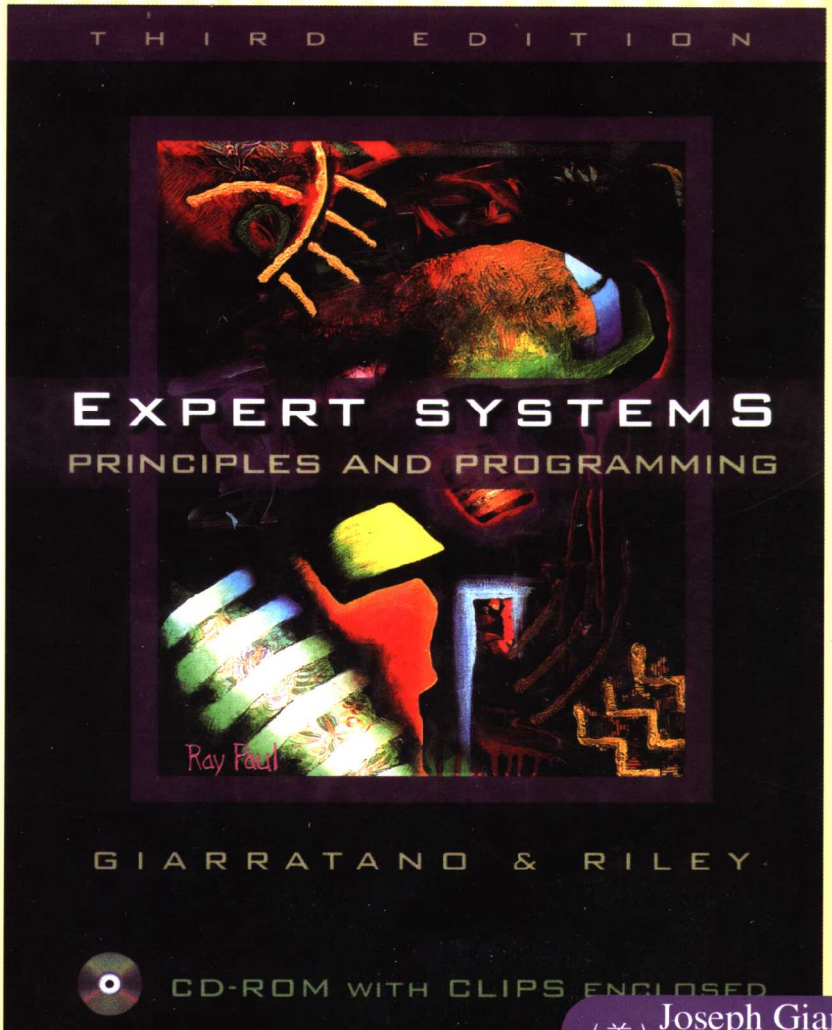
经 典 原 版 书 库

专家系统原理与编程

（英文版 · 第3版）



附赠 CD-ROM

（美）Joseph Giarratano
Gary Riley 著

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE
THOMSON

经 典 原 版 书 库

专家系统原理与编程

（英文版 · 第3版）

Expert Systems Principles and Programming

(Third Edition)

(美) Joseph Giarratano
Gary Riley 著

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

本书版权登记号：图字：01-2001-4884

图书在版编目（CIP）数据

专家系统原理与编程：第3版/（美）贾拉坦诺（Giarratona, J.）等著. - 北京：机械工业出版社，2002.8

（经典原版书库）

书名原文：Expert Systems Principles and Programming, Third Edition

ISBN 7-111-10844-2

Ⅰ. 专… Ⅱ. 贾… Ⅲ. ①专家系统 - 理论 - 英文 ②专家系统 - 程序设计 - 英文 Ⅳ. TP182

中国版本图书馆CIP数据核字（2002）第063182号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：华章

北京忠信诚印刷厂印刷 · 新华书店北京发行所发行

2002 年8月第1版第1次印刷

787mm × 1092mm1/16 · 38.25印张

印数：0 001-3 000册

定价：59.00元 (附光盘)

Preface

Expert systems have experienced tremendous growth and popularity since their commercial introduction in the early 1980s. Today, expert systems are used in business, science, engineering, manufacturing, and many other fields.

This book is meant to educate students about expert systems theory and programming. The material is written at the upper-division/graduate level suitable for majors in computer science, computer information systems, engineering, and other fields who are interested in expert systems. New terminology is shown in boldface and immediately explained. Numerous examples and references help clarify the meaning of the text and provide guidance for supplementary reading.

Expert Systems: Principles and Programming is divided into two parts. Chapters 1–6 cover the theory behind expert systems and how they fit into the scope of computer science; that is the logic, probability, data structures, AI, and other topics that form the theory of expert systems. Chapters 7–12 cover programming in expert systems. While a previous course in artificial intelligence (AI) is helpful, this book provides a self-contained introduction to AI topics that are appropriate for expert systems.

We have tried to explain the theory behind expert systems so that the student may make an informed decision regarding the appropriate use of expert system technology. The important point we emphasize is that like any other tool, expert systems have both advantages and disadvantages. The theory also explains how expert systems relate to other programming methods, such as conventional programming. Another reason for discussing theory is to prepare the student to read current research papers in expert systems. Because expert systems draw from so many diverse fields, it is difficult for a beginner to read papers without some grounding in theory.

The second part of this book is an introduction to the CLIPS expert system tool. This part is a practical introduction to expert system programming that serves to reinforce and clarify the theoretical concepts developed in the first part. As with the theory part of the book, the programming part can be understood by students with some programming experience in a high-level language. Students learn the practical problems associated with expert system development using CLIPS, a modern, powerful expert system tool developed by NASA at the Johnson Space Center. Today, CLIPS is used for real-world projects in government, business, and industry. It is available on virtually every type of computer including IBM-PC and clones, VAX, HP, Sun, Macintosh, and Cray. Code developed in CLIPS runs very fast and is very portable. The CD-ROM included with this book contains CLIPS executables for MS-DOS, Windows 3.1/Windows 95, and MacOS; the *CLIPS Reference Manual* and *CLIPS User's Guide*; and the C source code for CLIPS.

Some expert systems courses include a term project. A project is an excellent way to develop skills in expert systems. Students usually complete small expert systems of 50 to 150 rules in a semester project of their choice. A few projects that have been done by students using CLIPS include automobile diagnosis, taxi scheduling, personnel scheduling, computer network management, weather forecasting, stock market prediction, consumer buying advice, and diet advice.

The suggested plan for a one-semester course is as follows.

1. Cover Chapter 1 to provide a quick introduction to expert systems. In particular, assign Problems 1, 2, and 3.

2. Cover Chapters 7–10 to introduce the syntax of CLIPS. It is helpful for students to recode Problem 2 of Chapter 1 to contrast the expert system approach with the language they originally used in Chapter 1. This contrast is very useful in pointing out the differences between a rule-based language like CLIPS and LISP, PROLOG, or whatever the original language used for Problem 2.

3. Chapters 11 and 12 may be covered for a stronger programming emphasis in the course. Chapter 11, on efficiency in rule-based languages, is not essential to an introductory course. It may be discussed if there is interest in building large expert systems or if efficiency is important. Chapter 12 presents a collection of topics that illustrate interesting problems solved using CLIPS.

Alternatively, after Chapter 10 the instructor may return to the theory section. If students have strong backgrounds in logic and PROLOG, most of Chapters 2 and 3 may be skipped. Students who have had a LISP-based introductory AI course or none at all will benefit from Chapters 2 and 3 if a strong emphasis on logic and the fundamental theory of expert systems is desired. If students have strong backgrounds in probability and statistics, the material in Chapter 4 up to Section 4.11 can be skipped.

4. Sections 4.11 to the end of Chapter 4 and Chapter 5 discuss advanced topics in expert systems concerning methods of dealing with uncertainty. These topics include probabilistic inference, certainty factors, Dempster-Shafer theory, and fuzzy theory. Students will gain an understanding of these methods in sufficient detail so they can read current papers in the field and start doing research, if desired.

5. Chapter 6 discusses the software engineering of expert systems and is meant for those students planning to work on large expert systems. It is not necessary to discuss this chapter before assigning term projects. In fact, it would be best to cover this chapter last so that the student can appreciate all the factors that go into building a quality expert system.

A manual with solutions to the odd-numbered problems and some of the even-numbered programs and another manual of the tables and figures contained in the book suitable for viewgraphs is available from the publisher.

Contributors to CLIPS

We'd like to thank all of the people who contributed in one way or another to the success of CLIPS. As with any large project, CLIPS is the result of the efforts of numerous people. The primary contributors have been: Robert Savely, Chief Scientist of Advanced Software Technology at JSC, who conceived the project and provided overall direction and support; Chris Culbert, Branch Chief of the Software Technology Branch, who managed the project and wrote the original *CLIPS Reference Manual*; Gary Riley, who designed and developed the rule-based portion of CLIPS, co-authored the *CLIPS Reference Manual* and *CLIPS Architecture Manual*, and developed the Macintosh interface for CLIPS;

Brian Donnell, who designed and developed the CLIPS Object-Oriented Language (COOL), coauthored the *CLIPS Reference Manual* and *CLIPS Architecture Manual*, and developed the previous MS-DOS interfaces for CLIPS; Bebe Ly, who developed the X Window interface for CLIPS; Chris Ortiz, who developed the Windows 3.1 interface for CLIPS; Dr. Joseph Giarratano of the University of Houston–Clear Lake, who wrote the *CLIPS User's Guide*; and Frank Lopez, who wrote the original prototype version of CLIPS.

Many other individuals contributed to the design, development, review, and general support of CLIPS including Jack Aldridge, Carla Armstrong, Paul Baffes, Ann Baker, Stephen Baudendistel, Les Berke, Ron Berry, Tom Blinn, Marlon Boarnet, Dan Bochsler, B. L. Brady, Bob Brown, Barry Cameron, Tim Cleghorn, Major Paul Condit, Major Steve Cross, Andy Cunningham, Dan Danley, Mark Engelberg, Kirt Fields, Ken Freeman, Kevin Greiner, Ervin Grice, Sharon Hecht, Patti Herrick, Mark Hoffman, Grace Hua, Gordon Johnson, Phillip Johnston, Sam Juliano, Ed Lineberry, Bowen Loftin, Linda Martin, Daniel McCoy, Terry McGregor, Becky McGuire, Scott Meadows, C. J. Melebeck, Paul Mitchell, Steve Mueller, Bill Paseman, Cynthia Rathjen, Eric Raymond, Reza Razavipour, Marsha Renals, Monica Rua, Tim Saito, Gregg Swietek, Eric Taylor, James Villarreal, Lui Wang, Bob Way, Jim Wescott, Charlie Wheeler, and Wes White.

Acknowledgements

In writing this book, a number of people have made very helpful comments. These include Chris Culbert, Dan Bochsler, Tim Dawn, Ted Leibfried, Jack Aldridge, Bob Lea, Chet Lund, Andre de Korvin, Arlan DeKock, Terry Feagin, and the anonymous reviewers. We appreciate the excellent cooperation and facilities of the Desktop Publishing Center in the AV Center at the University of Houston–Clear Lake. Thanks also to Phillip Johnston, Kwok–Bun Yue, Naveed Quraishi, Jenna Giarratano, Anthony Giarratano, and Andrew Griffin, Ph.D.

Foreword to the First Edition

When AI was born in the mid-fifties, its primary concerns were centered on game playing, planning, and problem solving. In the environment of that era, it would have been very difficult to predict that three decades later the most important application areas of AI would be centered on knowledge engineering and, more particularly, on expert systems.

Expert systems as we know them today have their roots in the pioneering work of Feigenbaum, Lederberg, Shortliffe, and Buchanan at Stanford University in the late sixties and early seventies. What is remarkable about this work—and MYCIN in particular—is that it still serves as a paradigm for much of the current activity in expert and, more generally, knowledge-based systems.

During the past few years, demonstrable successes of expert systems in specialized application areas such as medical diagnosis and computer system configuration have led to an explosion of interest in a much wider variety of applications, ranging from battle plan management and optimal routing to fault diagnosis, optimal portfolio selection, and the assessment of creditworthiness of loan applicants.

The mushrooming of applications has generated two distinct needs: (a) the development of a better understanding of how to deal with the key problems of knowledge representation, inference, and uncertainty management; and (b) the development of expert system shells and/or programming languages which minimize the effort needed for constructing a knowledge representation system and an inference engine for a particular application.

The book written by Joseph Giarratano and Gary Riley is the only book at this juncture which addresses both of these needs. Its first part deals authoritatively with the basic issues underlying the representation of knowledge, logical inference, and the management of uncertainty. In this part, particularly worthy of note are the chapters on reasoning under uncertainty, which present a wealth of information on the basic approaches which are in use today. Included in these chapters are insightful descriptions of the techniques employing classical probability theory, certainty factors *à la* MYCIN and PROSPECTOR, the Dempster-Shafer theory, and fuzzy logic. No other text offers such a wide coverage.

The second part presents a detailed account of CLIPS, an expert system programming language written in C which is intended to serve as a versatile tool for the development and implementation of expert systems. CLIPS is related to OPS5 and ART and has a capability for inference under uncertainty.

In combination, the two parts provide an up-to-date and comprehensive account of the principles of expert systems and their practice. To write a book of this kind is clearly a difficult task and the reader has to be prepared to invest a substantial amount of effort to assimilate the wealth of information presented by

the authors. As one of the most sophisticated and most important areas of AI, the theory and practice of expert systems contain many complex and as yet not fully understood problems. Joseph Giarratano and Gary Riley deserve the thanks of all of us for undertaking a difficult task and making an important contribution to a better understanding of the fundamentals of expert systems and their application to real-world problems.

Lotfi A. Zadeh
Berkeley, CA. (January, 1989)

Contents

CHAPTER 1
Introduction to Expert Systems

1.1 INTRODUCTION

This chapter is a broad introduction to expert systems. The fundamental principles of expert systems are introduced. The advantages and disadvantages of expert systems are discussed and the appropriate areas of application for expert systems are described. The relationship of expert systems to other methods of programming are discussed.

1.2 WHAT IS AN EXPERT SYSTEM?

The first step in solving any problem is defining the problem area or **domain** to be solved. This consideration is just as true in artificial intelligence (AI) as in conventional programming. However, because of the mystique formerly associated with AI, there is a lingering tendency to still believe the old adage "It's an AI problem if it hasn't been solved yet." Another popular definition is that "AI is making computers act like they do in the movies." This type of mind set may have been popular in the 1970s when AI was entirely in a research stage. However, today there are many real-world problems that are being solved by AI and many commercial applications of AI.

Although general solutions to classic AI problems such as natural language translation, speech understanding, and vision have not been found, restricting the problem domain may still produce a useful solution. For example, it is not difficult to build simple natural language systems if the input is restricted to sentences of the form noun, verb, and object. Currently, systems of this type work well in providing a user-friendly interface to many software products such as database systems and spreadsheets. In fact, the parsers associated with popular computer text-adventure games today exhibit an amazing degree of ability in understanding natural language.

As Figure 1.1 shows, AI has many areas of interest. The area of **expert systems** is a very successful approximate solution to the classic AI problem of programming intelligence. Professor Edward Feigenbaum of Stanford University, an early pioneer of expert systems technology, has defined an expert system as "an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for

their solutions." (Feigenbaum 82). That is, an expert system is a computer system that **emulates** the decision-making ability of a human expert. The term *emulate* means that the expert system is intended to act in all respects like a human expert. An emulation is much stronger than a simulation, which is only required to act like the real thing in some respects.

Although a general-purpose problem solver still eludes us, expert systems function well in their restricted domains. As proof of their success, you need only observe the many applications of expert systems today in business, medicine, science, and engineering, as well as all the books, journals, conferences, and products devoted to expert systems.

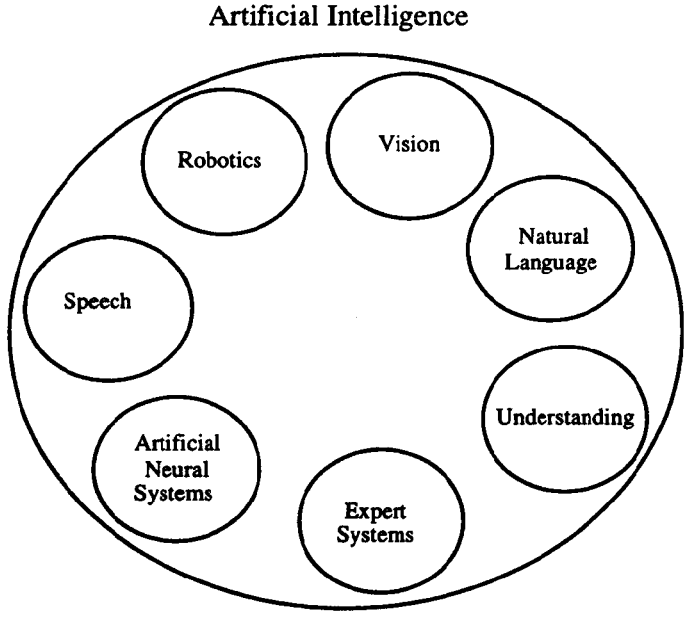


Figure 1.1 Some Areas of Artificial Intelligence

Expert systems is a branch of AI that makes extensive use of specialized knowledge to solve problems at the level of a human **expert.** An expert is a person who has **expertise** in a certain area. That is, the expert has knowledge or special skills that are not known or available to most people. An expert can solve problems that most people cannot solve or can solve them much more efficiently (but not as cheaply). When expert systems were first developed in the 1970s, they contained expert knowledge exclusively. However, the term *expert system* is often applied today to any system that uses expert system technology. This expert system technology may include special expert system languages, programs, and hardware designed to aid in the development and execution of expert systems.

The knowledge in expert systems may be either expertise or knowledge that is generally available from books, magazines, and knowledgeable persons. The terms *expert system*, **knowledge-based system**, or **knowledge-based expert system** are often used synonymously. Most people use the term expert system simply because it's shorter, even though there may be no expertise in their expert system, only general knowledge.

Figure 1.2 illustrates the basic concept of a knowledge-based expert system. The user supplies facts or other information to the expert system and receives expert advice or **expertise** in response. Internally, the expert system consists of two main components. The knowledge base contains the knowledge with which the **inference engine** draws conclusions. These conclusions are the expert system's responses to the user's queries for expertise.

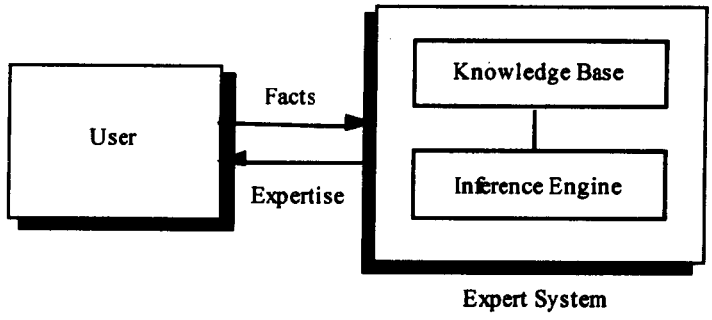


Figure 1.2 Basic Concept of an Expert System Function

Useful knowledge-based systems have also been designed to act as an intelligent assistant to a human expert. These intelligent assistants are designed with expert systems technology because of the development advantages. As more knowledge is added to the intelligent assistant, it acts more like an expert. Thus, developing an intelligent assistant may be a useful milestone in producing a complete expert system. In addition, it may free up more of the expert's time by speeding up the solution of problems. Intelligent tutors are another new application of artificial intelligence. Unlike the old computer-assisted instruction systems, the new systems can provide context-sensitive instruction (Giarratano 91a).

An expert's knowledge is specific to one **problem domain**, as opposed to knowledge about general problem-solving techniques. A problem domain is the special problem area such as medicine, finance, science, or engineering, and so forth that an expert can solve problems in very well. Expert systems, like human experts, are generally designed to be experts in one problem domain. For example, you would not normally expect a chess expert to have expert knowledge about medicine. Expertise in one problem domain does not automatically carry over to another.

The expert's knowledge about solving specific problems is called the **knowledge domain** of the expert. For example, a medical expert system designed to diagnose infectious diseases will have a great deal of knowledge about certain symptoms caused by infectious diseases. In this case, the knowledge domain is medicine and consists of knowledge about diseases, symptoms, and treatments. Figure 1.3 illustrates the relationship between the problem and knowledge domain. Notice that this knowledge domain is entirely included within the problem domain. The portion outside the knowledge domain symbolizes an area in which there is not knowledge about all the problems.

One medical expert system usually does not have knowledge about other branches of medicine such as surgery or pediatrics. Although its knowledge of infectious disease is equivalent to a human expert, the expert system would not