ARTIFICIAL INTELLIGENCE

STRUCTURES AND STRATEGIES FOR COMPLEX PROBLEM SOLVING

SECOND EDITION

GEORGE F. LUGER • WILLIAM A. STUBBLEFIELD

ARTIFICIAL INTELLIGENCE

Structures and Strategies for Complex Problem Solving

Second Edition

George F. Luger

William A. Stubblefield

University of New Mexico, Albuquerque

The Benjamin/Cummings Publishing Company, Inc.

Redwood City, California • Menlo Park, California
Reading, Massachusetts • New York • Don Mills, Ontario
Wokingham, U.K. • Amsterdam • Bonn • Sydney
Singapore • Tokyo • Madrid • San Juan

Sponsoring Editor: Carter Shanklin
Production Coordinator: Megan Rundel
Copyeditor: Anna Huff
Cover Designer: Rudy Zehnter
Compositor: GTS Graphics

Cover and part opener artwork by Thomas Barrow.

Library of Congress Cataloging-in-Publication Data
Luger, George F.
Artificial intelligence: structures and strategies for complex problem solving
George F. Luger, William A. Stubblefield.—2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-8053-4780-1
1. Artificial intelligence. 2. Knowledge representation (Information theory) 3. Problem solving. 4. PROLOG (Computer program language) 5. LISP (Computer program language) I. Stubblefield, William A. II. Title.
Q335.L84 1992 92-26477
006.3—dc20 CIP

ISBN 0-8053-4780-1

3 4 5 6 7 8 9 10–DO–95 94

The Benjamin/Cummings Publishing Company, Inc.
390 Bridge Parkway
Redwood City, California 94065

ARTIFICIAL INTELLIGENCE

Structures and Strategies for Complex Problem Solving

Second Edition

For my wife, Kathleen, and our children Sarah, David, and Peter.
For my parents, George Fletcher Luger and Loretta Maloney Luger.

Si quid est in me ingenii, judices . . .

Cicero, Pro Archia Poeta

GFL

For my parents, William Frank Stubblefield and Maria Christina Stubblefield.
And for my wife, Merry Christina.

She is far more precious than jewels.
The heart of her husband trusts in her,
and he will have no lack of gain.

Proverbs 31:10–11

WAS

PREFACE

What we have to learn to do
we learn by doing . . .

—Aristotle, *Ethics*

Purpose

This book is an introduction to artificial intelligence and its many applications. Our goal in writing it is to place AI within the larger context of engineering, science, and philosophy. This includes not only computer science, but the full range of scientific efforts to understand the nature of intelligent activity.

There are three features of this book that reflect our own attitudes towards teaching artificial intelligence and distinguish our book from others.

First, we unify the diverse branches of AI through a detailed discussion of its theoretical foundations. Throughout its history, AI has exhibited two major methodological schools, often called the *neats* and the *scruffies.* The neats emphasize formal treatment of the theory behind intelligent programs including the analysis of search algorithms, the establishment of a theory of knowledge representation using logic or other well-defined formalisms, and the careful characterization of the soundness and completeness of inference strategies. The scruffies take a more application-oriented approach, beginning with a problem to be solved and finding, through intuition and experiment, the best techniques for solving that particular problem.

A major goal in writing this text is to unify these two approaches; indeed, we believe that these schools have much more in common than is usually admitted. Theoretical work in AI leads to an increasing awareness that intelligent behavior relies on domain-specific knowledge, knowledge that is often obtained by observation, experimentation and revision.

Conversely, the best AI implementations are usually built on the knowledge representation and problem-solving techniques developed in theoretical research.

To construct a unified approach to AI, we start with a well-marked separation between theory and practice by formally considering search in Part II and later implementing search algorithms in PROLOG (Chapters 6 and 13) and LISP (Chapters 7, 14 and 15). However, even the more formal chapters use realistic examples to illustrate important issues. For instance, much of our theoretical discussion of representation and search is illustrated through an extended example of an expert system for financial advice, our chapter on expert systems is anchored by a case study of the MYCIN program, and our machine learning algorithms, in Chapter 12, are supported by numerous implemented examples. The chapter on natural language concludes with a detailed discussion of the implementation of a natural language front end for a data base system.

The second feature of this text is the development, justification, and use of advanced representational formalisms and search techniques. This begins with the classic approach to representation and problem solving using predicate calculus, recursion-based graph search, and heuristics. We then extend this model to the knowledge-intensive approaches used for modern intelligent systems. Much of the emphasis of AI reaserch is on the development of languages which directly support specific knowledge structures such as class hierarchies, frames and objects, and tools for describing the semantics of natural languages. We discuss the importance of structuring information and control with semantic nets, conceptual graphs, frames, inheritance systems, and other representational techniques.

Part V, and especially Chapter 15, introduces object-oriented programming and hybrid expert system design. We present three implementations of object-oriented representations, two in LISP, the first with OOPS, our own object system, the second with CLOS, the Common Lisp Object System; and one in PROLOG. Our presentation gives the reader the opportunity to design and build several simple object-based and hybrid applications.

Finally, we show how the algorithms and data structures of AI programming can be built using either LISP or PROLOG. We have covered both languages because they represent two very different but equally important AI programming paradigms: logic programming in PROLOG and functional programming in LISP. We feel that the skills of the programmer depend on a knowledge of all available tools, and each language has its representational strengths and search-based advantages. Indeed, in many modern programming environments the strengths of both languages are available.

The third feature of this book is that we place artificial intelligence within the context of empirical science. We do this in several ways. First, by introducing artificial intelligence in Chapter 1 as part of the long tradition of a science of intelligent systems. This context shows AI, not as some strange abberation from the scientific tradition, but as part of a general quest for knowledge about and understanding of intelligence. Furthermore, our AI programming tools, along with an exploratory programming methodology in Chapter 8, are ideal for exploring our environment. Our tools give us a medium for both understanding and questions. We come to appreciate and know phenomena constructively, that is, by progressive approximation.

Thus we see each design and program as an experiment with nature: we propose a representation, we generate a search algorithm, and then we question the adequacy of our characterization to account for part of the phenomenon of intelligence. And the natural world gives a response to our query. Our experiment can be deconstructed, revised, extended, and run again. Our model can be refined, our understanding extended. We consider artificial intelligence as empirical enquiry in detail in Chapter 16.

New in This Edition

Machine learning, currently a very important research topic in the AI community, is the first major addition to this edition. The ability to learn must be part of any system that would claim to possess general intelligence. Learning is also an important component of practical AI applications such as expert systems. The learning models presented in Chapter 12 include explicitly represented knowledge where information is encoded in a symbol system and learning takes place through algorithmic manipulation, or search, of these structures. We also present the sub-symbolic approach to learning. In a neural net, for instance, information is implicit in the organization and weights on a set of connected processors, and learning is a rearrangement and modification of the overall structure of the system. We also introduce genetic algorithms where learning is cast as an evolutionary and adaptive process. We compare and contrast the directions and results of each approach to machine learning and develop learning algorithms first in pseudocode and then in LISP and/or PROLOG.

Our second major addition is object-oriented design with CLOS, the Common Lisp Object System. CLOS is now the ANSI standard for object based design in Common LISP. We show the power of CLOS for building object systems, and in Chapter 15 we develop a simulation for a heating system that can be easily extended by the reader.

We have also extended the presentation of the other applications of the book. After presenting the knowledge representations for semantic meaning of natural language in Chapter 9, we discuss the integration of syntax and semantics in the implementation programs of Chapter 10. We design and build context-free, context-sensitive, and a recursive descent semantic net parser in Chapter 13. Other application areas explored in the text include planning (Chapter 5) and automated reasoning (Chapter 11). In addition to these traditional AI application areas, we briefly examine the role of AI in understanding human intelligence through a treatment of cognitive science (Chapter 16).

The Contents

Chapter 1 introduces artificial intelligence, beginning with a brief history of attempts to understand mind and intelligence in philosophy, psychology, and other areas of research. In a very important sense, AI is an old science, tracing its roots back at least to Aristotle. An appreciation of this background is essential for an understanding of the issues addressed in modern research. We also present an overview of some of the important application areas in AI. Our goal in Chapter 1 is to provide both background and a motivation for the theory and applications that follow.

Chapters 2, 3, 4, and 5 (Part II) introduce the research tools for AI problem solving. These include the predicate calculus to describe the essential features of a domain (Chapter 2), search to reason about these descriptions (Chapter 3) and the algorithms and data structures used to implement search. In Chapters 4 and 5, we discuss the essential role of heuristics in focusing and constraining search based problem solving. We also present a number of architectures, including the blackboard and production system, for building these search algorithms.

Part III presents AI languages. These languages are first compared to each other and to traditional programming languages to give an appreciation of the AI approach to problem solving. Chapter 6 covers PROLOG, and Chapter 7, LISP. We demonstrate these languages as tools for AI problem solving by building on the search and representation techniques of the earlier chapters, including breadth-first, depth-first and best-first search algorithms. We implement these search techniques in a problem-independent fashion so they may later be extended to form shells for search in rule based expert systems, semantic net and frame systems, as well as in other applications.

Chapters 8, 9, 10, 11, and 12 make up Part IV of the text: representations for knowledge-based systems. In Chapter 8 we present the rule-based expert system. This model for problem solving is a natural evolution of the material in the first five chapters: using a production system of predicate calculus expressions to orchestrate a graph search. Both data-driven and goal-driven search are presented in this context and the role of heuristics is demonstrated. We also discuss the unique problems encountered in applying expert systems. Examples are taken from the Stanford University research, including presentations of MYCIN and Teiresias.

Chapters 9 and 10 present AI techniques for modeling semantic meaning, with a particular focus on natural langauge understanding. We begin with a discussion of semantic networks and extend this model to include conceptual dependency theory, conceptual graphs, frames, and scripts. Class hierarchies and inheritance are important representation tools; we discuss both the benefits and difficulties of implementing inheritance systems for realistic taxonomies. This material is strengthened by an in-depth examination of a particular formalism, conceptual graphs. This discussion emphasizes the epistemological issues involved in representing knowledge and shows how these issues are addressed in a modern representation language. In Chapter 10, we also show how conceptual graphs can be used to implement a natural language data base front end.

Theorem proving, often referred to as automated reasoning, is one of the oldest areas of AI research. In Chapter 11, we discuss the first programs in this area, including the Logic Theorist and the General Problem Solver. The primary focus of the chapter is binary resolution proof procedures, especially resolution refutations. More advanced inferencing with hyper-resolution and paramodulation is also discussed.

In Chapter 12 we present a detailed look at algorithms for machine learning, a fruitful area of research spawning a number of different problems and solution approaches. The learning algorithms vary in their goals, the training data considered, the learning strategies, and the knowledge representations they employ. The symbol-based learning includes induction, concept learning, version space search, and ID3. The role of inductive bias is considered, as well as generalizations from patterns of data, and the effective use of knowledge to learn from a single example with explanation-based learning. Category learning, or conceptual clustering, is presented with unsupervised learning. We also present subsymbolic learning with genetic algorithms and neural nets. We take care to compare and contrast these diverse approaches to machine learning.

Part V, Chapters 13, 14, and 15, presents advanced AI programming in PROLOG and LISP. Here we discuss techniques for implementing the knowledge representation and problem-solving algorithms of Part IV. A major feature of these chapters is the implementation of expert system shells in both LISP and PROLOG. These shells include certainty measures, user queries, and complete explanation facilities. They illustrate the major architectural features of expert system shells and provide a useful tool for building such systems. We also implement the algorithms from Chapter 12 in machine learning, with several algorithms presented. Finally, we discuss the implementation of inheritance in frame and network representations. All three chapters discuss the formal underpinning of each language: resolution refutation systems for PROLOG and functional programming for LISP.

In Chapter 15, we discuss frame- or object-based design, showing how object-oriented techniques are used both for organizing traditional programs and for developing knowledge-based systems. We then give examples of the techniques used to implement object-oriented programming in both LISP and PROLOG. Finally, the chapter discusses the integration of rule- and frame-based approaches in hybrid knowledge engineering environments.

The final chapter, 16, serves as an epilogue for the text. It introduces the discipline of cognitive science, addresses contemporary challenges to AI, discusses AI's current limitations, and examines what we feel is its exciting future.

Using This Book

Artificial intelligence is a big field; consequently, this is a big book. Although it would require more than a single semester to cover all of the material in the text, we have designed it so that a number of paths may be taken through the material. By selecting subsets of the material, we have used this text for single semester and full year (two semester) courses.

Benjamin/Cummings (390 Bridge Parkway, Redwood City CA 94065) will provide an instructor's manual for this text. It includes suggestions for teaching topics, selected figures and formatted algorithms that can be used for making view graphs, and a selection of worked out exercises. The instructor's manual also suggests some paths through the book, depending on time allowed and topics chosen.

We assume that most students will have had introductory courses in discrete mathematics, including predicate calculus and graph theory. If this is not true the instructor should spend more time on these concepts in the sections at the beginning of the text (2.1, 3.1). We also assume that students have had courses in data structures including trees, graphs, and recursion-based search using stacks, queues, and priority queues. If they have not, they should spend more time on the beginning sections of Chapters 3, 4, and 5.

The algorithms are described using a Pascal-like pseudo code. This notation uses the control structures of Pascal along with English descriptions of the tests and operations. We have added two useful constructs to the Pascal control structures. The first is a modified case statement that, rather than comparing the value of a variable with constant case labels, as in standard Pascal, lets each item be labeled with an arbitrary boolean test. The case evaluates these tests in order until one of them is true and then performs the associated action; all other actions are ignored. Those familiar with LISP will note that this has the same semantics as the LISP cond statement.

The other addition to the language is a return statement which takes one argument and can appear anywhere within a procedure or function. When the return is encountered, it causes the program to immediately exit the function, returning its argument as a result. Other than these modifications we used Pascal structure, with a reliance on the English descriptions, to make the algorithms clear.

Supplemental material available via Internet

The PROLOG and LISP code in the book as well as a public domain C-PROLOG interpreter are available via FTP. To retrieve them: ftp bc.aw.com and log in as "anonymous", using your e-mail address as the password. Change directories by typing: cd bc/luger. View the "readme" file (get README) for current FTP status. Filenames are also available using the UNIX "ls", or the DOS "dir" command. Using FTP and de-archiving files can get complicated. Instructions vary for Macintosh, DOS, or UNIX files. Consult your local UNIX guide if you have questions.

Acknowledgments

First we would like to thank our reviewers, whose comments often suggested important additions of material and focus. These include Dennis Bahler, Skona Brittain, John Donald, Sarah Douglas, Ray Mooney, Bruce Porter, Jude Shavlik, Carl Stern, Marco Valtorta, and Bob Veroff. Carl Stern, a colleague at the University of New Mexico, has been especially helpful in the technical editing of this book. Mike Delleney, Jim Skinner, Steve Verzi, and numerous others participated in using and refining the code. We thank especially a decade of students who used this text in its preliminary stages and its first edition for their help in expanding its horizons, as well as in removing its typos and bugs.

We wish to thank Thomas Barrow, internationally recognized artist and University of New Mexico Professor of Art, who created the six photograms especially for this book after reading an early draft of the manuscript.

We would also like to thank Vivian McDougal, Lisa Moller, Megan Rundel, and Mary Tudor at Benjamin/Cummings for their work on the second edition of this text.

In a number of places, we have used figures or quotes from the work of other authors. We would like to thank the authors and publishers for their permission to use this material. These contributions are listed at the end of the text.

Artificial intelligence is an exciting and oftentimes rewarding discipline; may you enjoy your study as you come to appreciate its power and challenges.

George F. Luger
William A. Stubblefield
1 July 1992

ARTIFICIAL INTELLIGENCE

Structures and Strategies for Complex Problem Solving

Second Edition

TABLE OF CONTENTS