STRUCTURED INDUCTION IN EXPERT SYSTEMS

Alen D. Shapiro

Structured Induction in Expert Systems

Alen D. Shapiro
Cogensys Corporation (formerly at the Turing Institute)

TURING INSTITUTE PRESS
in association with

ADDISON-WESLEY PUBLISHING COMPANY
Wokingham, England · Reading, Massachusetts
Menlo Park, California · New York · Don Mills, Ontario
Amsterdam · Bonn · Sydney · Singapore · Tokyo
Madrid · Bogota · Santiago · San Juan

Cover design by Crayon Design, Henley-on-Thames.
Typeset by Columns, Reading.
Printed in Great Britain by The Bath Press, Avon.
First printed 1987

British Library Cataloguing in Publication Data
Shapiro, Alen D.
Structured induction in expert systems.
(Turing Institute Press).
1. Expert systems (Computer science)
I. Title II. Turing Institute
006.3′3 QA76.76.E95

ISBN 0–201–17813–3

Library of Congress Cataloging-in-Publication Data
Shapiro, Alen D., 1956–
Structured induction in expert systems.
(The Turing Institute Press)
Bibliography: p.
Includes index.
1. Expert systems (Computer science) 2. Electronic data processing – Structured techniques. I. Title.
II. Series: Turing Institute Press (Series)
QA76.76.E95S49 1987 006.3′3 87–14404

ISBN 0–201–17813–3 (Addison-Wesley)

Artificial Intelligence Titles from Addison-Wesley

AI VIDEO COURSES

Davis *Knowledge-Based Expert Systems: Planning and Implementation*

Waterman & Quinlan *Knowledge Acquisition: The Key to Building Expert Systems*

Winston *Artificial Intelligence: Foundations and Applications*

AI MASTERS VIDEOS

Brady *Machine Vision: The Advent of Intelligent Robots*

Kowalski & Kriwaczek *Logic Programming: Prolog and its Applications*

Michie & Bratko *Expert Systems: Automating Knowledge Acquisition*

AI BOOKS

Amble *Logic Programming and Knowledge Engineering*

Anderson, Corbett & Reiser *Essential LISP*

Barr, Cohen & Feigenbaum (Eds.) *The Handbook of Artificial Intelligence* (in 3 volumes)

Bratko *Prolog Programming for Artificial Intelligence*

Brownston, Farrell, Kant & Martin *Programming Expert Systems in OPS5: an Introduction to Rule-Based Programming*

Buchanan & Shortliffe *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*

Burton & Shadbolt *POP-11 Programming for Artificial Intelligence*

Charniak & McDermott *Introduction to Artificial Intelligence*

Clancey & Shortliffe (Eds.) *Readings in Medical Artificial Intelligence: The First Decade*

Conlon *Learning Micro-Prolog*

Craig *Introduction to Robotics: Mechanics and Control*

Fischler & Firschein *Intelligence: The Eye, The Brain and The Computer*

Gale (Ed.) *Artificial Intelligence and Statistics*

Giannesini, Kanoui, Pasero & van Caneghem *Prolog*

Gregory *Parallel Logic Programming in PARLOG: The Language and its Implementation*

Hayes-Roth, Waterman & Lenat (Eds.) *Building Expert Systems*

AI BOOKS (continued)

Jackson *Introduction to Expert Systems*

Kandel *Fuzzy Mathematical Techniques with Applications*

Kearsley (Ed.) *Artificial Intelligence and Instruction: Applications and Methods*

Klahr & Waterman (Eds.) *Expert Systems: Techniques, Tools and Applications*

Manna & Waldinger *The Logical Basis for Computer Programming.* Volume 1: *Deductive Reasoning*

Marcus *Prolog Programming: Applications for Database Systems, Expert Systems and Natural Language Systems*

Pearl *Heuristics: Intelligent Search Strategies for Computer Problem Solving*

Rogers *A Prolog Primer*

Rogers *A Turbo Prolog Primer*

Sager *Natural Language Information Processing: A Computer Grammar of English and its Applications*

Sager, Friedman & Lyman *Medical Language Processing: Computer Management of Narrative Data*

Silverman (Ed.) *Expert Systems for Business*

Sowa *Conceptual Structures: Information Processing in Mind and Machine*

Walker (Ed.), McCord, Sowa & Wilson *Knowledge Systems and Prolog: A Logical Approach to Expert Systems and Natural Language Processing*

Waterman *A Guide to Expert Systems*

Wilensky *Planning and Understanding: A Computational Approach to Human Reasoning*

Winograd *Language as a Cognitive Process.* Volume 1: *Syntax*

Winston *Artificial Intelligence*, 2nd Edition

Winston & Horn *LISP*, 2nd Edition

Structured Induction in Expert Systems

Turing Institute Press

Managing Editor Dr Judith Richards
Academic Editor Dr Peter Mowforth

The Turing Institute, located in Glasgow, Scotland, was established in 1983 as a not-for-profit company, named in honour of the late Alan M. Turing, the distinguished British mathematician and logician whose work has had a lasting influence on the foundations of modern computing.

The Institute offers integrated research and teaching programmes in advanced intelligent technologies – in particular, logic programming, computer vision, robotics and expert systems. It derives its income from research and training contracts, both governmental and industrial, and by subscription from its Industrial Affiliates. It assists Affiliates with the transfer of technology from research to application, and provides them with training for their technical staff, a wide range of software tools, and a comprehensive library and information service.

The Turing Institute is an Academic Associate of the University of Strathclyde, and its research staff work closely with different departments of the University on a variety of research programmes.

Other titles published in association with the Turing Institute Press

Applications of Expert Systems
J. Ross Quinlan (Editor)

Knowledge-Based Programming
Enn Tyugu

This book is dedicated to my wife Merryl who has stood by me through thick and thin, triumph and defeat, keeping me sane and showing me the meaning of love. Also to my parents Rene and Harvey for whose confidence and support I will always be grateful.

Preface

This book is a detailed report of an experiment to determine if machine learning may be used to alleviate the 'expert system bottleneck' (Feigenbaum, 1977). In order to ensure measurability of the results, restricted, yet surprisingly complex, chess endgames were used as an 'experimental test-bench'. Although this is not a book about chess, a level of detail consistent with an experimental report was necessary, specifically in Chapters 3, 6 and 7. A good insight into the complexity of the tasks attempted may be gained by reading these chapters; however, since most necessary information from these chapters is cross-referenced in the text, the casual reader need not feel over-burdened.

The book is aimed at a wide variety of readers and the techniques described have been implemented in commercial rule induction systems which have been successfully applied to 'real-world' problems. The layout of the book is as follows:

Chapter 1 states the problem to be solved, gives a brief history of computer induction and sets the scene for the use of chess as an experimental test-bench.

Chapter 2 describes the programming tools used, namely ID3, Interactive ID3, CLIP/C, decision-vector generators and database generators. The chapter starts with a definition of the chess notation used throughout the rest of the book. Chapter 3 describes the need for, and process of, database generation for result-checking purposes.

Chapter 4 describes how computer induction was leashed in order that it might produce usable products. The techniques of 'structured induction' and 'self-commenting' (including post-processing of self-commentary text) are described. Chapter 5 is an overview of the two experiments performed and their aims.

Chapter 6 describes in detail the three-piece (KPK) endgame solution that was generated. The latter portion of this chapter is given to comparing the cost of generating this structured solution (as a program manufacturing task) with more conventional solutions (unstructured induction and database lookup).

Chapter 7 contains a detailed description of the four-piece (KPa7KR) endgame solution that was generated. The latter portion of this chapter is a report on how structured and unstructured solutions were compared, their run-time efficiency and accuracy.

Chapter 8 is the most important chapter of the book. It contains a general discussion on: a) the effectiveness of computer induction; b) where 'rules of thumb' might fit in; c) if a domain-expert exists, the unsuitability of unstructured induction; d) the measured information content associated with the expert-supplied structure; e) human understandability of machine-generated rules (criteria that would allow such a rule-set, when run, to be called an expert system); f) the nature of rule languages that would only code human-compatible rules (machine-generated or otherwise); and g) the conclusions drawn from this work.

Acknowledgements

Part of the work described in this book was done while the author was in receipt of an SRC Studentship in association with a project funded by SRC grant no. GR/A/80327 made to Professor Donald Michie for software and microelectronic aids for design and implementation of expert systems. The author acknowledges provision of facilities from the University of Edinburgh, Intelligent Terminals Ltd, the Turing Institute and Citicorp/TTI and expresses his thanks to Professor Donald Michie for his encouragement and helpful suggestions in particular as regards the treatment of information measurement followed in Section 8.4. The author also wishes to express his thanks to Dr Timothy Niblett for his help and for the provision of the KPK and KPKR databases and to Ken Thompson for providing the KQKR and KRKR databases. Thanks also to former Scottish Chess Champion Danny Kopec for acting as an expert in providing examples for the KPKR domain and to Rob Gordon and Marcel Schoppers for their invaluable help with the early stages of KPKR. The author also acknowledges financial assistance from International Computers Ltd and the General Electric Company.

Part of Chapter 6 has appeared in *Advances in Computer Chess 3* (Ed. M. R. B. Clarke) Shapiro and Niblett (1982). Most of the work described in this book was submitted in partial requirement for the degree of Ph.D. in Machine Intelligence (Shapiro, 1983). Part of Chapter 4 has appeared in *Advances in Computer Chess 4* (Ed. D. F. Beal) Shapiro and Michie (1986).

The author and publishers would like to thank the *IBM Journal of Research and Development* for permission to reproduce an extract from A. Newell, J. C. Shaw and H. A. Simon's 'Chess-playing programs and the problem of complexity', *IBM J. Res. Dev.* **2**, 320–335. Copyright 1958 by International Business Machines Corporation; reprinted with permission.

Contents

Chapter 1
Introduction

1.1 Motivation

An 'expert system' is a computer program that aims to:

1. emulate or outdo one or more human experts in a skilled diagnostic or other decision-making task; and
2. explain its decisions to the user on demand.

The structure of an expert system can be split into three modules:

- the **inference engine**;
- the **knowledge-base**; and
- the **knowledge acquisition module**.

The knowledge-base contains a representation of expertise in the domain. There is also a **'database'** which contains transient information specific to the current state of the problem. The inference engine dictates how the rules in the knowledge-base are applied to the facts present from time to time in the 'database'. Database is placed in quotation marks because, firstly, this usage is misleading – 'situation model' would perhaps be better – and, secondly, 'database' is used later for something different. Use of the knowledge acquisition module usually requires a partnership between a computer scientist (knowledge engineer) and a specialist (domain expert) in the given field. Sometimes these are one and the same person.

To make an expert system one must choose (or develop) an inference engine and, consulting a domain expert, fill the knowledge-base with information of a type which can be called 'prescriptive'. This typically has the form of 'if–then' rules, each with associated degrees of confidence. For example, *if* (with some degree of certainty) the car battery is flat *then* conclude (with some measure of confidence) that the fan belt is loose. Some expert domains are such that a system with all confidence measures set to 0 or 1 (false or true) is adequate.

The choice of inference engine dictates the user–interface characteristics and defines some ordering over the information contained in the knowledge-base. Designing an inference engine is now well understood. At first glance, knowledge gathering from the domain expert may also not seem to be particularly hard. But it has become increasingly apparent that:

> 'the acquisition of domain knowledge [is] the bottleneck problem in the building of applications-oriented intelligent agents'. (Feigenbaum, 1977)

Even with domain experts who are regularly available (by no means the normal situation since by their nature their time is in heavy demand) one rule per man-day debugged and installed in the knowledge-base is reckoned adequate progress.

What is so difficult about getting correct rules out of an expert, since he is after all an expert? To answer this question it is important to realize that his expertise does not include the ability to explain the reasons for his professional decisions. When a chemical company hires a mass spectroscopist it is renting his ability to *interpret* spectra, not to explain how he makes the interpretations. Hence he is not to be regarded as necessarily expert in this second activity. Indeed in this activity he is not even in the normal sense a professional. Experts typically cannot describe their own reasoning processes. They have to a large extent forgotten how they learned their trade, which tends to be largely based on experience assimilated into a form of intuitive 'know-how'. Moreover, domain experts are seldom computer scientists; hence they do not know how to install rules in a given software system nor do they know the form the rules should take for a particular inference engine. A direct interface between domain expert and expert system is needed. At present the interface is via the knowledge engineer. The knowledge engineer talks to the expert and extracts rules from the explanations he supplies, converting them to machine-acceptable form and pointing out inconsistencies as they are discovered. This is the long, slow process of rule acquisition referred to before. The indications of the present work are that for moderately complex tasks complete success can never be achieved by this method alone, i.e. without use of rule induction. It is significant that the largest operational rule-bases to be built without using induction have not yet much exceeded 2000 rules. Nievergelt (1977) showed that a grand-master's store of chess patterns amounts to some 50 000 in number. Although one pattern is not always equivalent to one rule, the implications for the construction of expert systems for problems of grand-master chess complexity are clear.

However, there is one facet of the expert's skill that until recently has not been utilized: he is able to act as a skilled source of relevant examples to train an apprentice. If this skill could be tapped and fed into an expert system equipped with the power to generalize from examples it should alleviate the knowledge-gathering bottleneck. Michalski and Chilausky (1980) have shown that it is possible by the use of mechanized inductive learning to build a complete expert system from a file of examples. Moreover, the inductively built expert system was not only much cheaper to synthesize than a comparable system hand-built by conventional techniques but also showed strikingly superior accuracy of run-time decisions. The research, which was on diagnosing diseases in soy beans, showed that, at this level of problem complexity, the induced rules were understandable and mentally checkable by human experts in the test domain. These issues of cognitive compatibility are central and are more fully discussed in Section 1.6.

Another feature usually associated with expert systems is that of 'knowledge refinement'. The information content of an active knowledge-base tends to increase as it is tuned and rules are added. It becomes an increasingly accurate store of expert chosen rules that with very little reformatting can be turned into a tutorial manual.

1.2 Historical background

The following selection from published contributions on machine learning over the past 25 years is focused on just those which point to the possibility of incorporating learning in expert systems software. We omit work like that of Samuel (1957) based on the tuning of parameters of a pre-specified description as opposed to the structural modification of descriptions or the generation of new descriptions.

Hunt *et al.*'s (1966) CLS (Concept Learning System) was the first to generate rules automatically from examples. These generalizations were produced in the form of decision trees, functionally equivalent to compound conditional statements.

Michie and Chambers' (1968, 1969) real-time system BOXES 'learned' to balance a pole on a moving cart. The system modified a set of 225 production rules on the basis of trial runs with a simulation displayed on a video monitor. The system could acquire expertise either in stand-alone mode from its own trial and error, or by observing the real-time decisions of an expert trained on the control task.

Winston (1970) and Barrow and Popplestone (1971) independently introduced relational graphs ('semantic nets') to describe visual scenes. Their programs modified these visual descriptions from example scenes.

Michalski, together with Chilausky and Jacobsen (Chilausky *et al.*, 1976), showed cost-benefit advantages, both in the labour of rule-base construction and in run-time performance, of induction over traditional dialogue methods for building an expert rule-base. Michalski later (1980) took his soy bean diagnosis a stage further with new material and multiple sources of expert knowledge for a more detailed comparison. The induced expert system again outperformed an expert system generated by