal, Ronny Kohavi, James Kittock, Phillipe Lamarre, Fangzhen Lin, Eyal Mozes, Andrea Schaerf, Anton Schwartz, Grisha Schwarz, Moshe Tennenholtz, Becky Thomas, Mark Torrance, and Alvaro del Val; thank you all.

Software Availability

This book contains a substantial amount of Prolog code. The software is obtainable in one of the following ways:

- It may be retrieved through *anonymous FTP.*
- It may be ordered from the publisher.

The first service is free of charge; the second entails a charge to cover the publisher's costs. The sections below provide additional details about each option.

I regret that neither I nor the publisher will be able to provide software support, whether with regard to installing the software or to running it. However, I do welcome comments on the code and suggestions for improvements. Such comments should be sent only through electronic mail, addressed to `aitp@cs.stanford.edu`.

A word about quality control. All the code has been debugged and tested, but not at the level of commercial software. Accordingly, while every attempt has been made to provide correct code, no warranty is implied. Similarly, I have tried to make sure that the software being distributed matches the code given in the book, but some discrepancies are inevitable.

Using anonymous FTP

The *File Transfer Protocol (FTP)* is a standard protocol for transferring files over the Internet. In order to use it, you must be logged into a computer that is hooked into the net. If you do not have access to the net, then this method will be of no use to you. If you do have access to the Internet but have never used FTP, get help from someone who has.

The code is available for anonymous FTP from the computer `unix.sri.com`. It resides in the directory `pub/shoham`; the file `README` in that directory explains more about the various other files, and gives advice on what to copy.

A sample FTP session initiated by a user named `smith` at the Internet site `dept.univ.edu` might look as follows (user input in *slanted font*):

```
% ftp unix.sri.com (or ftp 128.18.10.3)
Name (unix.sri.com:smith): anonymous
331 Guest login ok, send indent as password
```

```
Password:  dept.univ.edu
230 Guest login ok, access restrictions apply
ftp> cd pub/shoham
250 CWD command successful.
ftp> ls
... (list of files)
ftp> prompt
Interactive mode off
ftp> mget*
...
ftp> bye
%
```

Anonymous FTP is a privilege, not a right. The site administrators at `unix.sri.com` have made the system available out of the spirit of sharing, but there are real costs associated with network connections, storage, and processing, all of which are needed to make this sharing possible. To avoid overloading the system, do not FTP between 7:00 a.m. and 6:00 p.m. local (pacific) time. If you are using this book for a class, do not FTP the code yourself; have the professor FTP it once and distribute code to the class. In general, use common sense and be considerate: none of us want to see sites close down because a few are abusing their privileges.

Ordering from the publisher

If you do not have access to the Internet, you may obtain the code for a modest fee from the publisher. You may contact the publisher either by mail or by phone:

Morgan Kaufmann Publishers, Inc.
340 Pine Street, Sixth Floor
San Francisco, CA 94104
415.392.2665
800.745.7323

When you do so, specify which of the following formats you desire:

Macintosh diskette
DOS 5.25 diskette
DOS 3.5 diskette
Unix TAR tape

Contents

Chapter 1

On Prolog

As explained in the preface, this book includes little material on Prolog itself. The present chapter is an exception; its purpose is threefold:

- to explain the required Prolog background;
- to provide a little additional Prolog material; and
- to define a small library of routine predicates that will be used later in the book.

There exist good introductory Prolog texts, including Clocksin and Mellish's tried-and-true *Programming in Prolog* [7], and Sterling and Shapiro's more advanced *The Art of Prolog* [77]. Among the truly advanced texts, O'Keefe's *The Craft of Prolog* [61] stands out. The material in this book presupposes a working knowledge of standard 'Edinburgh' Prolog. A rough criterion of the required background is familiarity with the material in Clocksin and Mellish's book. To further help the reader gauge his/her[1] preparedness, the next section provides a checklist of concepts and built-in predicates that the reader is expected to know.

The section following that introduces some additional Prolog material that a reader might have missed in previous exposure to Prolog. Some of the material, such as that on lists and all-solutions predicates, is likely to

[1] From here on I will use the generic masculine form, intending no bias.