Principles of Artificial Intelligence

Nils J. Nilsson

PRINCIPLES OF ARTIFICIAL INTELLIGENCE

This page intentionally left blank

Principles of Artificial Intelligence

NILS J. NILSSON
Stanford University

MORGAN KAUFMANN
PUBLISHERS, INC.

Library of Congress Cataloging-in-Publication Data

Nilsson, Nils J., 1933–
Principles of artificial intelligence.

Reprint. Originally published: Palo Alto, Calif. :
Tioga Pub. Co., © 1980.
Bibliography: p.
Includes indexes.
1. Artificial intelligence. I. Title.
Q335.N515 1986 006.3 86-2815
ISBN 0-934613-10-9

ISBN 0-934613-10-9
(Previously published by Tioga Publishing Co. under ISBN 0-935382-01-1)
FG-DO

for Kristen and Lars

This page intentionally left blank

TABLE OF CONTENTS

This page intentionally left blank

PREFACE

Previous treatments of Artificial Intelligence (AI) divide the subject into its major areas of application, namely, natural language processing, automatic programming, robotics, machine vision, automatic theorem proving, intelligent data retrieval systems, etc. The major difficulty with this approach is that these application areas are now so extensive, that each could, at best, be only superficially treated in a book of this length. Instead, I have attempted here to describe fundamental AI ideas that underlie many of these applications. My organization of these ideas is not, then, based on the subject matter of their application, but is, instead, based on general computational concepts involving the kinds of data structures used, the types of operations performed on these data structures, and the properties of control strategies used by AI systems. I stress, in particular, the important roles played in AI by generalized production systems and the predicate calculus.

The notes on which the book is based evolved in courses and seminars at Stanford University and at the University of Massachusetts at Amherst. Although certain topics treated in my previous book, *Problem-solving Methods in Artificial Intelligence,* are covered here as well, this book contains many additional topics such as rule-based systems, robot problem-solving systems, and structured-object representations.

One of the goals of this book is to fill a gap between theory and practice. AI theoreticians have little difficulty in communicating with each other; this book is not intended to contribute to that communication. Neither is the book a handbook of current AI programming technology; other sources are available for that purpose. As it stands, the book could be supplemented either by more theoretical treatments of certain subjects, for AI theory courses, or by project and laboratory sessions, for more practically oriented courses.

The book is designed as a text for a senior or first-year graduate course in AI. It is assumed that the reader has a good background in the fundamentals of computer science; knowledge of a list-processing language, such as LISP, would be helpful. A course organized around this book could comfortably occupy a full semester. If separate practical or

theoretical material is added, the time required might be an entire year. A one-quarter course would be somewhat hurried unless some material (perhaps parts of chapter 6 and chapter 8) is omitted.

The exercises at the end of each chapter are designed to be thought-provoking. Some expand on subjects briefly mentioned in the text. Instructors may find it useful to use selected exercises as a basis for class discussion. Pertinent references are briefly discussed at the end of every chapter. These citations should provide the interested student with adequate entry points to much of the most important literature in the field.

I look forward someday to revising this book—to correct its inevitable errors, and to add new results and points of view. Toward that end, I solicit correspondence from readers.

Nils J. Nilsson

ACKNOWLEDGEMENTS

Several organizations supported and encouraged the research, teaching, and discussions that led to this book. The Information Systems Program, Marvin Denicoff, Director, of the Office of Naval Research, provided research support under contract no. N00014-77-C-0222 with SRI International. During the academic year 1976-77, I was a part-time visiting professor in the Computer Science Department at Stanford University. From September 1977 to January 1978, I spent the Winter Semester at the Computer and Information Sciences Department of the University of Massachusetts at Amherst. The students and faculty of these departments were immensely helpful in the development of this book.

I want to give special thanks to my home organization, SRI International, for the use of its facilities and for its liberal attitude toward book-writing. I also want to thank all my friends and colleagues in the Artificial Intelligence Center at SRI. One could not find a more dynamic, intellectually stimulating, and constructively critical setting in which to work and write.

Though this book carries the name of a single author, it has been influenced by several people. It is a pleasure to thank here everyone who helped guide me toward a better presentation. Some of those who provided particularly detailed and extensive suggestions are: Doug Appelt, Michael Arbib, Wolfgang Bibel, Woody Bledsoe, John Brown, Lew Creary, Randy Davis, Jon Doyle, Ed Feigenbaum, Richard Fikes, Northrup Fowler, Peter Friedland, Anne Gardner, David Gelperin, Peter Hart, Pat Hayes, Gary Hendrix, Doug Lenat, Vic Lesser, John Lowrance, Jack Minker, Tom Mitchell, Bob Moore, Allen Newell, Earl Sacerdoti, Len Schubert, Herb Simon, Reid Smith, Elliot Soloway, Mark Stefik, Mabry Tyson, and Richard Waldinger.

I also want to thank Robin Roy, Judy Fetler, and Georgia Navarro, for patient and accurate typing; Sally Seitz for heroic insertion of typesetting instructions into the manuscript; and Helen Tognetti for creative copy-editing.

Most importantly, my efforts would not have been equal to this task had they not been generously supported, encouraged, and understood by my wife, Karen.

This page intentionally left blank

CREDITS

The manuscript for this book was prepared on a Digital Equipment Corporation KL-10 computer at SRI International. The computer manuscript file was processed for automatic photo-typesetting by W. A. Barrett's TYPET system on a Hewlett-Packard 3000 computer. The main typeface is Times Roman.

Book design: Ian Bastelier
Cover design: Andrea Hendrick
Illustrations: Marla Masterson
Typesetting: Typothetae, Palo Alto, CA
Page makeup: Vera Allen Composition, Castro Valley, CA
Printing and binding: R. R. Donnelley and Sons Company

This page intentionally left blank

PROLOGUE

Many human mental activities such as writing computer programs, doing mathematics, engaging in commonsense reasoning, understanding language, and even driving an automobile are said to demand "intelligence." Over the past few decades, several computer systems have been built that can perform tasks such as these. Specifically, there are computer systems that can diagnose diseases, plan the synthesis of complex organic chemical compounds, solve differential equations in symbolic form, analyze electronic circuits, understand limited amounts of human speech and natural language text, or write small computer programs to meet formal specifications. We might say that such systems possess some degree of *artificial intelligence.*

Most of the work on building these kinds of systems has taken place in the field called *Artificial Intelligence* (*AI*). This work has had largely an empirical and engineering orientation. Drawing from a loosely structured but growing body of computational techniques, AI systems are developed, undergo experimentation, and are improved. This process has produced and refined several general AI principles of wide applicability.

This book is about some of the more important, core AI ideas. We concentrate on those that find application in several different problem areas. In order to emphasize their generality, we explain these principles abstractly rather than discuss them in the context of specific applications, such as automatic programming or natural language processing. We illustrate their use with several small examples but omit detailed case studies of large-scale applications. (To treat these applications in detail would each certainly require a separate book.) An abstract understanding of the basic ideas should facilitate understanding specific AI systems (including strengths and weaknesses) and should also prove a sound basis for designing new systems.

AI has also embraced the larger scientific goal of constructing an information-processing theory of intelligence. If such a *science of intelligence* could be developed, it could guide the design of intelligent machines as well as explicate intelligent behavior as it occurs in humans and other animals. Since the development of such a general theory is still very much a goal, rather than an accomplishment of AI, we limit our attention here to those principles that are relevant to the engineering goal of building intelligent machines. Even with this more limited outlook, our discussion of AI ideas might well be of interest to cognitive psychologists and others attempting to understand natural intelligence.

As we have already mentioned, AI methods and techniques have been applied in several different problem areas. To help motivate our subsequent discussions, we next describe some of these applications.

0.1. SOME APPLICATIONS OF ARTIFICIAL INTELLIGENCE

0.1.1. NATURAL LANGUAGE PROCESSING

When humans communicate with each other using language, they employ, almost effortlessly, extremely complex and still little understood processes. It has been very difficult to develop computer systems capable of generating and "understanding" even fragments of a natural language, such as English. One source of the difficulty is that language has evolved as a communication medium between *intelligent* beings. Its primary use is for transmitting a bit of "mental structure" from one brain to another under circumstances in which each brain possesses large, highly similar, surrounding mental structures that serve as a common context. Furthermore, part of these similar, contextual mental structures allows each participant to know that the other also possesses this common structure and that the other can and will perform certain processes using it during communication "acts." The evolution of language use has apparently exploited the opportunity for participants to use their considerable computational resources and shared knowledge to generate and understand highly condensed and streamlined messages: A word to the wise from the wise is sufficient. Thus generating and understanding language is an encoding and decoding problem of fantastic complexity.

A computer system capable of understanding a message in natural language would seem, then, to require (no less than would a human) both the contextual knowledge and the processes for making the inferences (from this contextual knowledge and from the message) assumed by the message generator. Some progress has been made toward computer systems of this sort, for understanding spoken and written fragments of language. Fundamental to the development of such systems are certain AI ideas about structures for representing contextual knowledge and certain techniques for making inferences from that knowledge. Although we do not treat the language-processing problem as such in this book, we do describe some important methods for knowledge representation and processing that do find application in language-processing systems.

0.1.2. INTELLIGENT RETRIEVAL FROM DATABASES

Database systems are computer systems that store a large body of facts about some subject in such a way that they can be used to answer users' questions about that subject. To take a specific example, suppose the facts are the personnel records of a large corporation. Example items in such a database might be representations for such facts as "Joe Smith works in the Purchasing Department," "Joe Smith was hired on October 8, 1976," "The Purchasing Department has 17 employees," "John Jones is the manager of the Purchasing Department," etc.

The design of database systems is an active subspecialty of computer science, and many techniques have been developed to enable the efficient representation, storage, and retrieval of large numbers of facts. From our point of view, the subject becomes interesting when we want to retrieve answers that require deductive reasoning with the facts in the database.

There are several problems that confront the designer of such an *intelligent* information retrieval system. First, there is the immense problem of building a system that can understand queries stated in a natural language like English. Second, even if the language-understanding problem is dodged by specifying some formal, machine-understandable query language, the problem remains of how to deduce answers from stored facts. Third, understanding the query and deducing an answer may require knowledge beyond that explicitly represented in the subject domain database. Common knowledge (typically omitted in the subject domain database) is often required. For example, from the personnel facts mentioned above, an intelligent system ought to be able

to deduce the answer "John Jones" to the query "Who is Joe Smith's boss?" Such a system would have to know somehow that the manager of a department is the *boss* of the people who work in that department. How common knowledge should be represented and used is one of the system design problems that invites the methods of Artificial Intelligence.

0.1.3. EXPERT CONSULTING SYSTEMS

AI methods have also been employed in the development of automatic consulting systems. These systems provide human users with expert conclusions about specialized subject areas. Automatic consulting systems have been built that can diagnose diseases, evaluate potential ore deposits, suggest structures for complex organic chemicals, and even provide advice about how to use other computer systems.

A key problem in the development of expert consulting systems is how to represent and use the knowledge that human experts in these subjects obviously possess and use. This problem is made more difficult by the fact that the expert knowledge in many important fields is often imprecise, uncertain, or anecdotal (though human experts use such knowledge to arrive at useful conclusions).

Many expert consulting systems employ the AI technique of *rule-based deduction*. In such systems, expert knowledge is represented as a large set of simple rules, and these rules are used to guide the dialogue between the system and the user and to deduce conclusions. Rule-based deduction is one of the major topics of this book.

0.1.4. THEOREM PROVING

Finding a proof (or disproof) for a conjectured theorem in mathematics can certainly be regarded as an intellectual task. Not only does it require the ability to make deductions from hypotheses but demands intuitive skills such as guessing about which lemmas should be proved first in order to help prove the main theorem. A skilled mathematician uses what he might call judgment (based on a large amount of specialized knowledge) to guess accurately about which previously proven theorems in a subject area will be useful in the present proof and to break his main

problem down into subproblems to work on independently. Several automatic theorem proving programs have been developed that possess some of these same skills to a limited degree.

The study of theorem proving has been significant in the development of AI methods. The formalization of the deductive process using the language of predicate logic, for example, helps us to understand more clearly some of the components of reasoning. Many informal tasks, including medical diagnosis and information retrieval, can be formalized as theorem-proving problems. For these reasons, theorem proving is an extremely important topic in the study of AI methods.

0.1.5. ROBOTICS

The problem of controlling the physical actions of a mobile robot might not seem to require much intelligence. Even small children are able to navigate successfully through their environment and to manipulate items, such as light switches, toy blocks, eating utensils, etc. However these same tasks, performed almost unconsciously by humans, performed by a machine require many of the same abilities used in solving more intellectually demanding problems.

Research on robots or robotics has helped to develop many AI ideas. It has led to several techniques for modeling *states of the world* and for describing the process of change from one world state to another. It has led to a better understanding of how to generate *plans* for action sequences and how to monitor the execution of these plans. Complex robot control problems have forced us to develop methods for planning at high levels of abstraction, ignoring details, and then planning at lower and lower levels, where details become important. We have frequent occasion in this book to use examples of robot problem solving to illustrate important ideas.

0.1.6. AUTOMATIC PROGRAMMING

The task of writing a computer program is related both to theorem proving and to robotics. Much of the basic research in automatic programming, theorem proving, and robot problem solving overlaps. In a sense, existing compilers already do "automatic programming." They take in a complete source code specification of what a program is to

accomplish, and they write an object code program to do it. What we mean here by automatic programming might be described as a "super-compiler," or a program that could take in a very high-level description of what the program is to accomplish and produce a program. The high-level description might be a precise statement in a formal language, such as the predicate calculus, or it might be a loose description, say, in English, that would require further dialogue between the system and the user in order to resolve ambiguities.

The task of automatically writing a program to achieve a stated result is closely related to the task of proving that a given program achieves a stated result. The latter is called *program verification*. Many automatic programming systems produce a verification of the output program as an added benefit.

One of the important contributions of research in automatic programming has been the notion of *debugging* as a problem-solving strategy. It has been found that it is often much more efficient to produce an inexpensive, errorful solution to a programming or robot control problem and then modify it (to make it work correctly), than to insist on a first solution completely free of defects.

0.1.7. COMBINATORIAL AND SCHEDULING PROBLEMS

An interesting class of problems is concerned with specifying optimal schedules or combinations. Many of these problems can be attacked by the methods discussed in this book. A classical example is the *traveling salesman's problem*, where the problem is to find a minimum distance tour, starting at one of several cities, visiting each city precisely once, and returning to the starting city. The problem generalizes to one of finding a minimum cost path over the edges of a graph containing n nodes such that the path visits each of the n nodes precisely once.

Many puzzles have this same general character. Another example is the *8-queens problem*, where the problem is to place eight queens on a standard chessboard in such a way that no queen can capture any of the others; that is, there can be no more than one queen in any row, column or diagonal. In most problems of this type, the domain of possible combinations or sequences from which to choose an answer is very large. Routine attempts at solving these types of problems soon generate a *combinatorial explosion* of possibilities that exhaust even the capacities of large computers.

Several of these problems (including the traveling salesman problem) are members of a class that computational theorists call *NP-complete.* Computational theorists rank the difficulty of various problems on how the worst case for the time taken (or number of *steps* taken) using the theoretically best method grows with some measure of the problem size. (For example, the number of cities would be a measure of the size of a traveling salesman problem.) Thus, problem difficulty may grow linearly, polynomially, or exponentially, for example, with problem size.

The time taken by the best methods currently known for solving NP-complete problems grows exponentially with problem size. It is not yet known whether faster methods (involving only polynomial time, say) exist, but it has been proven that if a faster method exists for one of the NP-complete problems, then this method can be converted to similarly faster methods for all the rest of the NP-complete problems. In the meantime, we must make do with exponential-time methods.

AI researchers have worked on methods for solving several types of combinatorial problems. Their efforts have been directed at making the time-versus-problem-size curve grow as slowly as possible, even when it must grow exponentially. Several methods have been developed for delaying and moderating the inevitable combinatorial explosion. Again, knowledge about the problem domain is the key to more efficient solution methods. Many of the methods developed to deal with combinatorial problems are also useful on other, less combinatorially severe problems.

0.1.8. PERCEPTION PROBLEMS

Attempts have been made to fit computer systems with television inputs to enable them to "see" their surroundings or to fit them with microphone inputs to enable them to "hear" speaking voices. From these experiments, it has been learned that useful processing of complex input data requires "understanding" and that understanding requires a large base of knowledge about the things being perceived.

The process of perception studied in Artificial Intelligence usually involves a set of operations. A visual scene, say, is encoded by sensors and represented as a matrix of intensity values. These are processed by detectors that search for primitive picture components such as line segments, simple curves, corners, etc. These, in turn, are processed to

infer information about the three-dimensional character of the scene in terms of its surfaces and shapes. The ultimate goal is to represent the scene by some appropriate model. This model might consist of a high-level description such as "A hill with a tree on top with cattle grazing."

The point of the whole perception process is to produce a condensed representation to substitute for the unmanageably immense, raw input data. Obviously, the nature and quality of the final representation depend on the goals of the perceiving system. If colors are important, they must be noticed; if spatial relationships and measurements are important, they must be judged accurately. Different systems have different goals, but all must reduce the tremendous amount of sensory data at the input to a manageable and meaningful description.

The main difficulty in perceiving a scene is the enormous number of possible candidate descriptions in which the system might be interested. If it were not for this fact, one could conceivably build a number of detectors to decide the *category* of a scene. The scene's category could then serve as its description. For example, perhaps a detector could be built that could test a scene to see if it belonged to the category "A hill with a tree on top with cattle grazing." But why should this detector be selected instead of the countless others that might have been used?

The strategy of making hypotheses about various levels of description and then testing these hypotheses seems to offer an approach to this problem. Systems have been constructed that process suitable representations of a scene to develop hypotheses about the components of a description. These hypotheses are then tested by detectors that are specialized to the component descriptions. The outcomes of these tests, in turn, are used to develop better hypotheses, etc.

This *hypothesize-and-test* paradigm is applied at many levels of the perception process. Several aligned segments suggest a straight line; a line detector can be employed to test it. Adjacent rectangles suggest the faces of a solid prismatic object; an object detector can be employed to test it.

The process of hypothesis formation requires a large amount of knowledge about the expected scenes. Some AI researchers have suggested that this knowledge be organized in special structures called *frames* or *schemas*. For example, when a robot enters a room through a