

---

## **MACHINE LEARNING: A Guide to Current Research**

---

**THE KLUWER INTERNATIONAL SERIES  
IN ENGINEERING AND COMPUTER SCIENCE**

**KNOWLEDGE REPRESENTATION, LEARNING  
AND EXPERT SYSTEMS**

*Consulting Editor*

Tom M. Mitchell

---

# **MACHINE LEARNING**

## **A Guide to Current Research**

edited by

**Tom M. Mitchell**  
Rutgers University

**Jaime G. Carbonell**  
Carnegie-Mellon University

**Ryszard S. Michalski**  
University of Illinois



**KLUWER ACADEMIC PUBLISHERS**  
Boston/Dordrecht/Lancaster

---

**Distributors for North America:**

Kluwer Academic Publishers  
101 Philip Drive  
Assinippi Park  
Norwell, Massachusetts 02066, USA

**Distributors for the UK and Ireland:**

Kluwer Academic Publishers  
MTP Press Limited  
Falcon House, Queen Square  
Lancaster LA1 1RN, UNITED KINGDOM

**Distributors for all other countries:**

Kluwer Academic Publishers Group  
Distribution Centre  
Post Office Box 322  
3300 AH Dordrecht, THE NETHERLANDS

---

**Library of Congress Cataloging-in-Publication Data:**

Machine learning.

(The Kluwer international series in engineering and  
computer science ; SECS 12)

Bibliography: p.

Includes index.

1. Machine learning—Addresses, essays, lectures.
  2. Artificial intelligence—Addresses, essays, lectures.
- I. Mitchell, Tom M. (Tom Michael), 1951-  
II. Carbonell, Jaime G. (Jaime Guillermo)  
III. Michalski, Ryszard Stanislaw, 1937-  
IV. Series.

Q325.M318 1986 006.3'2 86-2766

ISBN-13: 978-1-4612-9406-1 e-ISBN-13: 978-1-4613-2279-5

DOI: 10.1007/978-1-4613-2279-5

---

**Copyright © 1986 by Kluwer Academic Publishers**

Softcover reprint of the hardcover 1st edition 1986

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording, or otherwise, without written permission of the publisher, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA 02061.

Third printing 1988

## **Table of Contents**

CONTRIBUTING AUTHORS	xi
PREFACE	xiii
JUDGE: A CASE-BASED REASONING SYSTEM William M. Bain	1
CHANGING LANGUAGE WHILE LEARNING RECURSIVE DESCRIPTIONS FROM EXAMPLES Ranan B. Banerji	5
LEARNING BY DISJUNCTIVE SPANNING Gary L. Bradshaw	11
TRANSFER OF KNOWLEDGE BETWEEN TEACHING AND LEARNING SYSTEMS P. Brazdil	15
SOME APPROACHES TO KNOWLEDGE ACQUISITION Bruce G. Buchanan	19
ANALOGICAL LEARNING WITH MULTIPLE MODELS Mark H. Burstein	25
THE WORLD MODELERS PROJECT: OBJECTIVES AND SIMULATOR ARCHITECTURE Jaime Carbonell and Greg Hood	29
THE ACQUISITION OF PROCEDURAL KNOWLEDGE THROUGH INDUCTIVE LEARNING Kaihu Chen	35
LEARNING STATIC EVALUATION FUNCTIONS BY LINEAR REGRESSION Jens Christensen	39
PLAN INVENTION AND PLAN TRANSFORMATION Gregg C. Collins	43
A BRIEF OVERVIEW OF EXPLANATORY SCHEMA ACQUISITION Gerald DeJong	47
THE EG PROJECT: RECENT PROGRESS Thomas G. Dietterich	51

LEARNING CAUSAL RELATIONS Richard J. Doyle	55
FUNCTIONAL PROPERTIES AND CONCEPT FORMATION J. Daniel Easterlin	59
EXPLANATION-BASED LEARNING IN LOGIC CIRCUIT DESIGN Thomas Ellman	63
A PROPOSED METHOD OF CONCEPTUAL CLUSTERING FOR STRUCTURED AND DECOMPOSABLE OBJECTS Douglas Fisher	67
EXPLOITING FUNCTIONAL VOCABULARIES TO LEARN STRUCTURAL DESCRIPTIONS Nicholas S. Flann and Thomas G. Dietterich	71
COMBINING NUMERIC AND SYMBOLIC LEARNING TECHNIQUES Richard H. Granger, Jr. and Jeffrey C. Schlimmer	75
LEARNING BY UNDERSTANDING ANALOGIES Russell Greiner	81
ANALOGICAL REASONING IN THE CONTEXT OF ACQUIRING PROBLEM SOLVING EXPERTISE Rogers Hall	85
PLANNING AND LEARNING IN A DESIGN DOMAIN: THE PROBLEMS PLAN INTERACTIONS Kristian J. Hammond	89
INFERENCE OF INCORRECT OPERATORS Haym Hirsh and Derek Sleeman	93
A CONCEPTUAL FRAMEWORK FOR CONCEPT IDENTIFICATION Robert C. Holte	99
NEURAL MODELING AS ONE APPROACH TO MACHINE LEARNING Greg Hood	103
STEPS TOWARD BUILDING A DYNAMIC MEMORY Larry Hunter	109

LEARNING BY COMPOSITION Glenn A. Iba	115
KNOWLEDGE ACQUISITION: INVESTIGATIONS AND GENERAL PRINCIPLES Gary S. Kahn	119
PURPOSE-DIRECTED ANALOGY: A SUMMARY OF CURRENT RESEARCH Smadar Kedar-Cabelli	123
DEVELOPMENT OF A FRAMEWORK FOR CONTEXTUAL CONCEPT LEARNING Richard M. Keller	127
ON SAFELY IGNORING HYPOTHESES Kevin T. Kelly	133
A MODEL OF ACQUIRING PROBLEM SOLVING EXPERTISE Dennis Kibler and Rogers P. Hall	137
ANOTHER LEARNING PROBLEM: SYMBOLIC PROCESS PREDICTION Heedong Ko	141
LEARNING AT LRI ORSAY Yves Kodratoff	145
COPER: A METHODOLOGY FOR LEARNING INVARIANT FUNCTIONAL DESCRIPTIONS Mieczyslaw M. Kokar	151
USING EXPERIENCE AS A GUIDE FOR PROBLEM SOLVING Janet L. Kolodner and Robert L. Simpson	155
HEURISTICS AS INVARIANTS AND ITS APPLICATION TO LEARNING Richard E. Korf	161
COMPONENTS OF LEARNING IN A REACTIVE ENVIRONMENT Pat Langley, Dennis Kibler, and Richard Granger	167
THE DEVELOPMENT OF STRUCTURES THROUGH INTERACTION Robert W. Lawler	173

COMPLEX LEARNING ENVIRONMENTS: HIERARCHIES AND THE USE OF EXPLANATION Michael Lebowitz	179
PREDICTION AND CONTROL IN AN ACTIVE ENVIRONMENT Alan J. MacDonald	183
BETTER INFORMATION RETRIEVAL THROUGH LINGUISTIC SOPHISTICATION Michael L. Mauldin	189
MACHINE LEARNING RESEARCH IN THE ARTIFICIAL INTELLIGENCE LABORATORY AT ILLINOIS Ryszard S. Michalski	193
OVERVIEW OF THE PRODIGY LEARNING APPRENTICE Steven Minton	199
A LEARNING APPRENTICE SYSTEM FOR VLSI DESIGN Tom M. Mitchell, Sridhar Mahadevan, and Louis I. Steinberg	203
GENERALIZING EXPLANATIONS OF NARRATIVES INTO SCHEMATA Raymond J. Mooney	207
WHY ARE DESIGN DERIVATIONS HARD TO REPLAY? Jack Mostow	213
AN ARCHITECTURE FOR EXPERIENTIAL LEARNING Michael C. Mozer, Klaus P. Gross	219
KNOWLEDGE EXTRACTION THROUGH LEARNING FROM EXAMPLES Igor Mozetic	227
LEARNING CONCEPTS WITH A PROTOTYPE-BASED MODEL FOR CONCEPT REPRESENTATION Donna J. Nagel	233
RECENT PROGRESS ON THE MATHEMATICIAN'S APPRENTICE PROJECT Paul O'Rorke	237
ACQUIRING DOMAIN KNOWLEDGE FROM FRAGMENTS OF ADVICE Bruce W. Porter, Ray Bareiss, and Adam Farquhar	241

CALM: CONTESTATION FOR ARGUMENTATIVE LEARNING	247
MACHINE	
J. Quinqueton and J. Sallantin	
DIRECTED EXPERIMENTATION FOR THEORY REVISION AND CONCEPTUAL KNOWLEDGE ACQUISITION	255
Shankar A. Rajamoney	
GOAL-FREE LEARNING BY ANALOGY	261
Alain Rappaport	
A SCIENTIFIC APPROACH TO PRACTICAL INDUCTION	269
Larry Rendell	
EXPLORING SHIFTS OF REPRESENTATION	275
Patricia J. Riddle	
CURRENT RESEARCH ON LEARNING IN SOAR	281
Paul S. Rosenbloom, John E. Laird, Allen Newell, Andrew Golding, and Amy Unruh	
LEARNING CONCEPTS IN A COMPLEX ROBOT WORLD	291
Claude Sammut and David Hume	
LEARNING EVALUATION FUNCTIONS	295
Patricia A. Schooley	
LEARNING FROM DATA WITH ERRORS	299
Jakub Segen	
EXPLANATION-BASED MANIPULATOR LEARNING	303
Alberto Maria Segre	
LEARNING CLASSICAL PHYSICS	307
Jude W. Shavlik	
VIEWS AND CAUSALITY IN DISCOVERY: MODELLING HUMAN INDUCTION	311
Jeff Shrager	
LEARNING CONTROL INFORMATION	317
Bernard Silver	
AN INVESTIGATION OF THE NATURE OF MATHEMATICAL DISCOVERY	321
Michael H. Sims	

x

LEARNING HOW TO REACH A GOAL: A STRATEGY FOR THE MULTIPLE CLASSES CLASSIFICATION PROBLEM Henri Soldano and Hélène Pigot	327
CONCEPTUAL CLUSTERING OF STRUCTURED OBJECTS R. E. Stepp	333
LEARNING IN INTRACTABLE DOMAINS Prasad V. Tadepalli	337
ON COMPILING EXPLAINABLE MODELS OF A DESIGN DOMAIN Christopher Tong	343
WHAT CAN BE LEARNED? L.G. Valiant	349
LEARNING HEURISTIC RULES FROM DEEP REASONING Walter Van De Velde	353
LEARNING A DOMAIN THEORY BY COMPLETING EXPLANATIONS Kurt VanLehn	359
LEARNING IMPLEMENTATION RULES WITH OPERATING- CONDITIONS DEPENDING ON INTERNAL STRUCTURES IN VLSI DESIGN Masanobu Watanabe	363
OVERVIEW OF THE ODYSSEUS LEARNING APPRENTICE David C. Wilkins, William J. Clancey, and Bruce G. Buchanan	369
LEARNING FROM EXCEPTIONS IN DATABASES Keith E. Williamson	375
LEARNING APPRENTICE SYSTEMS RESEARCH AT SCHLUMBERGER Howard Winston, Reid Smith, Michael Kleyn, Tom Mitchell, and Bruce Buchanan	379
LANGUAGE ACQUISITION: LEARNING PHRASES IN CONTEXT Uri Zernik and Michael Dyer	385
REFERENCES	391
INDEX	425

## CONTRIBUTING AUTHORS

William M. Bain	Ranan B. Banerji	Ray Bareiss
Gary L. Bradshaw	P. Brazdil	Bruce G. Buchanan
Mark H. Burstein	Jaime Carbonell	Kaihu Chen
Jens Christensen	William J. Clancey	Gregg C. Collins
Gerald DeJong	Thomas G. Dietterich	Richard J. Doyle
Michael Dyer	J. Daniel Easterlin	Thomas Ellman
Adam Farquhar	Douglas Fisher	Nicholas S. Flann
Andrew Golding	Richard H. Granger, Jr.	Russell Greiner
Klaus P. Gross	Rogers Hall	Kristian J. Hammond
Haym Hirsh	Robert C. Holte	Greg Hood
David Hume	Larry Hunter	Glenn A. Iba
Gary S. Kahn	Smadar Kedar-Cabelli	Richard M. Keller
Kevin T. Kelly	Dennis Kibler	Michael Kleyn
Heedong Ko	Yves Kodratoff	Mieczyslaw M. Kokar
Janet L. Kolodner	Richard E. Korf	John E. Laird
Pat Langley	Robert W. Lawler	Michael Lebowitz
Alan J. MacDonald	Sridhar Mahadevan	Michael L. Mauldin
Ryszard S. Michalski	Steven Minton	Tom M. Mitchell
Raymond J. Mooney	Jack Mostow	Michael C. Mozer
Igor Mozetic	Donna J. Nagel	Allen Newell
Paul O'Rorke	Hélène Pigot	Bruce W. Porter
J. Quinqueton	Shankar A. Rajamoney	Alain Rappaport
Larry Rendell	Patricia J. Riddle	Paul S. Rosenbloom
J. Sallantin	Claude Sammut	Jeffrey C. Schlimmer
Patricia A. Schooley	Jakub Segen	Alberto Maria Segre
Jude W. Shavlik	Jeff Shrager	Bernard Silver
Robert L. Simpson	Michael H. Sims	Derek Sleeman
Reid Smith	Henri Soldano	Louis I. Steinberg
R. E. Stepp	Prasad V. Tadepalli	Christopher Tong
Amy Unruh	L.G. Valiant	Walter Van De Velde
Kurt VanLehn	Masanobu Watanabe	David C. Wilkins
Keith E. Williamson	Howard Winston	Uri Zernik

## PREFACE

One of the currently most active research areas within Artificial Intelligence is the field of Machine Learning, which involves the study and development of computational models of learning processes. A major goal of research in this field is to build computers capable of improving their performance with practice and of acquiring knowledge on their own.

The intent of this book is to provide a snapshot of this field through a broad, representative set of easily assimilated short papers. As such, this book is intended to complement the two volumes of *Machine Learning: An Artificial Intelligence Approach* (Morgan-Kaufman Publishers), which provide a smaller number of in-depth research papers. Each of the 77 papers in the present book summarizes a current research effort, and provides references to longer expositions appearing elsewhere. These papers cover a broad range of topics, including research on analogy, conceptual clustering, explanation-based generalization, incremental learning, inductive inference, learning apprentice systems, machine discovery, theoretical models of learning, and applications of machine learning methods. A subject index is provided to assist in locating research related to specific topics.

The majority of these papers were collected from the participants at the Third International Machine Learning Workshop, held June 24-26, 1985 at Skytop Lodge, Skytop, Pennsylvania. While the list of research projects covered is not exhaustive, we believe that it provides a representative sampling of the best ongoing work in the field, and a unique perspective on where the field is and where it is headed.

We wish to express our thanks to the many authors who contributed research summaries. Special thanks go to Chris Tong, who generously contributed the subject index for the book, and to Patricia Riddle and Michael Barley who pulled off a minor miracle by collecting and reformatting the authors' papers and text files, and producing the camera-ready manuscript copy for the book.

We also wish to acknowledge those involved in supporting the Third International Machine Learning Workshop: Jo Ann Gabinelli and the rest of the Local Arrangements Committee, who organized a very smoothly run workshop, and the Office of Naval Research and the Rutgers University Laboratory for Computer Science Research, who provided financial support.

This book was prepared at Rutgers University, using the text processing and printing facilities in the Laboratory for Computer Science Research.

**Tom M. Mitchell**  
**Jaime G. Carbonell**  
**Ryszard S. Michalski**

---

## **MACHINE LEARNING: A Guide to Current Research**

# **JUDGE: A CASE-BASED REASONING SYSTEM**

**William M. Bain**

Yale University, Department of Computer Science  
10 Hillhouse Ave., New Haven, CT, 06520

## **SUMMARY OF RESEARCH**

People tend to improve their abilities to reason about situations by amassing experiences in reasoning. The more situations which a person knows about, the more able he is to account for feature differences between a new input and old knowledge. A computer program which can improve its ability to reason must also have access to situations which it has previously analyzed or reasoned about. Previous experiences thus require some mechanism for orderly storage and retrieval. The inability to save accounts of previous experiences for future application and modification represents a serious shortcoming of most, if not all, rule-based expert systems.

A reasoner's goals and beliefs can influence how he (or it) processes input situations in several ways. Chief among these are the abilities to store experiences differentially and to retrieve particular kinds of episodes. Goals can help to impart a systematicity to the task of organizing memory by virtue of the fact that they can be prioritized with respect to one another. A goal which is hierarchically more important than others will have an earlier and more substantive role in comparing and indexing episodes. For example, the fact that a person is wearing a particularly attractive blue shirt pales in significance to most people if the person is also using a gun to hold them up. The greater relevance of the latter feature can be demonstrated by the experiencer recalling the situation faster with a query such as, "Remember the person who held you up last month?" than with, "Remember the blue shirt that you liked that you saw someone wearing last month?" or by his not remembering the experience at all with the second query.

The JUDGE system has been developed to demonstrate the utility of using the goals of a program to reason about input situations by indicating differences (and their effects) between the input and other stored situations, and then storing the input along with some notation of the reasoning which was applied to it. This is called "case-based" reasoning. In addition to storing episodes in and retrieving them from its memory, the system is also able to construct, use and modify its own rules for

sentencing cases. In the domain of criminal law concerning the formation of sentences for criminal cases, the "goals of the program" refer to features which are significant to human judges. These include the extent of harm suffered by a victim, interpretations which both explain a person's actions, such as "self-defense," and which suggest a degree to which an action was justifiable, and the relative importance of mediating circumstances. The manner in which input items are stored reflects their relationships to those situations already stored, with respect to the goals of the system. Because the stored items in the system's memory change, JUDGE's reasoning changes; new inputs are compared with items in a memory which is evolving.

A second processing influence of goals and beliefs in the JUDGE system is manifested in the types of inferences the system makes about motives which underlie people's actions. People seem to interpret events not only in terms of the goals which may have motivated the actors involved in the events, but in terms of their own goals as well [56]. The program models this feature of human understanding by changing the inferences it makes if certain of the goals specified to it are changed. JUDGE analyzes criminal cases involving assault and murder from the points of view of the lawyers and judge who might be involved with such cases. In this domain, the justification (or lack of it) of an agent's actions is used as a metric for determining the extent to which an interpretation is favorable for satisfying the interpreter's goals.

## SOME CONDENSED OUTPUT SHOWING JUDGE IN ACTION SENTENCING TWO CASES

Now processing CRIMEO:

FIRST, TED SLASHED AT AL WITH A KNIFE ONE TIME, CUTTING AL'S SKIN SLIGHTLY.  
NEXT, AL SLASHED AT TED WITH A KNIFE ONE TIME, CUTTING TED'S SKIN SLIGHTLY.  
FINALLY, TED STABBED AL WITH A KNIFE SEVERAL TIMES. AL DIED.  
-- after an interpretation phase, the program considers indices  
-- found to any other crimes in memory.

NO CRIME IN MEMORY WAS FOUND TO MATCH CRIMEO.

CONSTRUCTING A SENTENCE FROM SCRATCH BASED ON FINAL ACTIONS WHICH WERE INTERPRETED AS ESCALATED-RETALIATION WITH AN ACHIEVED-RESULT. . .

THE SENTENCE TO BE GIVEN FOR CRIMEO, FOR VIOLATION OF MURDER AS DEFINED IN SECTION 53A-45 IN THE CONNECTICUT PENAL CODE, WILL BE A TERM OF IMPRISONMENT OF NOT LESS THAN 40 YEARS NOR MORE THAN 50

YEARS TO BE SERVED IN THE STATE'S PRISON. [...]

-- the crime and its sentence are stored in memory.

---

Now processing CRIME1:

FIRST, RANDY STRUCK CHUCK WITH HIS FISTS SEVERAL TIMES, BARELY HURTING CHUCK.

NEXT, CHUCK STRUCK RANDY WITH HIS FISTS SEVERAL TIMES, BARELY HURTING RANDY.

THEN, RANDY SLASHED AT CHUCK WITH A KNIFE ONE TIME, CUTTING CHUCK'S SKIN.

NEXT, CHUCK SLASHED AT RANDY WITH A KNIFE ONE TIME, CUTTING RANDY'S SKIN.

FINALLY, RANDY STABBED CHUCK WITH A KNIFE SEVERAL TIMES. CHUCK DIED.

-- again, after interpretation, the program looks at old crimes  
-- for which it has found indices.

STATUTORY SIMILARITY OF MURDER FOUND WITH INDEXED CRIME: CRIMEO. FOUND PREVIOUS CRIMES IN WHICH THE SAME FINAL ACTION AND RESULT WAS ALSO PERPETRATED AGAINST AN INITIAL-DEFENDER : (CRIMEO)

RECENT PREVIOUS INSTANCE FOUND WITH SAME FINAL RESULT AS FINAL RESULT IN CURRENT CASE-- KILL: CRIMEO

RECENT PREVIOUS INSTANCE FOUND WITH SAME FINAL ACTION AS FINAL ACTION IN CURRENT CASE-- STAB-KNIFE: CRIMEO

RECENT PREVIOUS INSTANCE FOUND WITH SAME FINAL RESULT INTERPRETATION AS FINAL INTERPRETATION IN CURRENT CASE--

ACHIEVED-RESULT: CRIMEO

RECENT PREVIOUS INSTANCE FOUND WITH SAME FINAL INTENT INTERPRETATION LEVEL AS FINAL INTERPRETATION IN CURRENT CASE--

ESCALATED-RETALIATION: CRIMEO

A PREVIOUS SITUATION INVOLVING SIMILAR ACTIONS AND A KILL RESULT, CRIMEO, WAS QUITE SIMILAR TO THE CURRENT CASE.

CRIME1 CONTAINED THE SAME SEQUENCE OF ACTIONS AND RESULTS AS CRIMEO, EXCEPT THAT IT WAS PRECEDED BY HITTING.

-- What follows is an in-depth comparison of the input crime  
-- with a crime retrieved from memory, CRIMEO.

---

===== Comparing CRIMEO from memory with the current input, CRIME1. =====

---

IN BOTH CRIMES, THE-VICTIM WAS KILLED. NOT ONLY WERE BOTH OF THESE OUTCOMES THE RESULT OF DIRECT-INTENTIONS, BUT THE ACTORS INTENDED AND CAUSED THE SAME AMOUNT OF HARM.

THE INTENT OF BOTH OFFENDERS WAS TO ACT REPEATEDLY TO STAB THE-VICTIM TO DEATH. IN ADDITION, NEITHER ACTOR'S INTENTIONS WERE JUSTIFIED, AND BOTH ESCALATED THE LEVEL OF VIOLENCE ABOUT THE SAME DEGREE.

\*\*\*\*\* Considering actions of the victims which led \*\*\*\*\*  
\*\*\*\*\* to the offender's actions just compared. . . \*\*\*\*\*

IN BOTH CRIMES, THE-OFFENDER'S SKIN WAS SPLIT OPEN. NOT ONLY WERE BOTH OF THESE OUTCOMES THE RESULT OF BUNGLING, BUT THE ACTORS INTENDED AND CAUSED THE SAME AMOUNT OF HARM.

THE INTENT OF BOTH VICTIMS WAS TO ACT ONLY ONE TIME TO STAB THE-OFFENDER. IN ADDITION, THE INTENTIONS OF BOTH ACTORS WERE JUSTIFIED, AND NEITHER INTENDED TO ESCALATE THE LEVEL OF

VIOLENCE.

\*\*\*\*\* Considering actions of the offenders which led \*\*\*\*\*  
\*\*\*\*\* to the victim's actions just compared. . . . \*\*\*\*\*

IN BOTH CRIMES, THE-VICTIM'S SKIN WAS SPLIT OPEN. ALTHOUGH BOTH OF THE OUTCOMES WERE THE SAME, RANDY INTENDED TO CAUSE MORE HARM THAN TED.

TED DEMONSTRATED AN EXTREME USE OF FORCE AGAINST AL WHEN HE ACTED TO STAB AL. THIS ACTION WAS UNPROVOKED.

RANDY DEMONSTRATED AN EXTREME USE OF FORCE AGAINST CHUCK WHEN HE ACTED TO STAB CHUCK IN RESPONSE TO BEING HIT HARD.

THE MAGNITUDE OF THE EXTREME FORCE USED IN THE FIRST CRIME WAS GREATER THAN THAT IN THE SECOND, AND SO THE FIRST CRIME WILL BE CONSIDERED WORSE.

THE INTENT OF BOTH OFFENDERS WAS TO ACT ONLY ONE TIME TO STAB THE-VICTIM. HOWEVER, ALTHOUGH NEITHER ACTOR'S INTENTION WAS JUSTIFIED AND BOTH OF THEM ESCALATED THE LEVEL OF FORCE, TED ESCALATED MORE.

COMPARISON FINISHED WITH RESULT THAT THE OLD CRIME, CRIME0, IS SOMEWHAT WORSE. BECAUSE CRIME1 HAS A GREAT DEAL OF SIMILARITY WITH CRIME0, CRIME1 WILL GET A SENTENCE WHICH IS JUST THE SAME AS THE SENTENCE FOR CRIME0.

THE SENTENCE TO BE GIVEN FOR CRIME1, FOR VIOLATION OF MURDER AS DEFINED IN SECTION 53A-45 IN THE CONNECTICUT PENAL CODE, WILL BE A TERM OF IMPRISONMENT OF NOT LESS THAN 40 YEARS NOR MORE THAN 50 YEARS TO BE SERVED IN THE STATE'S PRISON. OF THIS SENTENCE, 25 YEARS MAY NOT BE REDUCED OR SUSPENDED, ACCORDING TO STATE LAW.

-- because these crimes were quite similar, a general  
-- sentencing rule is formed.

FORMING GENERAL SENTENCING RULE:

FOR VIOLATION OF MURDER. . . .  
FOR CAUSING RESULT OF KILL. . . .  
FOR USING ACTION OF STAB-KNIFE. . . .  
FOR HARMING INITIAL-DEFENDER. . . .  
FOR RESPONDING TO A STAB-PUNCTURE-KNIFE LEVEL OF HARM. . . .  
FOR USING ESCALATED FORCE AND ACTING OUT OF RETALIATION. . . .  
FOR INTENDING TO CAUSE THE RESULT. . . .

THE SENTENCE FOR THIS VIOLATION WILL BE THE SAME AS GIVEN FOR CRIME1 ABOVE.

-----

## ACKNOWLEDGMENTS

This work was supported in part by the Air Force Office of Scientific Research under contract F49620-82-K-0010.

# CHANGING LANGUAGE WHILE LEARNING RECURSIVE DESCRIPTIONS FROM EXAMPLES

Ranan B. Banerji

Saint Joseph's University  
Philadelphia, Pa. 19131

## INTRODUCTION

Shapiro [330] has described a reasonably efficient algorithm for inducing first order theories of models from facts. He proved that the algorithm converges as long as the various parameters appearing in the learning algorithm satisfied certain well defined constraints. Within the limitations of these constraints, however, there was sufficient flexibility in the choice of the parameters.

One parameter was the language. He accepted the limitation that it would be the language of first order logic and that each sentence in the theory would be a clause, with universal quantifiers removed and existential quantifiers handled through Skolem functions. Within this limitation, the number and arity of predicate and function letters did not matter.

He accepted the possibility that only a sublanguage of the total language would be used for the formation of theories (a theory being a set of sentences). His examples and experiments almost all restricted this "hypothesis language" to restricted classes of Horn clauses. The "language for expressing facts" was a sublanguage of this latter language; Shapiro mostly used ground atoms and ground literals here.

The algorithm ran as follows. He started with the sentence "false" as the most general theory, capable of explaining all facts. As examples of a false fact were obtained, and it was found that the theory predicted the fact to be true, then one located the sentence in the theory from which the false fact followed. This culprit sentence was replaced by a set of sentences less general than the culprit sentence. This set was a well defined function of the culprit sentence (Shapiro called it rho).

Shapiro required the parameters to satisfy a number of criteria. Two major ones were: 1) That every possible sentence of the hypothesis language could be generated by applying rho repeatedly to the starting theory, and 2) On looking at a fact sentence one could predict a number (a function of the sentence) such that if a proof for the sentence existed in the theory, it would not involve more than this number of steps. As long as these and a few other more technical criteria were met, he could

prove that the algorithm would "learn in the limit" (a concept introduced by Gold [131]). That is, after a finite amount of time the algorithm would produce a theory which would explain all the true facts.

Followers of the VS approach to learning would find the idea of learning in the limit unsatisfactory, since while such an algorithm does learn in finite time, it is not known beforehand as to when that time will come. In contrast, when the VS collapses in the LEX-like [256] learning process, we know that we have reached an unique description in the language. However, as we shall see, the main thrust of this work is in not placing any faith on the adequacy of the language initially chosen. So, the collapse of the version space, instead of signalling the end of learning can also signal the inadequacy of the language. In such a situation, the Shapiro approach is the only one I know which has the same level of theoretical solidity as the VS approach.

Shapiro's rho was defined once and for all and was incorporated as a parameter (together with the hypothesis and fact languages) to the learning algorithm. Also, it was assumed that the predicates and functions in the language was known to the algorithm initially.

In my present work I am trying to remove these two restrictions, both of which seem to be unrealistic to me. The rho function essentially determines the rate and the efficiency of learning. A common experience in life is that ones ability to learn increases as one learns more. Also, very often the predicates used by the trainer (the "real world") is not always all known, but are experienced as learning proceeds. It therefore seems desirable to investigate systems where the language grows as more examples are seen and (more importantly, perhaps) where the search for theories gets guided by the knowledge acquired in previous learning. Sammut's MARVIN [310] had some of these properties; but it depended heavily on the facts being presented in a "proper" sequence.

In what follows I will describe my present design of a hypothesis language, a fact language and a rho which enriches itself as learning proceeds, both with respect to automatic enrichment of the input language as well as an automatic enrichment of the rho function. The language has a smaller expressive power than a full blown first order language but is adequate for doing the kind of stuff we do with LEX.

## THE LANGUAGES

The language of facts will consist of the names of the concepts we want to teach (and these could be names of sets as well as names of relationships - the need for relations to be learnable has been argued often

by me [13]). It will also consist of function symbols, which represents measurements made on example objects or measurements made on parts of them - so that functions need to be composed quite often. We shall also have variables and constants, as well as the equality sign. A fact would be a Horn Clause whose head has the concept name applied to a requisite number of variables and whose body is a set of predicates of the form  $f(x)=v$  where  $v$  is a constant and  $f$  a composition of functions. A typical example of a relation could be

$$\text{less}(x,y):-\text{car}(x)=1,\text{car}(y)=0,\text{car}(\text{cdr}(x))=0,\text{car}(\text{cdr}(y))=1$$

saying that the list 10 is lexicographically smaller than the list 01, reading from right to left.

The hypothesis language would have Horn Clauses also, but their bodies, instead of showing equality between constants and composition of measurements, could also contain concept names and other predicates. The heads could contain predicates which are not concept names. These latter would be what pattern recognition people call features. A typical sentence in a theory could be

$$\text{less}(x,y):-\text{less}(\text{cdr}(x),\text{cdr}(y)).$$

## THEORY UPDATING

Unlike Shapiro, we start with an empty theory, so nothing can be explained. This is because at this moment the language is empty also, so far as the learning algorithm goes. The language we described above was the language for the designer.

Both a language and the theory remains empty as long as facts come in which are tagged false (these are negative examples of concepts and so we do not need the theory to recognize them). When a positive fact comes in, the concept name is stored as the language, as well as the associated function and value names.  $P(x):-$  is stored as the theory. If any previous negative fact contradicts this, then  $P(x):-$  is replaced by  $P(x):-f(x)=v$ , where  $f(x)=v$  is one of the elements of the language which occurs in the body of the same concept. One sentence for each element is stored in the theory. At this point the description of the concept would be disjunctive.

At any time a sentence has to be updated, one adds to it's body an element of the body of a positive example of the same concept.

So far all I have done is to define a rho function which fulfills the Shapiro criterion that any sentence S in the theory implies all elements of  $\rho(S)$ . But this updating procedure is guided only by the language, not by any theory that has been learned. To incorporate learning into the learning algorithm itself we need a few more updating rules.

Let us suppose that one of the elements of the theory at a certain time contains the sentence

$$O(x):-B(x).f(x)=v$$

and the updating process has just introduced the sentence

$$P(x):-B(g(x)).A(x).f(g(x))=v.$$

Then, instead of introducing this last sentence into the theory, the algorithm would introduce

$$P(x):-A(x).O(g(x))$$

which is more general.

Thus the rho function modifies itself as learning proceeds. The advantage of doing this has been illustrated by Sammut in many examples. As a matter of fact, infinite classes can not be described except by such descriptions, since recursion has this form.

To introduce new predicates into the language I am presently foreseeing only the present subterfuge. If

$$P(x):-A(x).B(x)$$

and

$$P(x):-A(x).C(x)$$

are both in the theory, then they are replaced by

$$P(x):-A(x).T(x)$$

$$T(x):-B(x).C(x).$$

The updating rules have to be somewhat enriched so these new kinds

of sentences can also be updated. Space prohibits a detailed discussion of these. Basically, not only can we update these by adding conjuncts as before, but also by replacing the predicates in the body by the bodies of sentences in which the predicates appear as heads.

# LEARNING BY DISJUNCTIVE SPANNING

## Gary L. Bradshaw

University of Colorado, Institute of Cognitive Science  
Boulder, Colorado 80309

### ABSTRACT

A new concept-learning technique, **disjunctive spanning**, was developed to identify perceptual patterns. Similar to the A<sup>q</sup> algorithm [227], this technique identifies disjunctive concepts without searching through a hierarchically-ordered generalization space. The disjunctive spanning algorithm was implemented in **NEXUS**, a speech recognition system designed to acquire knowledge about minimal articulatory units in speech recognition. Performance tests of NEXUS show it to be superior to a traditional speech recognition system when both systems were tested on the same highly-confusable vocabulary set.

### INTRODUCTION

Developing effective computer-based recognition systems that work directly from raw sensory input has proven difficult. To date, no general-purpose systems have been demonstrated that approximate the accuracy or flexibility of human perception. A major challenge is to adequately represent natural perceptual concepts. In speech recognition, random variations in the trajectory of articulatory structures cause significant differences in the pattern of acoustic energy of sounds, even when the vocabulary is restricted to a small set of isolated words. Within-group variability (between instances of the same word) is often larger than between-group differences that distinguish similar words, leading to confusion errors.

In order to acquire a description of perceptual concepts, concept induction methods are used to develop models of patterns and their variability. Generalization methods include statistical processes of averaging and variance weighting [104], as well as familiar heuristic techniques (see [93]). Discrimination methods include statistical clustering procedures [104], discrimination nets [114], along with heuristic concept discrimination techniques.

Investigators working in pattern recognition have favored statistical techniques, due to their mathematical tractability, yet limitations are

becoming apparent. These include restrictions on the types of generalizations that may be formed, and dependence on assumptions (such as independence and normality of parameter distribution) being met. Natural perceptual concepts appear to lie outside the search space considered by statistical techniques. As a result, both vision and speech recognition systems founded upon these techniques have been limited and error-prone.

Heuristic machine-learning techniques represent a promising alternative for the discovery of sophisticated abstract concepts. Yet their application to perceptual patterns (corresponding to raw sensor data) posed unexpected difficulties. Perceptual concepts consist of a very large number of numeric measurements. The generalization space is larger by many orders of magnitude than spaces created when using simple symbolic concepts. Humans appear to be more sensitive to intensity relationships than actual to parameter values. Transforming the concept space to search for relationships (which correspond to features of speech) would permit a simpler expression of concepts, yet this space is intractably large. To encode even the most trivial of speech patterns, more than  $10^{90}$  states would be necessary. Needless to say, searching efficiently and effectively through such large spaces is an unpromising prospect.

## DISJUNCTIVE SPANNING

**Disjunctive spanning** was developed as an alternative method to search concept spaces without the necessity of moving through a generalization hierarchy. Disjunctive spanning may be described as an incremental data-driven technique capable of identifying a set of variant instances for each concept when provided feedback about correct and incorrect classifications. Most systems capable of both generalization and discrimination use positive classification evidence to generalize a concept, and utilize errors to discriminate the overly general concept. Disjunctive spanning maintains the generalization heuristic, but instead of splitting a generalized concept, negative evidence is used to begin a new variant of the concept. Although the technique is not guaranteed to identify the minimal set of disjunctive variants, a simple procedure can evaluate the utility of variants, eliminating redundant or useless items, thereby minimizing the number of variants needed to represent a particular concept.

The method utilizes three simple heuristics. The disjunction heuristic tests to see if a simple generalization process would have prevented a classification error. If not, the heuristic posits that the test instance is a new variant of the concept, and adds the instance into the concept set.

Whenever a correct recognition is made, the generalization heuristic causes the system to generalize the stored concept by averaging it with the correctly-identified instance. Finally, the redundancy heuristic periodically prunes variants from the concept set that have not proven useful in the identification process.

Disjunctive spanning is most appropriate in pattern matching situations where decisions are made on the degree of match between the test stimulus and stored patterns, instead of decisions based on an exact match between the test stimulus and stored patterns. Degree of match situations frequently occur when the patterns have numeric, instead of symbolic, features. Disjunctive spanning operates in a recognize-learn cycle, where a test stimulus is classified, then concept-set modifications are made based on performance feedback. The current implementation begins with the null set of concepts, adding new concept classes and concept variants to maximize its classification performance.

## LEARNING AND THE NEXUS SYSTEM

NEXUS [36], incorporates disjunctive spanning in a speech recognition system as a means to develop a set of minimal acoustic units used to recognize words. The database used to test NEXUS consists of thirty tokens of each of the spoken names of letters of the alphabet provided by two speakers. This compact vocabulary presents a difficult challenge for recognition systems due to the high similarity between many words ("B","C","D" etc.). Recognition rates averaged 93% and 94% for the two speakers. A traditional speech recognition system, CICADA, was tested on the same database. Recognition errors were 79% and 81%, respectively. This result would be of little interest if NEXUS found every instance of each word to be a unique variant, and was led to store all previously-heard instances of every word. The average number of variant descriptions identified by NEXUS for each word was only 1.73 and 1.23 variants per word for the two speakers.

## CONCLUSION

Disjunctive spanning has been shown to be a useful technique for concept induction. The technique can construct a set of concepts, where each concept is represented by a set of disjunctive instances. Because disjunctive spanning does not move up and down in a generalization hierarchy, a credit assignment routine capable of localizing the source of the error is unnecessary. This characteristic is very useful when working with

numeric patterns that rely on graded, instead of all-or-none, matching processes.

## **ACKNOWLEDGMENTS**

I developed the NEXUS speech recognition project for my doctoral thesis at Carnegie-Mellon University. I would like to thank my committee members, Herbert Simon, John Anderson, Raj Reddy, and Brian MacWhinney for their patience and support during this project. Thanks are also due to NSF, whose financial support from NSF grant DCR-8205539 provided the physical facilities necessary for the project.

# **TRANSFER OF KNOWLEDGE BETWEEN TEACHING AND LEARNING SYSTEMS**

**P. Brazdil**

Faculdade de Economia  
Rua Dr.R. Frias, 4200 Porto, Portugal

## **ABSTRACT**

A view is put forward that to obtain an effective model of the interaction between the teaching and the learning system, we need to examine the individual communicative acts together with the strategies that the individual systems might be following. In this paper certain representational issues are addressed, such as how we can talk about the behavior of various systems and subsystems in the framework of logic.

## **DIALOGUE GAMES AND LEARNING**

As intelligent systems will become more common in future, we shall have to consider how the knowledge acquired by one system could be transferred over to whoever needs it. Thus one system will be involved in teaching, and the other in learning. We believe that in order to obtain an effective model of the interaction between teaching system (TS) and learning system (LS) we need to consider them together, as if they were part of one complex system, and then see which rules they should follow. Both the teaching and the learning system are effectively involved in a dialogue, or as Carlson [62] puts it, in a dialogue game, both striving to achieve a common understanding on certain issues. It is the long term objectives of both participants and certain conventions that give a meaning to the individual communicative acts.

Teaching can thus be viewed as a sequence of communicative acts which need to be planned in general so as to make teaching more effective. The role of the teacher (learner) is to determine what the objectives of the learner (teacher) are, reconcile them with one's own objectives and then see how these objectives might best be achieved. In order to construct a model of this interaction, we have decided to divide it into two distinct phases: first, to consider all the individual communicative acts that the teaching and the learning systems might use in their interaction, and then to see how more complex teaching and learning strategies could be defined. Currently we are concerned mainly with the first phase.

When trying to create a model of this interaction, we had to make certain design decisions, for example, how to represent knowledge. We have chosen logic, or more precisely Horn clauses as in Kowalski [186], to represent all kinds of knowledge including the knowledge about knowledge (meta-knowledge). The meta-knowledge is represented using certain metalogical primitives discussed below.

## SOME ASSUMPTIONS CONCERNING SYSTEMS

Both the teaching and the learning system are considered to be two separate systems, each following its own reasoning and with an access to its own knowledge base only. Communication between different systems is established using the metapredicates send/receive which can be used to transfer logical expressions from the sender to the recipient. The metapredicate "receive" enables the recipient to read the expression in and execute it.

As in communication between people different systems may interpret the same expression differently, and this is why we want also the systems to be capable of reasoning about different interpretations and exploit it in various ways. In order to be able to do this, we shall follow Bowen and Kowalski [34] and add the definition of provability to our language, obtaining thus an amalgamation of object language with its own metalanguage. The provability relation will be represented here by

$$\text{do}(A:C1: Q \leftarrow As),$$

where A identifies the agent (system), C1 a particular set of clauses which may be used in the proof, Q represents the conclusion and As the assumptions. The clause set C1 may be regarded as a particular theory chosen for the proof of  $Q \leftarrow As$ . The metapredicate  $\text{do}(\dots)$  which is similar to the metapredicate  $\text{demo}(\dots)$  of Bowen and Kowalski [34], and also to the metapredicate  $\text{PR}(\dots)$  of Konolige [180], provides us with a convenient way of formulating a query. If

$$\text{do}(A:C1: Q \leftarrow As)$$

is supplied to system A as a goal, and As is left uninstantiated while Q is given, the system will try to use the clause set C1 to find the right substitution for As. In this context Q can be regarded as the query and As as the answer. The answer may be either "true" or "false" or alternatively it may be in the form of a logical expression specifying under what circumstances Q holds.

## REASONING ABOUT SYSTEMS'S ABILITIES

Every good teacher knows that in order to do effective teaching one has to consider what the student knows, and also what he can learn. In other words, the teacher has to be able to create a model of what the student knows at various stages. But how can such models be created?

Having adopted the metapredicate  $\text{do}(\dots)$  among the basic primitives, conceptually the simplest type of model consists of assertions of the form  $\text{do}(A:C1: Qi \leftarrow Ai) \leftarrow$ , stating that A's answer to query  $Qi$  is  $Ai$  for the clause set  $C1$ . The learning system may thus store all the queries and answers that the teaching system has given as this was done, for example, in Shapiro's debugging system [329]. The teaching system may create a similar model of the learning system. Although such models are not very sophisticated they are quite useful, as they enable the learning (teaching) system to evaluate one of its own models without the help of the teaching (learning) system. In general, however, models need not necessarily consist of assertions which we can regard as clauses of a somewhat special kind.

Given that the learning system can invoke two (or more) functionally similar models, it is important that we have a way of stating that two (sub)systems behave in a similar way, or that they differ. This can be done very conveniently in the metalanguage using clauses containing the metapredicate  $\text{do}(\dots)$ , and so we are able to define what a particular system can (i.e., is able to) or cannot do. We think that knowledge of this kind is useful not only to an external observer, but also to the teacher and the learner.

## MODELS OF TEACHING AND LEARNING

In order to construct a realistic model of teaching and learning, we shall endow both the teaching and the learning system with certain self-knowledge, that is knowledge about what various (sub)systems can do in relation to one another. This knowledge is useful in various ways: It helps the teaching system to assess the abilities of the learning system, and then decide what kind of training should be provided. This knowledge is also useful to the learning system when it comes to considering which problems could be accepted (or rejected) in accordance with its abilities. Also, the knowledge of one's own shortcomings provides the learning system with a means for assessing whether certain training is effective.

We must accept that the teaching (learning) system will have only limited understanding of the surrounding world which from our point of view includes both the teaching and the learning system. The success of

teaching will be dependent on how reliable the knowledge of this world is, and this will depend on:

- the knowledge of the current epistemological state of the LS
- the knowledge of how the LS will react.

Certain actions of the teaching system will be directed at obtaining more information about the learning system. This process may be compared to perception of physical objects in the real world. The teaching system is not concerned so much with the physical properties of the perceived objects, but rather with the inner epistemological state of the learner.

The learning system is, however, an active system, and so here the analogy with the perception of passive physical objects ends. The learning system may react in accordance with its own aims and rules. It has been recognized that questions, too, convey certain meaning to the hearer. This meaning can be derived from certain "presuppositions" which the questioner assumes to be true [62]. The learner may thus consider what the question implies, and if it is not in accord with his own beliefs, reject the question — for example this way: "Why do you ask me all these questions? I know it already!" It would be interesting to see how such an interchange could be represented in our model.

The main task of the teaching system will be to provide new information to the learning system. Making an entire copy of the teaching system's knowledge base is, however, not seen as a practical proposition. We think that the teaching system will develop — as most software products do — by responding to the needs of the environment and accumulating new knowledge. Teaching can be seen then as a problem of selecting a portion of one's own knowledge with the objective of providing the learning system with a way of responding to new requirements.

A great deal of work is still ahead of us, however, as we do not only want to be able to represent the individual communicative acts between the teaching and learning systems, but also model a rich variety of different interactions which arise in teaching and learning. In order to proceed with this work, we believe, we cannot consider teaching and learning in separation, but rather create a model which includes both systems and endow them with certain abilities of self-reflection. We also need to create an environment which permits each of the systems to verify its own hypotheses, and which may also be altered as a result of purposeful actions. For us the new approach provides us with a more adequate workbench which we intend to use for refining our own hypotheses concerning learning.

# **SOME APPROACHES TO KNOWLEDGE ACQUISITION**

## **Bruce G. Buchanan**

Stanford University, Department of Computer Science  
Stanford, CA, 94305

### **ABSTRACT**

Knowledge acquisition is not a single, monolithic problem for AI. There are many ways to approach the topic in order to understand issues and design useful tools for constructing knowledge-based systems. Several of those approaches are being explored in the Knowledge Systems Laboratory (KSL) at Stanford.

### **BACKGROUND**

In 1969, while working on DENDRAL, we recognized the "bottleneck" problem of acquiring knowledge from experts for use by a knowledge-based system [44]. From that initial recognition, born out of our first efforts at systematizing the process now known as "knowledge engineering", developed a line of research that is still active at Stanford. In the context of DENDRAL, we first initiated research on interactive editors and automatic rule induction [217]. Then, in the context of MYCIN, we were again confronted with very practical problems of knowledge engineering, and further worked on interactive debugging tools and languages for expressing new knowledge [48].

To date, knowledge engineering has been the only means of building a complex knowledge base, but this remains a tedious process. Thus, we are seeking to develop tools that aid knowledge engineers. Studies in progress include exploring methods by which programs can acquire knowledge by induction from examples, by analogy, by watching, by SOAR's process of chunking, by discovery, and by understanding written text. In this brief overview, we summarize research recently completed or in progress in the KSL. Although we are developing programs in the context of particular, suitable domains, we are seeking methods that are domain independent. Thus some of our research has resulted in papers that analyze and discuss general problems [45, 98, 97, 15].

## GENERAL MODEL

Knowledge acquisition cannot be thought of as a single problem; there are several dimensions to the transfer and transformation of problem-solving expertise from a human expert or other knowledge source into a program. In our research, we have identified three different stages of knowledge acquisition and are examining different kinds of learning appropriate to each stage. We have given the chess labels of the "opening," "middle game," and "end game" to these stages (see Chapter 5 of [47] ). All three can be seen as different perspectives on the general model for learning systems in [45].

In the opening, an expert must lay out the terminology and the problem-solving framework. All subsequent knowledge-acquisition work depends on making this conceptual foundation correct. The middle game builds on the framework that was established initially. In a rule-based system, a specialist provides a large block of rules to cover many cases. In the end game, the knowledge base is refined by applying it to test cases. The TEIRESIAS program [82], is perhaps the best known piece of research on this refinement stage.

The approaches to automating knowledge acquisition described below contribute mainly to the middle- and end-game stages. However, many ideas about the opening stage were recently codified in an experimental system called ROGET [18], which is an EMYCIN-based expert system whose domain is knowledge engineering. ROGET carries on a dialogue with an expert, much as a knowledge engineer does, in order to formulate the working vocabulary and organization for the knowledge base.

## CURRENT WORK

### **Knowledge Engineering**

We believe it is important for the immediate future to design tools that help experts build knowledge bases for expert systems. These include "smart" editing and debugging tools, as mentioned above, as well as systems that analyze the contents of an emerging knowledge base [366].

One interactive system we are building, called MARCK [144], learns control knowledge for PROTEAN [158]. When an expert is running PROTEAN interactively and overrides PROTEAN's choice of tasks in favor of another task, MARCK initiates a dialog with the expert to understand the reasons for the override. By examining the differences between the two tasks, MARCK is able to suggest new guidance heuristics that will

cause PROTEAN to execute the expert's preferred task in similar situations in the future.

Another interactive system, called OPAL [99], aids in the definition of new knowledge for the ONCOCIN program, an advisor for management of cancer therapy [333]. OPAL exploits the interactive graphics capabilities of Lisp workstations in order to present a specialist with easily read forms to be filled out. Considerable attention has been paid to human engineering, for example, in the use of menus to minimize the need for typing. Because OPAL has considerable knowledge about some types of cancer and about many different drugs, as well as knowledge of treatment plans in general, it can do more than a knowledge engineer who lacks these kinds of details.

### **Learning from Examples**

Induction can be an important method of acquiring new knowledge when libraries of previously solved cases already exist. Because such learning is itself a knowledge-based activity, knowledge about how to learn from examples can be expressed in much the same way as other problem-solving knowledge [217]. We are currently investigating methods of induction in the context of learning rules and meta-rules for diagnosing cases of jaundice [126, 127]. The program, called RL, uses a rough model, or half-order theory, of the domain in order to guide a systematic search through a space of plausible concept definitions and associations.

An important problem in theory formation is interpreting observed data in the first place. In the case that an emerging, partially formed theory is used to interpret the data, there is ample opportunity for erroneous extensions to the theory that is being developed to explain the data. We have defined a method for "theory-driven data interpretation" that propagates constraints in order to determine a consistent interpretation of the data. This has been implemented in a program called PRE [94].

### **Learning by Analogy**

Analogical reasoning is not only a powerful problem-solving method but also an important method for constructing knowledge bases. Once one knowledge base is built and refined for an expert system, we should be able to use it as a stepping stone for building other, analogous knowledge bases. There are two main topics to explore: finding appropriate analogies and using those analogies appropriately.

Currently, our focus is on the appropriate use of analogies. We are

seeking to acquire problem-solving knowledge in the domain of electrical circuits by exploiting plausible analogies to facts and relations about fluid flow. Given a test problem about a circuit and a statement such as "voltage is like fluid pressure," the problem solver, called NLAG [133], creates new relations in the electrical domain to solve the problem. Of the many new relations it explores, it will save those that allowed the test problem to be solved.

### **Learning by Watching**

Learning by watching is a form of learning from examples. The learning program is, in effect, an apprentice, watching the problem-solving behavior of a specialist, much as a medical intern observes and interacts with the chief resident. We are investigating the extent to which this learning method can automate the transfer of expertise in building expert systems.

The program we are developing, called ODYSSEUS [393], has several stages of operation. First it must induce the rule and frame knowledge for the system. Using this initial knowledge base as a half-order theory, the program then attempts to infer how the specialist reasons. It does this by enumerating and ranking various plausible reasoning paths that can explain the specialist's problem-solving behavior. When it can not find a path of reasoning that corresponds to the specialist's procedure, the program assumes that it is lacking either strategic or domain knowledge. It can attempt to acquire the missing knowledge automatically or by asking specific questions of the expert.

ODYSSEUS can assist the transfer of expertise for both learning and tutoring. In the HERACLES system (a generalized version of NEOMYCIN [71]), it serves as the knowledge acquisition subsystem by inferring a model of expert behavior. In GUIDON2, the same modeling system is used to infer the model of a novice's behavior.

### **Learning by Chunking**

What we call learning by chunking [304] represents a somewhat different approach to knowledge acquisition than the others described here. Often in the process of solving a problem, we find a way to combine several steps into a single operation, or "chunk," thus improving performance and efficiency. A chunk can be represented as a rule that identifies which aspects of the task were useful in reaching the desired goal. In the context of SOAR, we are developing ways in which a general

problem-solving system can use chunking to improve all aspects of its performance.

### **Learning by Discovery**

Lenat's thesis on the AM program [210], defines a paradigm of learning by discovery. That program begins with an initial set of basic concepts about a subject area, elementary set theory, and explores the space of new concepts that can be conjectured on the basis of the old ones. Its exploration is guided by numerous heuristics that define the "interestingness" of new concepts formed in various ways from old ones. AM was generalized into the program EURISKO [212].

The Molgen project is currently exploring machine discovery in the context of regulatory molecular genetics. The discovery system will attempt to improve the performance of a qualitative simulation program which predicts the results of experiments in molecular genetics. When incorrect predictions are made the discovery system will suggest modifications to the simulator's theory of molecular genetics. Credit assignment will be accomplished by allowing the discovery system to determine the validity of some of the more observable intermediate states of the simulation, and by giving it predictions for a number of very similar experiments. New terms will be introduced into the theory to account for otherwise unexplainable effects. And, because these generated objects will be described very abstractly when first introduced, the representation of their behavior must be successively refined through the analysis of subsequent experiments. Through subsequent experimentation, the system will modify and verify its emerging theory of gene regulation.

### **Learning from Written Text**

One important source of knowledge is published articles. We have implemented a prototype version of a knowledge acquisition program, called REFEREE [135], that can use this knowledge source more directly. It interacts with an informed (though nonexpert) reader, thus postponing problems of a program's understanding unconstrained English. REFEREE critiques the design and execution of a reported clinical trial on a drug, for example inferring how well the subjects were randomized.

The critique represents a justification for why someone should (or should not) believe the conclusions of the paper. From that justification, then the program can find the new knowledge to be concluded from the study. In the future we will also use the justification to integrate new

reports that come from several sources and that may be in conflict with each other.

## **CURRENT STATUS**

Prototype programs in most of these areas of knowledge acquisition are essentially complete. Active research continues in these approaches, and other approaches are being assessed continually.

# **ANALOGICAL LEARNING WITH MULTIPLE MODELS**

**Mark H. Burstein**

Bolt Beranek and Newman Inc.  
Cambridge, Ma.

## **ABSTRACT**

When students are learning from multiple analogies, they must relate and integrate several analogical models, in a fashion consistent with what they already know from examples in the domain they are studying. This analogical combination process often involves relating causal or structural models at several different levels of abstraction. Debugging such composite conceptual models thus requires attention to mappings between levels, as well as analogical mappings at a single level. We are studying the processes involved in this kind of learning by a combination of protocol analysis and the construction of computer models.

## **INTRODUCTION**

In prior papers, we have reported on the use of multiple analogical models in learning about a new domain [54, 53, 78, 79, 363]. Burstein showed how students used three very different analogies to construct a composite model of how assignment works in the BASIC programming language, and developed a computer simulation of that behavior. Collins, Gentner and Stevens have developed a characterization of the interactions between partial analogical models of evaporation processes. Both of these studies showed analogies interacting at *multiple levels of abstraction*. For each of the analogical systems considered, one analogy provided an initial model of some central level of abstraction. Other analogies, making independent contributions at other levels of abstraction, were then "tied" to the first model by analogical associations at a level at which the two systems overlapped. Debugging inconsistencies in such systems of analogies should be facilitated by the consideration of examples or problems calling for inferences relating terms of different levels of abstraction.

## BACKGROUND

Burstein [54, 53] examined how three analogies can play complementary roles in the formation of a composite conceptual model of variables and assignment for the programming language BASIC. Simply stated, the three analogies were:

- A variable is like a box. The computer stores numbers in variables.
- Assignment is like algebraic equality. Typing 'X=5' assigns the number 5 to the variable X.
- Computers have memory. They remember the values assigned to variables.

CARL, a computer model of causally-based analogical learning, incrementally relates these three analogies to form a conceptual model of the behavior and use of assignment statements.

Each analogy is represented in the target model at several levels of description. The target model includes a contextual or abstract plan level, a causal level, and a lexical representation level. Each of the analogies provides some portion of the final model not derivable from the other analogies. The three are tied together by cross-domain correspondences between the roles of objects and relations in versions of the central, causal level model. The box analogy provides the initial causal model of assignment, along with some potentially useful plans involving the temporary storage of objects. The algebra domain provides knowledge of numbers, the operations that can be performed on them, the symbols for representing those operations, and the rules for interpreting them. The algebraic equality analogy is related to the mechanistic "box" model by an association between the relation resulting from "putting" a number in a box, and the inferred relation between an algebraic variable and its value that results from an analysis of an equality statement. The analogy to human memory is active primarily at the planning level, providing a context for many of the operations that computers can perform. It plays a role in many students' early models of a number of computer commands, especially input and output functions, which can be viewed analogically as statement assimilation and question answering, respectively.

Collins and Gentner [78, 79] have also developed a multi-level decomposition of three analogical models and their variations that they use to explain subjects answers to novel questions about evaporation. One combination of three analogies they considered was:

- (Billiards) Molecules are like billiard balls bouncing in space. The

warmer the water (or air) is, the more velocity the average molecule has.

- (Rocket ship) A molecule escaping from the water is like a rocket ship escaping from the earth. A narrow range of velocities and directions is required for escape.
- (Crowded room) Molecules in an air mass are like people in a room. As more molecules collect in the air mass, the room becomes more crowded and harder to enter. When the room is very crowded, some molecules may be forced out. Warmer air masses are larger and hence less dense than colder ones.

These three analogies cover different aspects of the evaporation process. The billiard analogy is used to model the air and water. The rocket-ship analogy models the escape of molecules to the air. The crowded room analogy refines the model of the air, and models the return of molecules to the water. In [79], the contributions of each of these analogies to subjects reasoning about evaporation processes was examined at three levels of abstraction; a macroscopic functional level, an aggregate microscopic level, and a molecular unit level. The billiard analogy provides the model for the intermediate, aggregate molecular level of description. The rocket ship analogy uses the lower molecular unit level to model the escape of a single molecule, and must be generalized to be related to the aggregate level. The crowded room analogy relates macroscopic properties like temperature to a refined aggregate level model of the air space. The macroscopic and aggregate levels were represented in terms of Forbus's Qualitative Process Theory [123], as a series of qualitative proportionalities. The molecular level, describing interactions between individual particles, was represented using the incremental qualitative analysis of deKleer [91] and Forbus [122].

Collins and Gentner found that each dependency in a macroscopic level model was supported by one or more aggregate level models, and that each aggregate model was supported by at least one molecular level model. For example, temperature and vaporization rate at the macroscopic level were related to density and average molecular velocity at the aggregate microscopic level, which in turn affected the rate of achievement of "escape velocity" as modeled at the aggregate and molecular unit levels. These analogical models were related to each other at the aggregate level, as primarily described by the billiard model.

People apparently vary greatly in the degree to which they connect these component models into a consistent whole. At one extreme, the analogies may be maintained largely unintegrated, each covering only a small piece of the target domain, and each somewhat inconsistent with the

others. At the other extreme, some people appear to have connected their component models of evaporation into a consistent overall model, enabling them to make more predictions about how the overall target system will behave.

## DEBUGGING MULTI-LEVEL COMPOSITE MODELS

In both of these studies, several analogies were related together by correspondence mappings at a central level of abstraction. One of the analogies involved was mapped freely to form the basic model for this central level, and others were "tied" to that initial model at one level, but made their major contributions to the composite model by mappings at an adjacent level. We are attempting to characterize the reasoning processes by which analogies focused at different levels of abstraction generate predictions at other levels, potentially resulting in more closely unified models.

Integrating and debugging several related analogies involves testing the developing target model on concrete problems that reveal conflicting predictions of two analogies at a single level of abstraction. If, as appears to be the case, different analogies in composite systems are primarily used to characterize different levels of abstraction, then problems that call for novel cross-level predictions will reveal the most inconsistencies, and debugging these inconsistencies will lead to more refined, well integrated domain models.

# THE WORLD MODELERS PROJECT: OBJECTIVES AND SIMULATOR ARCHITECTURE

Jaime Carbonell and Greg Hood

Carnegie-Mellon University, Department of Computer Science  
Pittsburgh, PA 15213

## WHAT IS WORLD MODELING?

Machine learning has long sought to construct complete, autonomous learning systems that start with general inference rules and learning techniques, and gradually acquire complex skills and knowledge through continuous interaction with an information-rich external environment [59]. The World Modelers project provides a simplified artificial environment -- a continuous three-dimensional physical model of the world -- to facilitate the design, implementation and testing of integrated learning systems. Thus, the world simulator can be viewed as a tool that enables researchers to investigate significant intermediate steps between the single-function learning systems of today (systems largely incapable of multi-task learning, attention focusing, experiment formulation, etc.) and the real-world autonomous learning robot of tomorrow.

The theoretical tenets underlying our simulated world can be summarized as follows:

- The world must be sufficiently rich and complex such that no organism can fully internalize its function and predict all future events (unlike Becker's world [17]).
- The simulator must obey a set of realistic and consistent laws, although such laws are not known *a priori* by the organisms that populate it. (We chose the laws of Newtonian physics.)
- The organisms should communicate with the world only via sensor and effector interfaces corresponding to sight, touch, etc.
- Different organism architectures, incorporating different learning strategies and different faculties, must be supported for comparative experimentation.
- Software tools for displaying and configuring the world, for

analyzing and suspending simulations, and for other necessary experimental functions, such as re-running a simulation with subtly altered learning rules, must be provided to make the world modeling system a true research tool.

## MACHINE LEARNING OBJECTIVES

In the context of a complex reactive environment, many different learning methods applied to various cognitive tasks can be investigated. First, we discuss the qualitative impact that the presence of a reactive environment can make. Then, we list the specific projects underway. Some of these are presented in greater detail in other papers in this volume.

Machine learning systems have evolved from *tabula rasa* approaches, typified by perceptrons, to isolated concept acquisition from examples, to incremental knowledge-intensive systems [59]. With integrated tools such as the world simulator, we have begun to look at reactive, long-term integrated learning systems, with the following major properties:

- **Reactivity** -- The learner must react to a continuously changing environment, and thus acquire a partial, qualitative, and imperfect conception of the external world, which is gradually refined through experience. Moreover, the learner can design and carry out experiments to improve its knowledge (i.e., by exploration, examination, trial-and-error, imitation, hypothesis-formation from examples, hypothesis-testing by attempting to disprove the least-likely predictions of a new hypothesis, etc.)
- **Continuity** -- The learner will have a long-term existence in the simulated world, much like a biological organism or a robot -- and unlike a present-day AI program that is turned off after running a few minutes and acquiring the desired concept. As such, learning can be *gradual, incremental*, and subject to periodic *refinement* from external feedback. Concepts and skills acquired should be available as the building blocks to learn newer, more complex concepts and skills, as advocated in the derivational analogy method [60].
- **Generality** -- The learner has many tasks to perform, and much knowledge to acquire, thus presenting a golden opportunity to test out the hypothesis that a unified learning method may serve to acquire such diverse cognitive functions such as

communication through natural language, goal-directed planning, and lower level tasks such as object and event perception, navigation, etc. The diversity of tasks prevents one from building the knowledge to acquired into the learning mechanism itself, however subtly or unwittingly.

- **Intentionality** -- The learner is an autonomous simulated organism subject to the rules of the world simulation in which it is embedded. We have chosen to endow our organisms with a set of physiological drives and other goals (such as self-preservation and curiosity) that focus its behavior. This creates a crucial bias in the knowledge and skills the organism chooses to devote attention to acquiring -- namely, those that further its own ends. Such a focus is needed to simulate most learning tasks. (Why does a lion learn where prey is likely to be found, but does not bother to theorize about the origin of geological formations in its environment?) Of course, the actual goals, and relative-importance relations among the goals, given to any organism are determined by the experimenter. In several of our investigations we endow organisms with diffuse goals such as curiosity, imitativeness, wanderlust, etc., in order to explore less focused learning behavior.

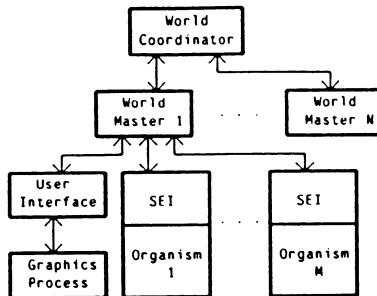
Our first steps using the world simulator involve several learning and related performance tasks, some of which are reported elsewhere in these proceedings. In order to learn from experience, whether it be the learner's own experience or the observed actions of a wiser individual, we must have a general method of perceiving, encoding, structuring, and retrieving events of different degrees of complexity. For such a purpose, we developed an event memory method reported in [263]. We are investigating a unified cognitive architecture using this event memory [Mozer (this volume)] in which we are building the first prototype versions of a first-words language acquisition module and a simple plan-acquisition module that formulates and acquires plans to handle recurring goals (such as learning to squirrel away food if it is available now, but may not be available later when the organism extrapolates it is likely to get hungry). We are also investigating alternative architectures for lower-level learning at the neural modeling stage [Hood (this volume)], and for investigating more "goal-free" initial learning of infants imitating their elders, which gradually transitions into more goal-directed behavior [Rappaport (this volume)].

In order to investigate knowledge-rich learning in humans or in artificial autonomous devices, we must bear in mind that existing knowledge

guides learning, and in particular that knowledge acquired in one domain facilitates learning in a related domain. For instance, in acquiring communicative skills (such as language), the learner must refer to objects, actions and concepts already acquired or in the process of being acquired. Moreover, focusing on the relevant part of the world denoted by an incoming utterance is integrally tied to the present task in which the organism is engaged. Therefore, having a unified cognitive architecture and general learning methods, as discussed above, is a central aspect of our project.

## THE WORLD SIMULATOR ARCHITECTURE

The World Modeling System consists of several cooperating processes, most of which run on a VAX<sup>1</sup>. Figure 1 shows the various types of processes and the communication paths in the system. Many simulations may be running concurrently, each of which may contain multiple organisms and user interfaces. The purpose of the World Coordinator is to keep track of all simulations which are currently running and to allow a user to start a new simulation or enter into a running one.



**Figure 1:** Processes in the World Modeling System

For each simulation there is one World Master. The World Master contains the definitive copy of the simulated world, i.e., all objects and their physical properties. It is responsible for updating the world every time step according to the physical laws. For example, if a force is applied to an object, the World Master will update the object's position and velocity. It detects any collisions among objects and adjusts positions and velocities accordingly. The World Master also handles all requests for

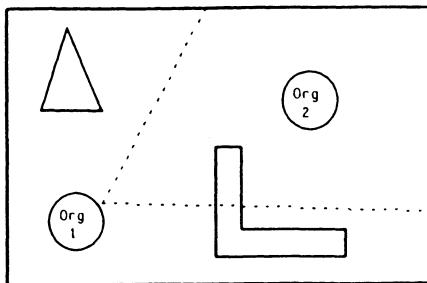
---

<sup>1</sup>VAX is a trademark of Digital Equipment Corp.

changes to the world data structure. After the new state of the world has been computed, the changes are sent to all subprocesses so that they may make corresponding changes in their copy of the world data structure.

For every organism in the world there is a SEI/Organism process. The SEI (Sensory-Effector Interface) is a set of routines which the organism may call to receive sensory information or to produce actions in the world. The SEI contains a local copy of the world data structure and from this it generates the sensory information which the organism can perceive. This may include visible object properties (e.g. color), sounds, smells, tastes, and forces applied to the organism. It also takes actions (such as move forward 1 meter) and translates these into change requests which are then sent to the World Master. The organism code may be written in any of several languages including Franz Lisp, C, and Prism (a production system language [195]).

The User Interface process allows the user to monitor the progress of the simulation. The user may start and stop the simulation and may add, remove, or modify objects in the simulated world. A separate process, but closely associated with the User Interface, is the Graphics process. This process runs on a graphics workstation (currently a Perq-II<sup>2</sup>). It can produce a graphical view of the world as seen from an arbitrary point in space. An example of such a view is shown in Figure 2. This world contains two organisms and two inanimate objects; one organism is within the other organism's cone of vision. The Graphics process may also display the world as seen from the viewpoint of an organism so the user can see what objects are visible to it. The package has the capability of generating perspective shaded graphics with hidden surface elimination as well as simple line drawings.



**Figure 2:** An Example View of the World

---

<sup>2</sup>Perq is a trademark of Perq Systems Corp.

To provide some insight into how the system functions and the nature of the inter-process communication which takes place, we present a simplified sequence of events which happen during the simulation of one time step:

1. The World Master computes the new state of the world, including new positions for all objects.
2. The World Master sends a set of update commands to all subprocesses, i.e., all User Interfaces and Organisms in that simulated world.
3. Each SEI receives these updates and computes the new sensory data which is perceivable by its corresponding Organism.
4. Each Organism uses this sensory information, and has the opportunity to "think" over the new information, along with any internal state that it has saved. As a result, it may perform some actions by calling appropriate SEI routines.
5. The SEI converts these actions into a change request protocol which it then sends to the World Master.
6. After all SEI's and User Interfaces have finished sending change requests and have indicated that they are ready to advance to the next time step, the entire cycle repeats starting with Step 1.

## ACKNOWLEDGMENTS

We would like to thank the other members of the World Modelers Group (Keith Barnett, Klaus Gross, Pat Langley, Mike Mozer, Alain Rappaport, and Hans Tallis) for their ideas and comments on this paper. This research was supported in part by ONR grants N00014-79-C-0661 and N0014-82-C-50767, and DARPA contract number F33615-84-K-1520.

# THE ACQUISITION OF PROCEDURAL KNOWLEDGE THROUGH INDUCTIVE LEARNING

## Kaihu Chen

Artificial Intelligence Laboratory, Department of Computer Science  
University of Illinois at Urbana-Champaign

### ABSTRACT

Most induction systems previously developed are concerned with the acquisition of generalized characteristic descriptions from training instances represented in either attribute tuples or first order predicate (or their variations). An extension to the problem can be achieved by allowing each training instance to take the form of a sequence of descriptions, which may represent an instance of an operation sequence, or the description of a temporal event.

### INTRODUCTION

Procedural knowledge, where the temporal relationship between the knowledge components are important, plays important roles in many AI systems. Two types of temporal information can be identified: temporal ordering and time interval. Temporal ordering is concerned the order of occurrence of two events/observations, while time interval indicates the time lapse between the occurrence of two events/observations. To simplify the problem we shall discuss the type of procedural knowledge where only the *temporal ordering* between knowledge components are important.

Our aim in this paper is to propose a general methodology for the inductive acquisition of procedural knowledge through observations of multiple training instances. The procedural knowledge thus acquired could represent the sequence of operations that achieve a certain goal, hidden causal relations, or simply recurring patterns in the observations.

General induction methodologies previously developed such as version space [15] or AQ11 [230], although still valid for the inductive acquisition of procedural knowledge, are much too inefficient for reasons to be presented below.

## BACKGROUND

The type of inductive learning of procedural knowledge we shall discuss is essentially the classical n-classifier problem where classifiers (hypotheses) are generated in order to discriminate instances in one class from all training instance in other classes. However, two problems in procedural domain rendered the previous inductive learning methodologies [15, 230, 235] impractical. The first problem is in the immensely greater problem space introduced by the temporal information. The heuristics of generalizing/specializing any intermediate hypothesis on ground of one positive/negative training instance is too weak because too many useless hypotheses will be generated. Stronger heuristics for the generation of new hypotheses is thus required for the procedural domain. The second problem is in the redundant information introduced by the temporal relation. For example, in the following first order predicate expression which represents a hypothesis for a class of procedural knowledge:

$$\text{next-operation}(A,B) \& \text{next-operation}(B,C) \& \text{next-operation}(A,C)$$

The term *next-operation(A, C)* is redundant given the fact that the predicate *next-operation* is transitive. The number of such redundant terms grow exponentially with the number of operations (the A, B, C etc.) in the hypothesis. Refinements applied to hypotheses with redundancy may either generate inconsistent hypotheses, or otherwise offer no improvement at all. The existence of redundant information in the hypotheses is thus a great evil to be eradicated.

It is assumed that two sets of refinement operators are given, the set of generalization operators  $G$  and the set of specialization operators  $S$ . A hypothesis  $H_1$  is the *generalization* of another hypothesis  $H_2$  if there exists a generalization operator  $g$  in  $G$  such that

$$H_1 = g(H_2)$$

Two hypotheses  $H_1$  and  $H_2$  have a **common generalization**  $cg(H_1, H_2)$  if both  $H_1$  and  $H_2$  can be generalized to  $cg(H_1, H_2)$ . Similarly **specialization** and **common specialization** can be defined.

## CONSTRAINING HYPOTHESIS REFINEMENT

Previous induction methodologies [15, 230, 235] failed to exploit the relationships between competing hypotheses. Each intermediate hypothesis stands on its own for refinement. Neither its relationship to other hypotheses nor its refinement history is considered. It is realized that given the generalization relationship of two hypotheses and their relative efficacy in covering training instances, it is possible to identify more promising refinement strategies. For example, two hypotheses  $H_1$  and  $H_2$  describing the same class, if  $H_1$  and  $H_2$  are equally efficacious<sup>1</sup> in describing the class, and  $H_1$  is more general than  $H_2$ , then  $H_2$  may be eliminated from further refinement.

Similar refinement heuristics based on the relationship of two or more hypotheses can be defined. For example, associations between competing hypotheses can be established so that hypotheses with higher degree of association can be specialized to a common specialization to form a new hypothesis. Competing hypotheses with high degree of syntactical similarity can also be generalized to a common generalization.

## CANONICAL REPRESENTATION

The canonical representation for a hypothesis can be defined as its alternative equivalent representation that satisfies certain criteria. For example, given a hypothesis  $H$  and the set of its equivalent alternative representations  $H'$ , we can define the canonical representation of  $H$  as the representation in  $H'$  that has the least number of terms.

Canonical representations contain the least amount of redundancy and thus is the most efficient hypotheses to be induced with. The key in using canonical representation is to define specialization/generalization operators in such a way so that only canonical hypotheses are ever generated. Canonical representation is also useful for any relations that are transitive.

<sup>1</sup>Various functions can be used to define the efficacy of a hypothesis in describing a class. The function  $(Np-Nn)/(Np+Nn)$  gives a value of one or minus one when the hypothesis covers all positive/negative instances, where  $Np/Nn$  represents the number of positive/negative instances covered by the hypothesis. The entropy function

$$-Pp \cdot \log(Pp) - Pn \cdot \log(Pn)$$

where  $Pp = Np/(Np+Nn)$  and  $Pn = Nn/(Np+Nn)$ , is also a good measurement.

## CONCLUSION

It is realized that previous induction methodologies conduct **memoriless refinement** where refinements were applied to intermediate hypotheses disregarding their history of refinements and their relationship to other hypotheses. Exploiting the dependencies between competing hypotheses allows us to have more control on the generation of promising hypotheses. The canonicalization of representation and refinement operators helps to alleviate the problem of combinatorial explosion and avoid generating inconsistent hypotheses.

The methodology presented in this paper can be used to discover causal relations or recurring patterns from multiple training instances for the purpose of diagnosis or prediction. It can also be used for the acquisition of operation sequences that achieve certain goal.

## ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grant NSF DCR 84-06801 and by the Office of Naval Research under grant no. N00014-82-K-0186.

# **LEARNING STATIC EVALUATION FUNCTIONS BY LINEAR REGRESSION**

**Jens Christensen**

Department of Computer Science, Columbia University  
New York, NY 10027

## **ABSTRACT**

We present a technique for learning the coefficients of a linear static evaluation function for two-person games based on playing experience. This is accomplished by using linear regression to modify the coefficients based on the difference between the static evaluation of a state and the value returned by a mini-max look-ahead search. In an initial experiment, the technique was used to learn relative weights for the different chess pieces.

## **INTRODUCTION AND PREVIOUS WORK**

We adopt the standard game-playing model of mini-max search with static evaluation at the search frontier [328]. While other learning experiments have focused on openings or endgames [21, 243, 282], we have addressed the mid-game. Samuel [312] observed that the most effective way of improving mid-game performance is to modify the evaluation function.

The first game program to improve its performance by learning was Samuel's checkers program [312]. Although it also employed other learning techniques, it is mostly known for learning the coefficients in its polynomial evaluation function. Samuel's idea was that the difference between the static evaluation of a board position and the backed-up value derived from a mini-max search could be used to modify the evaluation function. This is based on the assumption that for a given evaluation function, values based on looking ahead are more accurate than purely static evaluations. Samuel's program altered the coefficients of the function at every move where there was a significant difference between the value calculated by the evaluation function and that returned by the mini-max search. The idea is to alter the evaluation function so that it can calculate the backed-up value at the original state without having to look ahead. An ideal evaluation function eliminates the need for a mini-max search since it always calculates the correct value for any state.

The main difference between our approach and Samuel's is in how

the value returned by the mini-max search is used to modify the evaluation function. Samuel employed an ad-hoc technique based on correlation coefficients and somewhat arbitrary correction factors. Our method is based on the well-understood technique of linear regression. A secondary difference is that while his investigation focused on checkers, our experiments have been carried out in the more complex game of chess.

## COEFFICIENT MODIFICATION BY REGRESSION

For pedagogical reasons, we will explain the technique using the simple example of a checkers evaluation function based only on the numbers of single pieces and kings. In other words, we want to determine the relative value of the kings and pieces in an evaluation function of the form  $C_1F_1 + C_2F_2$  where  $F_1$  and  $F_2$  are the numbers of pieces and kings, respectively. Of course, there would also be terms for the opponent's material, but we assume that the coefficients have the same magnitude and opposite signs.

We start with an initial estimate of the coefficients, e.g. both equal to one. Given a particular board position, we can plug in values for  $F_1$  and  $F_2$ . Then, we perform a look-ahead search to some depth, evaluate the nodes at the frontier using the initial estimate of the coefficients, and back-up these values using the mini-max algorithm, resulting in a numerical value for the original position. This information can be represented as an equation of the form  $C_1F_1 + C_2F_2 = R$ , where the  $C_i$  are the parameters of the equation, the  $F_i$  are the factors of the evaluation function or dependent variables, and the  $R$  is the backed-up mini-max value. One can then perform a linear regression on this data to determine the best-fitting values for the parameters of the equation, thus in effect establishing the coefficients of the factors in the evaluation function.

Unfortunately, the result of the regression is not the best choice of coefficients but rather a better estimate. The reason is that the right-hand sides of the equations are not exact but approximate values since they are based on the same estimated coefficients. Thus, the entire process must be repeated using the new coefficients derived from the regression. These iterations are continued until the values converge.

This iterative algorithm can be viewed as hill-climbing in the space of coefficients, with potentially all the normally associated problems of hill-climbing. In particular, there may exist values which are locally stable but not globally optimal. No effective way exists to detect such local stabilities except by drastically altering some of the coefficients in the regression analysis to see if different maxima are encountered. If that is the

case, then these different evaluation functions can be played against each other to see which one is indeed the best.

This learning method can be applied to any game which can be implemented using mini-max search with static evaluation. Note that the learning is accomplished simply by the program playing games against itself, without any outside input.

## EXPERIMENTS WITH CHESS

Our method was first explored in the simple game of 4x4x4 tic-tac-toe, and performed remarkably well. We used six factors in the evaluation function, namely the number of rows, columns, and diagonals in which a side could win with either one, two, or three pieces of the same color already in place. Not only did it correctly order factors of the evaluation function in terms of importance, but it also quickly recognized which coefficients had incorrect signs and reversed them.

As a serious test, we chose the game of chess and a simple evaluation function consisting only of material advantage. The experiment was to see if the learning program would approximate the classically accepted weights for the pieces: 9 for the queen, 5 for the rook, 3 for the bishop, 3 for the knight, and 1 for the pawn. The chess program was implemented using a two-ply (one full move) mini-max search with alpha-beta pruning and quiescence. 1400 half-moves were made between each regression. If neither side won during a game it was stopped after 100 half-moves and a new game was started. For purposes of the experiment, a win was assigned one more than the total initial material value and the individual piece values were rounded off to the nearest 0.5. The pieces stabilized at: Queen, 8; rook, 4; bishop, 4; knight, 3; pawn, 2.

The above results were based on a search of only two ply, but using quiescence. This means that the chess program was playing a tactical game, trying to maximize material in the short run rather than to achieve checkmate. Since the equations correspond to moves from every phase of the game, the final values are average weights from the opening, midgame, and endgame. Berliner has observed, however, that the optimal evaluation function is in general a function of the stage of the game [20]. Because of the weakness in the end game caused by the lack of planning the chess program could not take advantage of the rook's increased strength during the end game. Other pieces might suffer from similar effects.

When we played the derived function against the classical function in one hundred games, the derived function won seventeen games and lost sixteen. The rest were draws. This does not mean that our derived

function is optimal, only that it is at least as good as the established one in our program with its shallow search and an evaluation function which only takes material into account.

## CONCLUSIONS

The main contribution of this work is the idea of using linear regression to learn static evaluation functions. A secondary contribution is the application of parameter learning to a game as complex as chess. Our initial experiments suggest that the technique is powerful enough to learn at least the relative weights of the chess pieces. Further work will determine what level of play such a learning machine can ultimately aspire to.

## ACKNOWLEDGMENTS

I wish to acknowledge Richard Korf whose guidance and encouragement has been indispensable throughout this research and especially for his help in all phases of writing this paper. This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165, and by the National Science Foundation under grant IST-84-18879.

# **PLAN INVENTION AND PLAN TRANSFORMATION**

## **Gregg C. Collins**

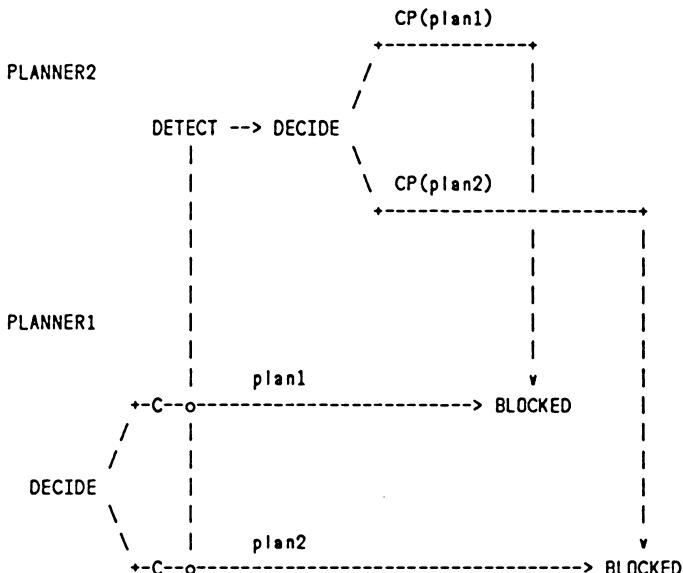
Department of Computer Science, Yale University  
P.O. Box 2158 Yale Station, New Haven CT 06520

Most work on planning in A.I. is directed at the problem of how to manage the application of previously specified plans in familiar domains. However, an important aspect of planning behavior in general is the ability to plan intelligently in novel domains. The first step towards accounting for this behavior must be to characterize the process whereby general planning knowledge can be used to construct new plans in a specific domain. In my thesis I propose a model of such a process. This model assumes that the planner possess some plans which are appropriate for the domain in question, but that the current set is by no means adequate to operate effectively in the domain. The model I propose monitors the execution of these known plans and creates new plans in response to specific problems.

The basis of the model is the observation that plans in different domains can often be viewed as instances of the same strategy. For example, the "option play" in football is designed around the strategy of forcing a defensive player to commit himself to one of two mutually exclusive defensive plans, then choosing the offensive plan on the basis of the choice the defender makes. The same basic strategy is found in other sports and games, including chess, where the "fork", at least in some manifestations, corresponds, or in tic-tac-toe, where this strategy arises as the attempt to produce two lines of attack so that the opponent can only succeed in blocking one. However, the strategy of forcing an opponent to commit himself first, then responding accordingly, is also a well-known strategy among people who frequently have to negotiate, for example. In fact, it is hard to imagine a domain in which competition occurs where this strategy would not be potentially applicable.

Given that plans in diverse domains are instances of the same general strategy, and that known general strategies are potentially applicable to novel planning domains, a method of inventing new plans in novel domains suggests itself: represent and index knowledge about strategies in such a way that a strategy can be recalled by the planner when the type of situation in which it would be useful arises, and use the strategy as a blueprint to construct a new plan which fits the situation. My thesis is an effort to implement such an approach, and to catalogue some of the high-level strategies that human planners appear to possess.

In my approach, each strategy is represented by a memory structure which can be seen as a TOP (Thematic Organization Packet) [319]. Functionally, each of these structures takes the form of an annotated transformation rule. The antecedent of the rule is an abstract description of the planning situation to which the strategy applies. The consequent of the rule is a reorganization of the situation which mitigates the problem inherent in the situation. For example, the "option" strategy mentioned above is intended to apply to a situation which can be represented as follows:



Planner one faces a decision between plan. At some point after making this decision, planner one becomes committed to the chosen course of action – that is, he can no longer reverse the decision and pursue a different option (this point is marked by a C in the above diagram). After this point the option planner one has chosen is detected by planner 2, who uses the information he has derived to make a decision between counterplans. When the correct counterplan is chosen, it successfully blocks the corresponding plan. Planner one is thus in a no win situation.

The "option" transformation postpones planner one's decision, and inserts a DETECT of planner two's choice (subsequent to a time when planner two can be shown to be committed to the choice), thus essentially reversing the situation depicted above.

The process in general consists of four steps. First, expectation failures in planning are detected and analyzed to construct characterization of problems with known plans. Second, heuristics are applied to determine if a given problem may be addressable with a known strategy. Third, the situation is matched to the antecedent of a strategy transformation rule, and the rule is used to construct a new plan. Finally, the plan is installed in memory with a note that its genesis was the strategy in question, so that knowledge about the strategy can be used to help debug the plan when it is used.

I have analyzed a variety of examples of plan creation from this perspective, most of which are in the domain of football. I have also written a program which invents – or "reinvents", in the current idiom – some famous football plays via an implementation of the process described above. I believe a number of results of general interest have come out of this work, including:

1. Specification of a vocabulary for representing what we know about planning in general which is adequate to support the kind of reasoning needed to specify problems and strategies in the ways necessitated by this approach.
2. Identification of a number of specific planning strategies and the construction of representational structures for them.
3. A tentative theory of failure explanation aimed at producing useful characterization of planning problems.
4. Some suggestions about how design processes can be applied to the task of creating plans which carry out general strategies.

# A BRIEF OVERVIEW OF EXPLANATORY SCHEMA ACQUISITION

## Gerald DeJong

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign  
Urbana, IL 61801

### ABSTRACT

Explanatory Schema Acquisition is a form of learning by untutored observation. It can be used to automatically acquire problem solving schemata. The goal of this research is to formalize explanatory schema acquisition in a domain-independent fashion. To do so we are comparing explanatory schema acquisition systems implemented for a number of different concrete domains. This paper presents a brief overview of the learning approach. Five other papers in these proceedings, by Mooney, O'Rorke, Rajamony, Segre, and Shavlik, describe the individual systems that we have been working on.

### INTRODUCTION

During the last four years we have been working on learning schemata by generalizing explanations [87, 89]. We call this technique *Explanatory Schema Acquisition*. It is a form of *Explanation Based Learning* and is similar in spirit to the work of Mitchell [254] and to a lesser extent also [243, 249, 342].

Central to this view of learning is the notion of schematic problem solving which holds that problem solving ought not be looked upon as a search task but rather as a knowledge acquisition and organization task. In schematic problem solving, problems are solved by connecting a few known schemata together to achieve a desired effect. The schemata themselves are general, situation-specific problem solving techniques. The more schemata a system possesses, the better its problem solving ability. The major obstacles to problem solving now become 1) how to endow a system with many schemata and 2) how to index schemata efficiently.

## OVERVIEW

The basic learning procedure involves the system observing the behavior of an expert problem solver. As the system observes the expert's behavior it tries to "understand" his actions. By "understand" we mean that the system builds an internal representation in which all of the causal and motivational information is explicitly stated. Thus, the system must justify why each action was possible, how each state came about, and why each volitional action was desired by its agent.

The observed expert may in the course of his problem solving achieve a goal that is beyond the current problem solving capacity of the system. When the system recognizes that this has happened it constructs an *explanation* of how the goal was achieved from its understanding of the expert's actions. An explanation is the portion of the causally complete understood representation concerned with achieving the goal in question. In other words it is a data-dependency graph of causations showing the justification of why the goal is believed to have been achieved.

Next the explanation is generalized into a schema and stored away for future use. The generalization process consists of making a structural copy of the explanation with semantically empty nodes in place of the explanation's action, state, and object nodes. Then all of the constraints necessary to preserve the veracity of the explanation are imposed on this structure. This constrains the structure to have just those features required for observed problem solving technique to have been successful. Action nodes are then resolved to the most general action known that fulfills the required constraints. The resulting generalized structure is filed away in the system's schema library for use in future problem solving and understanding.

Explanatory schema acquisition is similar in flavor to the way STRIPS [119] acquires MACROPS. There two important differences. First strips generalizes objects in a plan but could not generalize the component operators nor did it allow temporal reordering of operators. Explanatory schema acquisition supports temporal and operator generalization in addition to object generalization. Second, STRIPS acquired MACROPS in response to its own planning activities while explanatory schema acquisition acquires new schemata from observing the problem solving behavior of others. This is an important difference because relying on ones own problem solving limits the complexity of learnable concepts. Learning from observing a more intelligent planner does not bound the difficulty of concepts to those that the system could have previously solved.

A major goal of our current research is to construct a general theory of explanatory schema acquisition. We hope to produce a domain-

independent formalization of this learning paradigm through a comparison study of implementations in diverse domains. To this end we have begun explanatory schema acquisition projects in natural language processing, robotics, theorem proving, and physics problem solving which are described elsewhere in these proceedings. Several of these projects are nearing completion. The comparison work has just begun.

## CONCLUSION

There are several important points to note. First, the domains of these systems are comparatively complex and open-ended. The natural language project allows a number of characters each of which is capable of many actions. Many relations are allowed among characters and other objects in the text. The robotic project, which learns concepts associated with robot manipulator assembly tasks, requires only a relatively few arm commands, but these commands can be composed to create very complex action sequences. This together with the system's solid modeling system which permits the system to represent and manipulate novel pieces make this system open-ended as well.

A second point is that the observed behavior need not be that of a goal-directed expert. We introduced explanatory schema acquisition above in the context of observing an expert, but the observation can also illustrate an undesired action, an unintentional side-effect, or a serendipitous accident. It is important that there be a consistent causal explanation for the observation. However, the observed "expert" might only have stumbled upon the solution through some fortuitous accident. Even such accidental or unintentional problem solving can yield useful new schemata.

A third important aspect of this approach is that it solves the feature selection problem in a rather elegant way. The feature selection problem is particularly difficult for similarity-based systems. It consists of selecting the few features which are salient to the definition of a new concept from among a myriad of possible features. In explanatory schema acquisition the explanation of how the goal is achieved immediately specifies which components of the observation are central. Anything that does not explicitly contribute through the data-dependency network to the support of the goal is irrelevant. Once the example is understood, the complexity of feature selection depends on the size of the explanation; it is independent of the total number of features that make up the example.

The price for all of these advantages is that the system must possess a rather sophisticated theory of the world before it can learn anything. The similarity-based systems, on the other hand, work well in

domains in which there is little or no *a priori* domain knowledge.

## ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under grant NSF IST 83-17889 and by the Air Force Office of Scientific Research under grant F49620-82-K-0009.

# THE EG PROJECT: RECENT PROGRESS

## Thomas G. Dietterich

Oregon State University, Department of Computer Science  
Corvallis, OR, 97331

The long term goal of the EG project is to construct, implement, and evaluate a model of scientific inquiry. Figure 1 shows the model of scientific inquiry that we have developed to date.

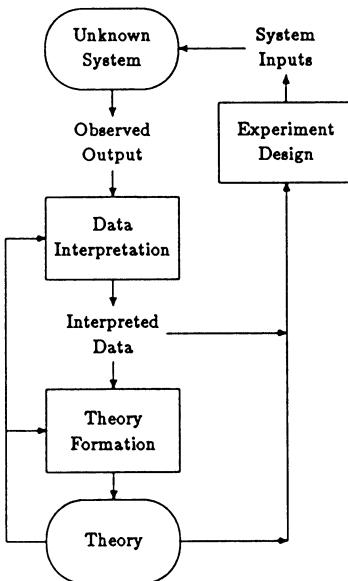


Figure 1: Simple model of scientific inference

According to this model, the goal of a scientist is to construct an accurate theory of an unknown system. There are three main processes that work together to achieve this goal: data interpretation, theory formation, and experiment design. The data interpretation process takes observed behavior of the unknown system (i.e., "raw" data) and applies the current theory (or theories) to develop a set of interpretations of the data. The theory formation process modifies the current theory (or theories) so that they are consistent with this interpreted data. During the data interpretation and theory formation processes, whenever ambiguity is

encountered the scientist faces a choice. He or she may either proceed with data interpretation and theory formation — maintaining alternative interpretations and theories as necessary — or the scientist can design an experiment that would resolve the ambiguity. It is the task of the experiment design process to design and carry out such experiments.

In order to investigate and refine this model, we are implementing it in a specific task domain. The specific learning task is the problem of forming theories about the file system commands of the UNIX operating system. The goal is to develop a computer program that starts with knowledge of two of the UNIX commands (the directory listing command and the command to display the contents of a file) and proceeds, through experimentation and observation, to develop theories of how thirteen different commands work. This task has been discussed in detail elsewhere [95, 94], so we will simply describe how the components of Figure 1 are instantiated in this domain. The unknown system, of course, is the UNIX operating system. It accepts typed text as inputs and produces strings of text as outputs. It also contains state information (e.g., the file system, current working directory, etc.). A theory in this domain consists of 13 procedures — one for each UNIX command. The task of data interpretation is to apply existing theories about some of the commands to infer as much as possible about the current state of UNIX (see below). This can be viewed as applying the current (partial) theory to “parse” the raw data. From the parsing process, EG obtains an input/output pair (training instance) for each command that was executed. The theory formation process takes these input/output pairs and uses them to guide the modification of the current set of theories. Hence, it constitutes a kind of procedure induction from I/O pairs. The experiment design process can be invoked whenever ambiguities arise during data interpretation and theory formation. It designs and executes experiments to gather additional data.

Since the start of this project (in 1983), we have focused our efforts primarily on the process of data interpretation. The data interpretation task can be formally defined by reference to the deductive-nomological (D-N) model of scientific explanation. According to this model, a body of data  $D$  can be *explained* by a theory  $T$  and a set of initial conditions  $C$  if from  $T$  and  $C$  you can deductively infer  $D$ . Data interpretation is the process of finding a set of initial conditions  $C$  given theory  $T$  and data  $D$ . It can be shown that  $C$  is a deductive consequence of  $T$  and  $D$ , so data interpretation can be performed by a deductive process.

For example, suppose that the EG program wants to form a theory of the **rm** command. It might have the following interaction with UNIX in which a directory listing is obtained for the **/csm/eg** directory before and

after the execution of the unknown **rm** command:

```

1: ls /csm/eg
file1
file2
file3
2: rm /csm/eg/file2
3: ls /csm/eg
file1
file3
4:

```

By applying its theory of the directory listing (**ls**) command, EG can make the following inferences based on the first **ls** invocation: (a) the root directory contains a directory named **csm**; (b) the **csm** directory contains a directory named **eg**; (c) the **eg** directory contains three files **file1**, **file2**, and **file3**. Similarly, the second **ls** command can be interpreted to infer that after the **rm**, the **eg** directory contains only two files: **file1** and **file2**. To obtain each of these interpretations, EG takes its given theory  $T$  of **ls** and finds a set of initial conditions  $C$  (i.e., facts about the existence of various files and directories) such that the observed data  $D$  (the printed output of the **ls**) can be explained as a deductive consequence of  $T$  and  $C$ .

Once these interpretations have been constructed, EG has a complete I/O pair for the unknown **rm** command. This I/O pair is passed to the theory formation process, which will construct one or more theories to explain how **file2** could have disappeared.

From this example, we can see that data interpretation plays a role in scientific inquiry analogous to the role of goal regression in planning---both techniques support the incremental development of composite structures (either theories or plans). The fundamental idea behind incremental planning is to "break off" a piece of the overall goal and construct a plan for that piece. Once this is done, goal regression techniques [384] can be applied to "push" the original goal "through" the part of the plan that has already been constructed. This regressed version of the goal specifies the subgoals that remain to be achieved.

In an analogous fashion, data interpretation serves to "push" the observed data through the parts of the theory that have already been developed. The interpreted data then serve to guide further theory formation. This residual theory formation problem is easier to solve than the original task of inferring a theory based on the raw data alone.

Dietterich [94] describes the implemented system PRE that applies constraint propagation techniques to the data interpretation problem. PRE

(Program Reasoning Engine) employs the "deep plans" representation developed in the Programmer's Apprentice project to represent theories as constraint networks. The observed printed output from UNIX is asserted on these networks, and then constraint propagation methods are applied to propagate the outputs *backwards* through the theory to infer possible input states that would have caused the outputs. Hence, PRE solves the data interpretation problem by "backwards execution" of programs. PRE is implemented in Interlisp-D on the Xerox 1108 lisp machine.

Now that we have implemented the data interpretation component of our model, we are now focusing on the experiment design component. Previous work in this area has included a series of "idea" papers that discuss the value of experimentation (e.g., [45, 98, 343]) and a collection of computer programs that perform some kind of instance selection or experimentation (e.g., [169, 213, 256, 282, 330]). We are extending this work in two directions. First, we are investigating experimentation and theory formation in domains where the "unknown system" under study is a system that contains *state information*. Second, in addition to developing effective experiment design methods, we are attempting to understand the space of possible methods and strategies. In particular, we are exploring the ways in which experimentation strategies introduce additional biases into the theory-formation process.

# **LEARNING CAUSAL RELATIONS**

**Richard J. Doyle**

Artificial Intelligence Laboratory, Massachusetts Institute of Technology  
Cambridge, Massachusetts 02139

## **ABSTRACT**

I briefly describe a learning system which constructs causal models of simple physical systems such as cameras, sinks, and toasters. The learning system employs both inductive and deductive inference, making use of knowledge of the forms of causal relations, and of what kinds of causal mechanisms exist.

## **INTRODUCTION**

Causal reasoning has been recognized by AI researchers as an important component of common-sense reasoning. This research addresses the issue of how causal models which support such reasoning can be acquired.

The learning system I am developing uses a hybrid inductive and deductive learning approach. Inductive learning is used to hypothesize causal relations and is supported by general knowledge of the forms of causal relations in terms of functional dependencies between quantities. Deductive learning is used to hypothesize and *verify* causal relations and is supported by knowledge of the kinds of causal mechanisms which exist in the domain of physical systems.

Such a mixed inductive and deductive approach to learning will likely be necessary in most realistic domains where background knowledge exists, but is incomplete. Inductive methods can be used in the absence of domain knowledge, and deductive methods can bring domain knowledge to bear when available.

### **The Learning Task**

Given a structural description and a sequence-of-events behavioral description of a physical system, my learning program hypothesizes causal relations which can account for the observed behavior. It also tries to show how the system's structure supports the proposed causal relations.

### **Representations**

Physical systems are represented as collections of objects participating in various *structural* relations. Continuous properties of objects are represented as *quantities* associated with the objects. My representations for quantities and dependencies are adopted from Forbus [123].

## INDUCTIVE LEARNING

### The Inductive Approach

The inductive approach to learning causal relations I have developed is based on the following representational choices and assertions: 1) Causal relations manifest in observable, *qualitative* changes in the values of quantities, e.g., a quantity increases, a quantity stops decreasing. 2) The basic causal relation is taken to be the functional dependence between quantities. 3) The value of a quantity can be the net result of the contributions of several functional dependencies. Contributions can be classified as positive or negative.

This rudimentary knowledge already supports some simple inductive inferences. For example, a quantity which is increasing may be accounted for by a single positive contribution, several positive contributions, or by mixed contributions whose net result is positive. These inductive inferences can be further constrained by noting that there are only a finite number of ways in which the underlying contributions on a quantity can change to produce an observable change in the value of a quantity.

For example, a quantity in a state of increase can change to a steady quantity either if the positive contributions go away or if balancing negative contributions are added. This knowledge was used by the learning system to understand why water rising in a sink stops rising when the safety drain is reached. Since the positive contribution on the height of the water associated with the faucet being on was still intact, the learning system hypothesized a new, negative contribution and associated it with the safety drain.

The representations and knowledge supporting this inductive method, along with further implemented examples, are described more fully in [102].

### Associative vs. Mechanistic Causal Relations

The causal hypotheses that can be generated within this framework are purely *associative*. They are empirically-based only; there is no explanation of why a particular cause should result in a particular effect, only the observation that it has.

But the concept of causality carries more than the notion of

*regularity.* There is also *direction* and *mechanism* [220]. The notion of mechanism is at the heart of what philosophers call the *necessity* of causes -- the tie between cause and effect which makes the effect inevitable, given the cause. The notion of mechanism is also at the heart of the deductive approach used by my learning system in the construction of causal models.

## DEDUCTIVE LEARNING

### Causal Mechanism

My concept of causal mechanism holds that there is always some *process* underlying causal relations. This process is represented by a signed functional dependence between typed quantities, perhaps with a temporal duration. In addition, causal relations are always supported by a *medium*, the structural link between the physical objects whose quantities are linked in a functional dependence. The concept of medium suggests that of *barrier*, the means by which objects participating in a causal interaction can become decoupled. Medium and barrier are specific kinds of enabling and disabling preconditions tailored to the concept of causal mechanism.

### Fundamental Causal Mechanisms

Within this representation, I have enumerated a set of *fundamental* causal mechanisms. These include mechanical couplings, flows and material transfers, and field interactions. For example, a mechanical coupling is a dependence between the motions of two objects, enabled by a physical connection between the objects. If the connection is rigid, the interaction is expected to be instantaneous. A possible barrier is a break in the connection.

Although I do not see a way of arguing for the *completeness* of this set of causal mechanisms, I would suggest that the set is *adequate* for describing most causal relations in the domain of physical systems.

### The Deductive Approach

The deductive inference method employed by the learning system is simple template matching, or frame instantiation. The template for a particular causal mechanism, i.e., types of quantities, duration, medium, relevant barriers, etc., is instantiated into the structural and behavioral description of the physical system. A match generates a ready-made causal model.

As an example, the coupled motions of the depressed lever and the

disappearing bread in a toaster suggest a mechanical coupling. The learning system can look for the physical connection of the lever, toast carriage, and bread to verify this causal relation. It might also try breaking the connection (removing the bread) as an experiment, and confirming that depressing the lever now has no effect on the bread.

## STATUS

The inductive learning method for constructing causal models has been implemented and tested in two domains -- a sink and a toaster [102]. The deductive method is now being implemented and the combined method will be tested in several (5-6) domains in the coming year.

## CONCLUSIONS

Learning systems should use both inductive and deductive inference methods. I believe part of the distinction lies in the nature of the knowledge driving the learning process. In my learning system, inductive learning relies solely on knowledge of the *form*, or syntactic features of causal relations. The evidence for inferring some number of positive and negative contributions is empirical only, and biases [371] may be needed to choose between several possible inductive inferences.

On the other hand, the deductive learning is driven by an *independently justified* domain theory -- in this case, knowledge of the kinds of causal mechanisms which exist in the domain of physical systems. Inferences become proof-like verifications. A causal relation is asserted when all the attributes of a known causal mechanism are noted.

Neither kind of learning should be considered a panacea. Learning systems should utilize domain theories whenever possible but should also be able to operate with weaker, domain-independent inductive methods. Furthermore, a knowledge engineering phase to support deductive learning should not be considered inappropriate in the construction of a practical learning system.

## ACKNOWLEDGMENTS

This research is supported by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505.

# **FUNCTIONAL PROPERTIES AND CONCEPT FORMATION**

**J. Daniel Easterlin**

Irvine Computational Intelligence Project

Department of Information and Computer Science, University of California  
Irvine, California 92717

## **ABSTRACT**

Previous research in Machine Learning suggests that the process of concept formation can be divided into three distinct components. The first of these -- aggregation -- involves grouping instances of experience into collections. The second component -- characterization -- involves generating a description of the instances in the aggregate. The final process -- utilization -- consists of making use of the resulting description. These components are examined in more detail in this paper and the derivation of functional properties of a real-world environment is discussed within this framework.

## **AGGREGATION**

Aggregation is a process of collection, in which instances of experience are grouped together into a set. It is not a process of description, but a process of coalescing experience into aggregates from which a description may be subsequently formed.

There is usually an underlying motivation for forming an aggregate. In the task of learning from examples, this motivation is obscured, because a tutor provides classification of instances into positive and negative examples from which the concept may be formed. However, in the task of learning search heuristics, where aggregation is performed by the learning system itself, the motivation is apparent because experiences are aggregated on the basis of their contribution to a successful problem solution. In those systems, aggregates are generated by the learner on the basis of their contribution to system performance, rather than relying on the judgment of a tutor.

The importance of identifying aggregation as a distinct process in concept formation is that it focuses attention on what constitutes significance for the system. An aggregate represents an important collection of experience. The purpose of forming a concept to characterize

the aggregate is to recognize and formalize that importance. Thus, for example, a concept is formed to identify those problem states to which an operator may be successfully applied because those states are important in solving a problem.

## CHARACTERIZATION

Characterization is the process that is usually discussed in Machine Learning under the name "concept learning". It involves constructing a description for an aggregate of experience, based on individual descriptions of each member of the aggregate. Researchers in Machine Learning have proposed a variety of techniques for characterization and these techniques constitute the usual material for courses in Machine Learning.

The significant issue of characterization is that the generated description is intended to capture the essential information of the aggregated experience in terms of characteristics that are useful to the system. Characterization attempts to thresh out those aspects of the aggregated experience that make that particular aggregate important to the learning system and formulate those aspects into descriptions that are meaningful. Consider the careful crafting of a generalization hierarchy to be used for characterizing the particular constituents of a string of mathematical symbols that make it amenable to application by an integration operator.

Thus, while aggregation identifies important experience, characterization attempts to explicitly represent what is important about that experience.

## UTILIZATION

The utilization process integrates the characterization or concept description with the performance element of the system. The significance of utilization is that it tries to evoke the important property of the aggregate through use of the characterization, and so it serves as a test of the efficacy of the characterization process in capturing the important aspects of the aggregate. Thus, applying heuristic search rules in a problem solving domain tests the effectiveness of the characterization process in identifying problem states that are significant in finding solutions. Similarly, attempting to recognize new instances of a tower in a blocks world domain tests the effectiveness of the characterization process in elucidating those characteristics of towers that are significant for recognition.

In summary, we see the significance of concept use in concept

formation. Where characterization attempts to capture explicitly the importance of an aggregate, utilization completes the circle by determining the efficacy of the characterization in capturing that importance.

## FUNCTIONAL CONCEPTS IN A REAL-WORLD ENVIRONMENT

This view of concept formation is being employed in a learning system that operates in the C-MU/UCI World Modelers Project. The domain consists of a simulated real-world environment containing simplified objects that behave according to physical laws. The learning system is an agent performing in the environment that is driven by both external task demands and internal needs, both of which are represented in the agent as goals to be satisfied.

For the agent, concepts play a significant role in achieving goals. Using concepts gleaned from its simulated world, the agent constructs plans that guide the application of its operators to the world. These plans are developed using concepts containing functional information about objects in the world. Our desire is to enable the agent to use objects in satisfying goals by exploiting their functional properties.

Functional properties in the world are viewed as "world-operators" that map from world states and the operators of the agent into other world states. Thus, functional properties in the world can be exploited under certain world conditions by the application of certain operators to yield desirable world states. For example, a car's functional property of providing transportation is only manifested under certain world conditions (like there is a car and there is a road) and under certain operators (like getting in and steering). Without these world conditions and operators being present the functional property could not be exploited. Learning functional properties of the world is learning the correct mapping from world states and operators to consequent world states. Knowledge of functional properties enables the agent to make use of objects in pursuing its goals.

Within the presented framework for concept formation, experiences in the simulated world are aggregated by goals that are satisfied. An experience's importance derives solely from its presence in a procedural sequence satisfying a goal. The experience's representation consists of a world state description, the operators applied by the agent in that world state, and the resultant world state. The functional properties of the constituents of the world state are already embodied in the state description, but must be made explicit so they can be used in planning. Thus, characterization attempts to elucidate these functional properties by

identifying general mappings from world states and operators indexed under the same goals to consequent world states. This is a process of abstraction wherein those characteristics of world states and operators are determined which are essential for manifesting the functional property that yielded the goal.

Finally, since these concepts about functional properties of the world are created for the purpose of being used in planning, they describe explicitly what operators and attending world conditions are necessary to achieve goals. They may be used directly for planning how to get to a desired goal state from the current state. The execution of the plan tests the degree to which the agent has been able to extract functional properties of the world. Thus, utilization provides the test of the efficacy of embodied knowledge in capturing the essential characteristics of experiences aggregated because of their participation in satisfying goals. In conclusion, the three component view of concept formation has contributed to an understanding of how to extract functional properties from the real-world environment.

## ACKNOWLEDGMENTS

This work was supported by the IBM Corporation.

# **EXPLANATION-BASED LEARNING IN LOGIC CIRCUIT DESIGN**

**Thomas Ellman**

Department of Computer Science, Columbia University  
New York, New York 10027

## **ABSTRACT**

Our research is focused on a technique known as *explanation-based learning*. We are studying this technique by applying it to the task of learning to design logic circuits. This approach involves utilizing domain knowledge to analyze example circuits and produce generalized circuit designs. In particular, proofs of design correctness are used to guide the process of generalization. This method has been demonstrated in a program which generalizes a circular shift register into a schema describing devices capable of computing arbitrary permutations.

## **INTRODUCTION**

Research in machine learning has identified two contrasting approaches to the problem of generalizing from examples. The traditional "empirical" approach involves looking at a large number of examples in order to identify similar features and build a generalization. The alternative "analytical" approach takes the point of view that generalization may proceed on the basis of a single example, if the system is provided with sufficient background knowledge of the domain under study. Our research involves one particular analytical technique known as explanation-based learning. This technique works by taking an input example and building an explanation of how its features or components are related to each other. The explanation is then used to guide the process of generalization, by identifying constraints which must be maintained as the example is generalized.

## **CURRENT WORK**

Our learning program is envisioned as part of a complete system for designing circuits according to explicit specifications. The problem solving module for such a system would take circuit specifications ( $S$ ) as input,

and produce a design (D) as output. The learning module takes as input a pair (S,D) consisting of specifications and a design which implements the specifications. The goal of the learning process is to produce a generalized schema ( $S^*, D^*$ ) containing generalized specifications and a generalized design. This is achieved in two steps. First the system uses its background knowledge to explain the behavior of the example circuit. In particular, the system builds a proof verifying that the original design (D) correctly implements the original specifications (S). The proof is then used to guide the process of generalization. The resulting schema, ( $S^*, D^*$ ), represents all designs which can be proven correct using the original proof tree.

Our technique for generalizing circuit designs has been demonstrated in a program which generalizes a four bit circular shift register. The shift register circuit is shown in Figure 1. This circuit is constructed using d-type flip-flops, labeled "DFF", and multiplexers, labeled "MUX". It can perform any one of the four operations listed in Figure 2. The specific operation is determined by the state of the two bit "SELECT" line.

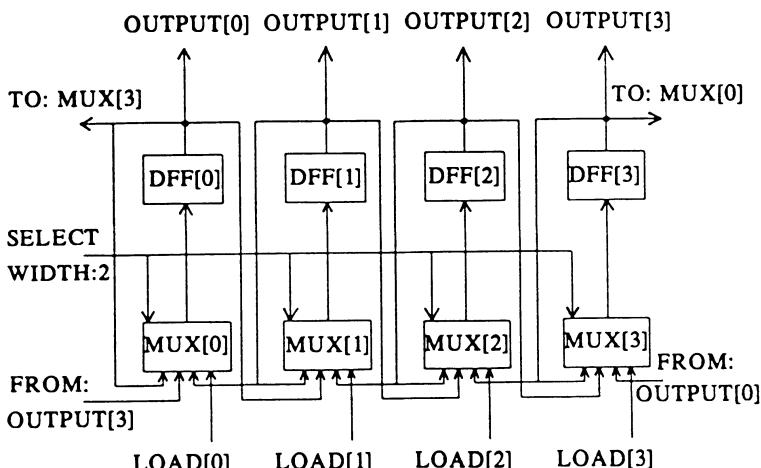


Figure 1: Circular Shift Register

Our program generalizes this circuit into a schema describing devices capable of computing arbitrary permutations of four bits. Any permutation of four bits can be implemented using a circuit similar to the shift register shown above. The only changes involve wire connections between the d-flip-flop outputs and the multiplexer inputs. The d-flip-flop outputs must be connected to the multiplexer inputs so that the data moves in a manner

<u>SELECT-CODE</u>	<u>OPERATION</u>
(0 0)	No Operation
(0 1)	Rotate Right
(1 0)	Rotate Left
(1 1)	Load

**Figure 2:** Control Codes for Circular Shift Register

consistent with the chosen permutation. Our program produces a schema which explicitly describes how the wire connections depend on the chosen permutation. (For additional information, see [108]). This research is similar to the work reported in [250]. Our work differs primarily by focusing on the task of building generalized schemata, rather than the task of designing a new circuit by analogy with a known circuit.

## ISSUES FOR FUTURE WORK

We have demonstrated that explanations of circuit behavior, (i.e. proofs of correctness), can facilitate the process of generalizing logic circuit designs. Nevertheless, a great deal remains to be learned about how this technique works most effectively. Proofs can be represented in a variety of ways. The behavior of a circuit can be explained at various levels of abstraction. These choices can have a significant impact on the resulting generalizations. In the future, we will be examining different types of explanations and their relative usefulness for the purpose of building generalizations.

A related issue involves the choice of representations for specifications and designs of logic circuits. These choices can also have a significant impact on the resulting generalizations. In the shift register example, the choice of representation limited the resulting generalization so that the length of the shift register could not be generalized. (See [108].) We plan to investigate methods of overcoming such limits arising from the choice of representation.

Additional research will address the question of how much an example design should be generalized. Our proof-guided method allows generalizing an example into a schema representing all designs which can be proven correct using the original proof tree. It might be desirable to build a less general schema instead. There is an apparent trade-off here. A specialized schema can be more easily applied to a new example than a highly generalized schema; however, a more general schema has the advantage of applying to a wider variety of examples.

Our research is based on the idea that intelligent systems can learn more effectively when they utilize background knowledge of the domain under study. Such prior knowledge may be represented and applied to learning tasks in a variety of ways. Building explanations is only one way of applying background knowledge to the problem of learning from examples. We would like to identify other mechanisms for utilizing prior domain knowledge.

## **ACKNOWLEDGMENTS**

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165. In addition, this research has benefited from contributions by Michael Lebowitz.

# A PROPOSED METHOD OF CONCEPTUAL CLUSTERING FOR STRUCTURED AND DECOMPOSABLE OBJECTS

Douglas Fisher

Department of Information and Computer Science  
University of California, Irvine, C.A. 92717

## ABSTRACT

A method of clustering structured, decomposable objects in the presence of background knowledge is proposed, with the ability to construct its own 'bias' (generalization hierarchies), thus allowing it to evolve from search-intensive (knowledge limited) behavior to a knowledge intensive (search limited) behavior.

## INTRODUCTION

Conceptual clustering is a process abstraction originally defined by Michalski [234] as an extension of numerical taxonomy, a class of methods developed by social and natural scientists for creating classification schemes over object sets. Conceptual clustering methods discover concept (intensional) descriptions for object classes (i.e., clusters) participating in a classification scheme and use these concept descriptions for object classes (i.e., clusters) participating in a classification scheme and use these concept descriptions to evaluate the *quality* of object classes, whereas methods of numerical taxonomy yield only extensional object class representations. Two problems must be addressed in conceptual clustering.

- The *aggregation* problem involves determining useful subsets of an initial object set. Thus it consists of identifying a set of object classes, each defined as an extensionally enumerated set of objects.
- The *characterization* problem involves determining a useful characterization (concept) for some (extensionally defined) object class, or for each of multiple object classes. This is simply the problem of *learning from examples*.

Current conceptual clustering methods exploit well-understood methods

of learning from examples, by making such a process subordinate to a higher-level aggregation process. This interaction can be viewed as a two-tiered search, which can be used to frame a number of conceptual clustering algorithms [121].

## LIMITATIONS OF EXISTING CONCEPTUAL CLUSTERING SYSTEMS

Current conceptual clustering systems are limited in a number of important respects. Foremost among these is the limited object and concept languages utilized. Present systems allow objects to be represented in terms of attribute – value pairs. This language can be extended in two ways.

- *Structured* object representations can be allowed, where relations between attribute values of an object can be explicitly represented.
- *Decomposable* object representations can be allowed, where attribute values of an object are themselves objects which may be further decomposed.

A second means by which current conceptual clustering systems could be extended is to endow them with the ability to utilize *background knowledge* to augment object descriptions. Background knowledge as used by Vere [381] is a body of relations defined over the attribute values of objects. For example, to identify a concept, 'straight' in poker, it is not sufficient to simply know the cards of individual hands (5 of spades, 4 of hearts, 3 of spades, . . . ), but background knowledge (5 is just greater than 4, 4 is just greater than 3, . . . ) is also required.

## CLUSTERING STRUCTURED AND DECOMPOSABLE OBJECTS

A method of clustering structured and decomposable objects in the presence of background knowledge is currently being investigated. The method, originally inspired by Vere's THOTH system [381], constructs a hierarchical classification scheme in a bottom-up manner. Given a set of objects, aggregation of objects is accomplished by repeatedly 'fusing' individuals to form higher-level classes. Characterization of these classes is dependent on aggregation and characterization over low-level components of class members. Thus, the clustering task is recursively defined. Aggregation of higher-level objects constrains aggregation at lower levels.

Consider this example: Assume the algorithm observes a number of (teacher generated) solution paths (a sequence of operator applications) for solving linear equations of 1 variable (i.e.,  $ax + b = c$ , where  $1x + 0 = d$  is a 'solution'). The operators (represented as *relational productions* [381]) used by the teacher (but not known by the clustering algorithm) correspond to 'subtract' (i.e.,  $ax+b=c \Rightarrow ax+0=< c-b >$ , where  $b > 0$ ), 'add' ( $ax-b=c \Rightarrow ax+0=< c+b >$ , where  $b > 0$ ), and 'divide' (i.e.,  $ax+0=c \Rightarrow 1x+0=< c/a >$ ). The algorithm is given only a set of relational production instances (e.g.,  $3x-2=1 \Rightarrow 3x+0=3$ ), each considered an 'object'. Each production has two components – a 'before' and 'after' state. Each state corresponds to a linear equation with (integer) components  $a$ ,  $b$ , and  $c$ . Intuitively, the algorithm is expected (through clustering) to identify production classes, each of which corresponds to an operator class for transforming linear equations. In aggregating production instances, classes corresponding to 'subtract', 'add', and 'divide' operator instances will be generated. By recursively clustering over the components of the members of each of these classes, hierarchical classifications over the bottom-level integer components are generated. These hierarchies (containing nodes corresponding to integer classes such as *< positive >* and *< nonpositive >*) in turn can be used to guide the characterization process of higher-level objects. As clustering proceeds, we would expect that the structuring of low-level objects would serve to increasingly constrain the search for higher-level object class characterizations. Thus, one hope of the proposed work is that a clustering system can be realized which evolves from search-intensive behavior to increasingly search-limited behavior.

## VALIDATING THE PROPOSED ALGORITHM

We wish to validate the algorithm in a variety of domains. Methods of conceptual clustering have thus far been tested on such domains as micro-computers, Spanish folk songs, and animals. However, the behavior of and classification scheme resulting from a conceptual clustering process are open to many varied interpretations. For example, GLAUBER, a system for discovering qualitative empirical laws in the domain of chemistry, by Langley, et.al [200] has been framed as a conceptual clustering system [121]. In creating a classification over a set of molecular structures, concepts derived for molecule groups can be viewed as scientific laws relating these groups. A similar view can be taken of another discovery system, BACON [200]. Of interest also are problem-solving tasks in the domains of algebra and integral calculus.

## CONCLUDING REMARKS

A conceptual clustering system for structured and decomposable objects in the presence of background knowledge has been proposed. The algorithm is to be tested in a number of domains, including integral calculus, algebra, as well as interpreting the algorithm's behavior as scientific discovery. Investigations into the algorithm's ability to build its own 'bias' should demonstrate that clustering becomes increasingly directed (i.e., less search intensive) over the duration of a given clustering task. The view throughout this work is that conceptual clustering, despite its original motivation as a data analysis tool, may be viewed as concept formation with behavior open to widely varying interpretations.

# **EXPLOITING FUNCTIONAL VOCABULARIES TO LEARN STRUCTURAL DESCRIPTIONS**

**Nicholas S. Flann and Thomas G. Dietterich**

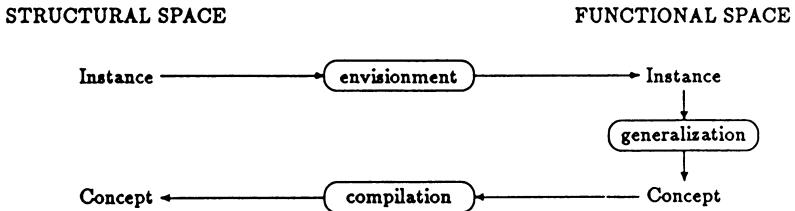
Oregon State University, Department of Computer Science  
Corvallis, OR, 97331

## **INTRODUCTION**

In the idea paper entitled "Learning Meaning," Minsky [241] stresses the importance of maintaining different representations of knowledge, each suited to different tasks. For example, a system designed to recognize examples of cups on a table would do well to represent its knowledge as descriptions of observable features and structures. In contrast, a planning system employing cups to achieve goals would require a representation describing the purpose and function of cups. When we turn from the issue of *employing* a description of a cup to the task of *learning* such a description, it is not immediately obvious what vocabulary should be used. One approach might be to choose the vocabulary appropriate for the performance task (i.e., structural descriptions for recognition, functional descriptions for planning, etc.). This approach has been pursued, e.g., by Winston [396], Buchanan & Mitchell [46], Quinlan [282], and Minton [243]. In the case of Winston's ARCH learner and Buchanan & Mitchell's Meta-DENDRAL system, this approach worked well because good structural vocabularies were available. However, Quinlan and Minton confronted much more difficult problems in constructing structural vocabularies that concisely captured the desired game-playing concepts. Quinlan, for example, spent two man months developing the vocabulary for the concept of "lost-in-3-ply."

The difficulty that these researchers encountered is that in addition to considering the representation requirements of the performance task, one must consider the representation requirements of the learning process. By their very nature, inductive learning systems must operate on the syntactic form of the descriptions being learned. The biases of such systems are stated in syntactic — hence, vocabulary-specific — terms. To meet the combined requirements of the performance task and the learning process, the vocabulary must be very carefully engineered!

An alternative to this "single-vocabulary" approach is to employ two different vocabularies: one for learning and one for performance. In many domains, the desired concept can be expressed very succinctly in one



**Figure 1:** Functional Learning Scheme

vocabulary, and yet this vocabulary cannot be efficiently employed to perform the desired task. For example, Winston et al. [401] have shown that the concept of "cup" can be expressed very succinctly in a functional vocabulary, and yet it is difficult to use such a description to recognize pictures of cups. Figure 1 summarizes this approach to acquiring an efficient structural description by first learning a succinct functional description. In this approach, the given examples (presented in a simple structural vocabulary) are converted into functional examples through a process of "envisionment." Then the functional examples are inductively generalized to obtain a functional concept description. Finally, this concept description is converted to a structural concept description through a compilation process.

We have developed a program (named *Wyl*) that learns simple concepts in chess and checkers via this "two-vocabulary" approach. We have chosen board games such as chess and checkers because there are many interesting concepts that have simple functional characterizations (e.g., "trap," "skewer," "fork," "lost-in-2-ply") and yet have quite complex structural definitions. *Wyl* has been applied to learn definitions for "trap" and "move-to-trap" in checkers and "skewer" in chess. We illustrate *Wyl*'s functioning with "trap".

## AN EXAMPLE: "TRAP"

### Envisionment

*Wyl* starts with a given structural training instance (i.e., board position), which it is told is an instance of "trap" (Figure 2). To convert this into a functional instance, it conducts a forward minimax search according to the rules of checkers looking for "interesting" positions (e.g., win, loss, draw, loss of important piece, etc.). The result of this search

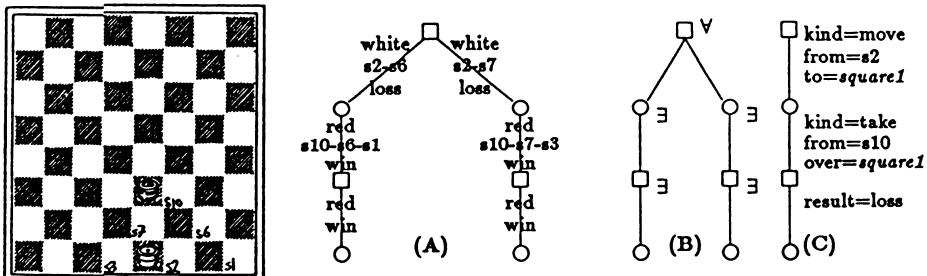


Figure 2: Checkers concept trap

process is a game tree (A) in which each move is tagged with its computed consequences, returned from the min/max search. The tree is then traversed to construct a proof of the backed-up value of the root. The “theorem” to be proved always takes the form of a series of nested quantifiers (e.g., for a *win*, the proof states that “there exists a move for me such that for all of my opponent’s moves ... I win.” For a *loss*, the proof states that “for all moves I make there exist moves for my opponent such that ... I lose.”). The nodes of the tree are labeled with the appropriate quantifiers, and branches of the tree that do not contribute to the proof are eliminated. This yields tree (B), which in Mitchell’s terms [249] could be called an “explanation tree” for the computed result.

### Generalization

Tree (B) can be viewed as providing two positive functional training instances of a loss for the first player. The generalization step “compresses” these two parallel branches of the tree to obtain tree (C). Two biases are applied to drive this generalization. The first bias is the familiar bias toward maximally specific conjunctive generalizations. The final functional concept definition must be a conjunction (with interspersed nested quantifiers as dictated by tree (B)). To implement this bias, two generalization rules are applied: (i) constants are changed to variables and (ii) the alternative branches of the tree are changed to a universal quantifier. This second generalization rule is equivalent to the inverse-enumeration rule:

$$P(C_1) \wedge P(C_2) \wedge \dots \wedge P(C_n) \Rightarrow \forall c P(c)$$

The second bias employed by Wyl states that “There are no coincidences.” More concretely, if the same constant appears at two different points within a single training instance, it is asserted that those

two different points are *necessarily* equal. Figure 2 demonstrates this bias. In the left branch of tree (A), the white piece first moves to square s6. Then the red piece in square s10 captures the piece in square s6. The "no coincidences" bias says that these two occurrences of s6 were necessarily identical. Hence, when they are matched against the right branch of tree (A) to s7, they are both generalized to the *same* variable *square1*. The final functional concept description states that a "trap" is a situation in which your opponent has a single piece (at s10) that is able to capture your piece (at s2) no matter what move you make. This is an overly specific version of trap. A second structural training instance must be presented to Wyl before the correct definition of "trap" is found.

### Compilation

The third stage of the learning process is termed "operationalization" by Mostow and Keller. Several techniques have been developed for performing this process. Keller [165] and Mostow [258] apply program transformation methods to transform the functional description into a structural description. Utgoff [370] and Minton [243] apply constraint back-propagation. An alternative approach of enumeration and compaction was explored in Wyl as no structural language was available. First, an intelligent generator is applied to generate all possible board positions consistent with the generalized functional description. In the "trap" case, 146 possible positions are generated. Second, two algorithms are applied to compress this set of 146 positions into a disjunction of 12 general descriptions. One algorithm identifies common internal disjunctions over squares and move directions to define terms such as "center," "north," and "double-corner side." The second algorithm invents relational terms as compositions of primitive relationships. For example, the term *north-2-squares(square1 square2)* is defined as a conjunction of *nw(square1 square3)* and *ne(square3 square2)*.

### CONCLUSION

The approach of employing two vocabularies (structural and functional) and performing induction only in the functional vocabulary has significant advantages for acquiring efficient structural descriptions of concepts that have succinct functional descriptions. First, fewer examples are required to learn the concept. Second, the bias built into the learning program is very simple (maximally-specific conjunctive generalization). Third and last, the learning system starts with less built-in knowledge and hence requires less domain-specific engineering.

# **COMBINING NUMERIC AND SYMBOLIC LEARNING TECHNIQUES**

**Richard H. Granger, Jr. and Jeffrey C. Schlimmer**

Irvine Computational Intelligence Project  
Department of Information and Computer Science  
University of California, Irvine, California 92717

## **ABSTRACT**

Incremental learning from examples in a noisy domain is a difficult problem in machine learning. In this paper we divide the task into two subproblems and present a combination of numeric and symbolic approaches that yields robust learning of boolean characterizations. We have implemented this method in a computer program and present its empirical learning performance in the presence of varying amounts of noise.

## **INTRODUCTION**

A learning engine in an unconstrained, active environment faces a difficult task, for it must be able to accurately predict the outcomes of its actions. Moreover, there are a large number of features in the environment, the environment is occasionally inconsistent, and the learner must adapt its performance in a continuous manner. This is a task for which neither numeric or symbolic learning methods alone appear to be sufficient. Existing machine learning approaches have various shortcomings: either they are non-incremental (e.g., [229]), requiring a large number of instances before learning can occur; they require an explicit theory of the domain (e.g., [254]) or a benevolent tutor; or as many do (e.g., [396, 248]), they have significant trouble with inconsistent instances or don't learn at all in the presence of noise. The section on related work discusses this in more detail.

In this paper we describe an alternative approach that combines numeric and symbolic learning methods and operates in the domain of learning from examples. Each characterization under consideration by the program is assigned a measure of effectiveness as a result of a statistical summarization process. Instances are not compared with other instances, but with this summary of past instances. The summary is also used to guide formation of new compound characterizations through the exponential space of possibilities while remaining within reasonable memory limitations.

Because the algorithm is incremental and does not rely on a large memory of accrued instances, it is able to track changes in the environment over time and remains robust with respect to noise.

## SALIANCE ASSIGNMENT

The domain considered in the context of this paper consists of simple, non-relational blocks, a traditional domain for AI systems. Blocks may be described in terms of their size, color, and shape (e.g., a large red circle). The proper characterization of the concept to be learned may involve a conjunction across slots (e.g.,  $\text{Size}[\text{large}]$  and  $\text{Color}[\text{red}]$ ), a disjunction of fillers within a slot ( $\text{Color}[\text{red or blue}]$ ), a disjunction between slots ( $\text{Size}[\text{large}]$  or  $\text{Color}[\text{red}]$ ), or the negation of any of the above.

There are two subproblems an algorithm in this domain must solve: which of the features are the significant ones (the *salience assignment* problem) and which of the possible boolean functions over the features should be formed (the *composition* problem). Much of the previous work in learning from examples has assumed that conjunctive descriptions are sufficient to characterize the instances presented (see e.g., [396, 380, 248]). However, this approach addresses degenerate forms of the salience assignment and composition problems, because determining which features are important immediately equates to determining which boolean conjunction is appropriate.

The outputs of the salience assignment problem are a classification of each of the characterizations into one of three roles: an accurate predictor of positive instances, an accurate predictor of negative instances, or an 'uncorrelated' characterization which does not enable an accurate prediction of either kind.

The inputs for this problem may be divided into one of four logical possibilities: a characterization is present in a positive instance (*prediction*), present in a negative instance (an error of *commission*), absent in a positive instance (an error of *omission*), or absent in a negative instance (*non-prediction*). The first and last possibilities indicate a positively predictive characterization. Errors of commission and omission indicate a negative or uncorrelated cue.

A constraint from psychology helps identify an appropriate algorithm. Specifically, animal experiments [300] clearly identify a necessary condition for learning an association between two events E1 and E2: the likelihood of E2 must be greater following E1 than it is without E1. Formally,  $p(E2/E1)$

$> p(E2/\overline{E1})$ . This is called the law of *contingency*. This principle is in

contrast to approaches which simply strengthen an association whenever E1 and E2 are paired together and weaken it when they are not (e.g., when Color[red] (E1) is tagged as a positive or negative instance (E2)).

Bayesian statistics (see e.g., [103]) provide similar formulae for the calculation of two values in inductive logic: *Logical Sufficiency* (LS), which indicates the extent to which the presence of one event predicts another particular event; and, reciprocally, *Logical Necessity* (LN), which represents the extent to which the *absence* of an event *decreases* expectation or prediction of the second event. Our algorithm makes use of an incremental method of calculating approximations of these two values, via a simple formula composed of precisely the four possible categories of pairwise feature occurrences given above.

$$LS = \frac{s(n+o)}{o(s+c)} \quad LN = \frac{c(n+o)}{n(s+c)}$$

where  $s$  is the count of successful predictions,  $c$  is errors of commission,  $o$  is errors of omission, and  $n$  denotes non-predictions.

Our algorithm utilizes these approximations of LS and LN to assign a role to each of the characterizations in an incremental manner. The role assigned is based on the interpretation of the LS and LN values. LS and LN values range from 0 to *infinity*, with high LSs corresponding to a characterization strongly predictive of positive instances and very low LSs corresponding to the case where the characterization implies a *negative* instance. When the value of LS is approximately 1 then the characterization is considered uncorrelated. LN values are interpreted reciprocally. That is, high LN values indicate a necessary characterization for a negative instance; LN values about equal to 1, a lack of correlation; and LN values much less than 1, the necessity of a characterization for a positive instance.

## COMPOSITION

Learning in a complex domain necessitates noting useful *combinations* of characterizations. For instance, a conjunction of the size and color may indicate a positive instance while neither the size nor the color alone do.

Our algorithm for the composition problem is failure driven and, upon an error, it compares an instance with the classifications of characterizations provided by the salience assignment algorithm in the

process of composing new characterizations.<sup>1</sup> When the algorithm makes an error of omission, it is behaving as if the characterizations taken as a whole are too specific; as a result, a new characterization is formed which is the disjunction of the most significant characterizations (measured by sufficiency, i.e. LS). Upon an error of commission (behaving too generally) a new characterization may be formed which is the conjunction of two significant characterizations. The measure of necessity, or LN, is used to determine the best candidates. Negations are also proposed following an error of commission.

This composition algorithm does not continue to form new characterizations without bound. Two mechanisms serve to limit this growth: (1) new characterizations are only introduced following a failure; and (2) newly introduced characterizations *compete* against their components. When a new characterization out performs its components (measured by a thresholding mechanism), the components are deactivated. This threshold mechanism allows the algorithm to correct some erroneous characterization formations, and to follow changes over time in the true characterization of the concept (i.e. when the world changes).

Figure 1 depicts the performance of our computer implementation with various rates of inconsistency. The program was trained for the concept Color[red or blue] with various ratios of inconsistent instances to consistent ones. As the ratio of inconsistent instances approaches 0.2, the performance falls toward a chance level of 50%. As one would expect, rates of inconsistency in excess of 0.5 cause the program to acquire the opposite of the concept and perform at less than a chance level. The performance curves for disjunction and conjunction between slots are similar.

## RELATED WORK

There has been a significant amount of work done in machine learning on the task of learning from examples. Numeric techniques (e.g. [312]) tend to be tolerant with respect to noise but have some difficulty exploring the exponential composition space of characterizations. Michalski and Stepp [229] address the larger task of conceptual clustering and deal

<sup>1</sup>Given the specific algorithm presented above, this amounts to comparing an instance against a statistical summary of the previous instances. This is in contrast to approaches which compare instances with other instances (see e.g., [198]), or instances with symbolic generalizations of other instances (e.g., [248]). Utilizing this methodology allows the algorithm to propose disjunctions and retain tolerance to noise.

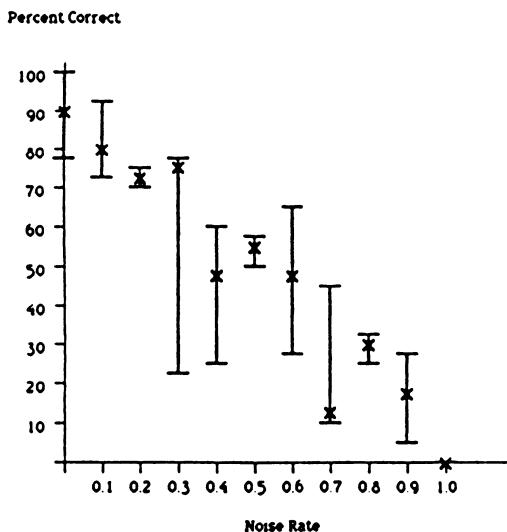


Figure 1: Performance as a function of noise.

effectively with the space of characterizations and noise. However, they use a non-incremental algorithm which also requires a large storage of instances for learning to occur.

Symbolic techniques, on the other hand, have been used successfully by a number AI learning programs though they fail to tolerate noise [248] and may require specific instances to maximize learning [396]. Recent analytic approaches minimize the number of instances required to accurately characterize a concept [254]. Yet they rely heavily on a nearly complete domain theory, which restricts the applicability of this work to only those domains and problems which offer pre-built-in domain theories.

Cognitively oriented machine learning researchers have focused on developing psychologically plausible models of learning. In the ACT\* family of programs [6], the basic associational unit, the production rule, is strengthened when it is reinvented and when it is activated through the spread of activation in memory. Rules are weakened by negative feedback. Similarly, IPP [207] strengthens (or weakens) proposed feature groups by a prespecified amount given positive (or negative) instances. Though this scheme for assigning rule strength may be empirically valuable, it does not appear to be consistent with basic psychological data concerning contingency (see Section entitled "Salience Assignment"). IPP also has some difficulty when any subset of features in a group are predictive while any single one isn't. Lacking a representation for explicit boolean functions prohibits IPP from assigning predictiveness to only a subset of features.

## CONCLUSIONS: CONSTRAINTS AND HEURISTICS FROM COGNITION

Our work on the problems of learning addressed in this paper arose from our twin concerns with accurately modelling human and animal learning behavior, and generating useful AI learning algorithms. We began by examining a number of results from the literature on animal learning psychology. An experimental result of specific interest gave a precise quantification of a necessary condition for animal learning [300] called *contingency*. Pursuing the implications of this class of experiments, we found that a large number of algorithms currently in use in AI systems which are based on a simple 'strengthening and weakening' scheme (strengthen when it works, weaken when it doesn't) are clearly inconsistent with these experimental data (see [132]).

Although this is a valuable observation for cognitive modelling efforts, it wasn't immediately obvious what impact this would have on researchers in machine learning. However, this is an area in which results from AI and cognitive science converge: analysis of the relevant experimental data led us to a highly robust algorithm for the salience assignment task [132]; one that it is doubtful we otherwise would have considered.

We plan to continue to investigate algorithms that account for existing psychological results and incorporate those that yield effective solutions into our ongoing work on machine learning.

## ACKNOWLEDGMENTS

This research was supported in part by the Office of Naval Research under grant N00014-84-K-0391, the National Science Foundation under grant IST-81-20685, and by the Naval Ocean Systems Center under contract N66001-83-C-0255.

# LEARNING BY UNDERSTANDING ANALOGIES

## Russell Greiner

Knowledge Systems Laboratory  
Stanford University

This research describes a method for learning by analogy — *i.e.*, for proposing new conjectures about a target analogue based on facts known about a source analogue. We formalize this process as a rule of plausible inference and present heuristics which guide this process towards efficiently postulating useful new conjectures. The most important rule involves the use of abstractions — abstract relations that encode solution methods to past problems.

This summary paper addresses the issues: [1] what is an analogical inference (read “learning by analogy process”) in general, [2] what is a useful analogical inference, within the context of solving a particular problem and [3] how can such useful analogies be found effectively. En route, it justifies using abstractions for this process. A more complete description of this work appears in [134].

The *general analogical inference* process uses a statement of the form “A is like B” to propose plausible new facts about the target analogue, A, guided only by the weak constraint that these new assertions resemble some facts known about the source analogue, B. For example, the hint “FlowRate is like Current” allows us to infer that FlowRate satisfies some known property of Current. Depending on what we initially know about FlowRate, this hint might suggest that FlowRate is conserved through a junction, that FlowRate’s material (corresponding to Current’s charge) can be stored in a capacitor-like device, that FlowRate obeys Faraday’s Law or even that the modern study of FlowRate began with William Gilbert’s early treatise.

This example suggests the large number of possible analogical inferences. The first refinement, *useful analogical inference*, uses the objective of solving a problem to reduce the space (see [249]). Its input includes a particular problem (dealing with the target analogue) as well as the pair of analogues; and it returns just those analogies which help to solve this problem. This means the useful analogical inference process considers only the subset of legal analogical inferences whose conjectures are “useful” to the given problem. For example, the target problem, “Find the FlowRate through a certain pipe in a particular configuration of pipes”, eliminates all but the first proposed analogy shown above.

The second refinement, *abstraction-based analogical inference*, moves

this "usefulness constraint" into the generator. Rather than allow any arbitrary commonality between the two analogues, this process permits only *common abstractions*, where these abstractions are selected pre-defined relations. The pertinent abstraction for the current example is the resistance analogue, RKK, which consists of a parameterized form of Ohm's and Kirchoff's Laws ([p80-81] [74]).<sup>1</sup> This reduces the analogical inference process to the task of finding an abstraction which has a derivable source instance, followed by a model-driven satisfaction task which seeks a target instance of this same abstraction which is consistent with known facts.

Before presenting the further refinements, we answer three questions about abstractions:

**What is an abstraction?** In general, each abstraction corresponds to a cluster of facts which, collectively, can be used to solve some set of problems. (E.g., this RKK abstraction supplies the information needed to solve problems in electricity and hydraulics, as well as heat flow, force and torque systems.) Other examples of abstractions include Group (from abstract algebra), Aggregation-Loop (a program plan, see [301]), the restaurant script (see [318]) and the height metaphor (see [193]).

**Why can abstractions be used?** This follows from the intuitive (and empirically confirmed) observation that certain clusters of facts can be used to solve many different problems, even in diverse domains. Another reason is that many abstractions correspond to designs, especially in artificial domains. (E.g., program plans are abstractions in the artificial domain of programming.) Here, their re-usability follows from the fact that many artifacts (and so, perhaps, both analogues) have been built following this specific plan.

**What are the advantages of using abstractions?** First, they provide the models needed to focus the search for useful analogies. The difficult part of the analogical inference process is finding the set of useful target assertions to conjecture. Here, the proposed common abstraction suggests these conjectures: they are the underivable component clauses of the target abstraction instance. For example, instantiating RKK for FlowRate may require proposing its component clause K#1(FlowRate); i.e., asserting

<sup>1</sup>Stated formally, RKK is a 4-ary relation, defined by

$$\text{RKK}(t, a, r, l) \Leftrightarrow \text{Ohms}(t, a, r, l) \wedge K\#1(t) \wedge K\#2(a).$$

(Here, K#1 represents Kirchoff's First Law, etc. Each of these conjuncts is considered a *component clause* of the RKK abstraction.) In this context, the phrase "source abstraction instance" refers to an instantiation of this RKK abstraction which involves Current, e.g., RKK(Current, VoltageDrop, Resistance, Resistors). Similarly, "target abstraction instance" refers to an RKK abstraction instantiation involving FlowRate.

that Kirchoff's First Law holds for `FlowRate`. Hence, the analogical inference "completes" the abstraction, as the target abstraction instance becomes derivable after these conjectures have been added.

Secondly, these abstractions make explicit precisely what changes from one analogue to the other and what remains invariant. The commonality is the particular formula used to define the abstraction (e.g., the formula defining `RKK` is shown in Footnote 1). As the domain of the analogical connection is this specific set of component clauses, the analogy cannot be over-extended (c.f., [136]). Different sets of terms are used to instantiate this *n*-ary relation for the source and target analogues; these pairs of analogous terms specify how the two analogues differ. As these "Skolem variables" are explicit, the analogical inference process can transfer some `VoltageDrop`-property over to `PressureDrop`, based on their respective connections to `Current` and `FlowRate`; e.g., this means the analogy process can propose that `K\#2(PressureDrop)` holds.

Even given a specific problem, a pair of analogues and initial knowledge base of known facts and abstractions, there may still be many distinct legal common abstractions. This motivates a third refinement, which is expressed as two sets of heuristics for pruning and ordering the legal analogies. The first set is based on the claim that abstractions are "coherent" clusters of facts, which have been used to solve problems in various domains before. One of these rules provides a particular way of using the target problem to focus the search for relevant abstractions; another insists that all the arguments of an abstraction belong to a single context. The other set of rules claim that "better analogies impose fewer constraints on the world". Basically, these rules prefer the analogies which require the fewest additional conjectures.

This illustrates one important way in which my definition of analogy differs from many others: Most other systems attempt to infer as much as possible about the target. The "fewer constraints" view, on the other hand, claims that it is sufficient to add only the conjectures needed to form a "coherent" cluster, one which suggests a solution to the motivating problem. As this analogy is adequate, there is no need to propose other information.

This progression of refinements culminates in an actual implementation, the *NLAG* program. [Chapter 6 of [134]] includes a variety of minor rules used to operationalize this model of analogy. [Chapter 7 of [134]] then describes some results obtained using this program. For example, this data suggests that, while the idea of abstractions is

important, the abstraction *label* is not. In particular, a program which seeks common *relations* works almost as well as one which seeks common abstractions. That chapter argues that this is because people seldom reify an arbitrary formula unless it has proven useful in solving some problem. As this precisely fits the intuitive definition of abstractions, it suggests that most pre-defined relations are, in fact, already abstractions.

In conclusion, this research proposes a particular mechanism for using an analogy to acquire new facts. It extends many of the current analogy systems by using a non-lexical method, based on abstractions, for determining the set of useful target conjectures. This project also includes a running implementation of this model of analogy and provides a semantic account of the process, based on a variant of Tarskian semantics. Two areas of future research involve generating these abstractions and accepting other types of hints.

## ACKNOWLEDGMENTS

I would like to acknowledge the intellectual contributions from my research advisors, Drs Michael R. Genesereth, Douglas B. Lenat, Bruce G. Buchanan and Peter Hart; and the financial support of ONR #N00014-81-K-0004.

# **ANALOGICAL REASONING IN THE CONTEXT OF ACQUIRING PROBLEM SOLVING EXPERTISE**

## **Rogers Hall**

Irvine Computational Intelligence Project

Department of Information and Computer Science, University of California  
Irvine, California 92717

### **INTRODUCTION**

Computational studies of analogical reasoning and "learning by analogy" are just beginning. As infants in the developmental scale of research on these topics, we stand to inherit much from historically more mature academic approaches. Two of the most important contributions of other academic traditions are:

- metaphor and analogy are common in everyday thinking and expression;
- they arise routinely while assimilating novel experience to existing knowledge.

Although many computational approaches to analogical reasoning have been sensitive to these contributions, most end up proposing special-purpose mechanisms without explicitly considering a larger context in which this form of reasoning occurs. This is probably understandable given the complexity of reasoning and learning under uncertain circumstances; when reasoning by analogy, things literally "are not what they seem." What has yet to be achieved is an approach which firmly integrates analogy into a more general reasoning context and which clarifies the role of analogy in learning.

This thesis will examine components of "learning by analogy" in the context of acquiring problem solving expertise. As described in the existing literature, learning by analogy amounts to an amorphous set of techniques which are not clearly distinguished from other forms of machine learning. An argument will be developed that learning by analogy, apart from the obvious transfer of information from a source to a target situation, can be described as being composed of other forms of learning traditionally segregated in the literature as learning by being told, learning by taking advice, learning from examples, learning by doing, and learning by discovery. In this thesis work, analogical reasoning is embedded in a problem solving

context in which each of these techniques offers a viable mechanism for learning. However, given the uncertainty inherent in pursuing analogies, each of these learning mechanisms must be re-evaluated with respect to strategies for aggregating disconnected events into instance descriptions, clustering instances into classes, characterizing these classes in a useful fashion, and storing characterizations in a manner which will facilitate future performance.

By allowing new situations to be understood *as if* they were instances of known situation classes, a hypothetical element is added to problem solving which both complicates and constrains the learning task. Complications arise in that errorful conceptualizations must be expected, and learning mechanisms must support conceptual evolution in an incremental fashion. Constraints arise in that the learning task is strongly connected to existing knowledge sources, potentially bypassing the need for extensive experience in learning superficially variant problem solving strategies. The ramifications of embedding analogical reasoning within a system which acquires problem solving expertise is the central question of this research.

## A FRAMEWORK FOR ANALOGICAL REASONING

A survey of existing computational studies suggests that analogical reasoning can be divided into three interdependent process components:

- **recognition** of candidate source situations given a target situation,
- **elaboration** of a correspondence mapping between target and source situations,
- and **evaluation** of extensions to this correspondence mapping.

General processing issues are evident for each component. Recognition requires that search in some store of source situations be constrained so that only "promising" candidate sources be considered. This notion of promise requires a careful consideration of the context in which reasoning occurs, and an organization of existing source situations to facilitate such retrieval requires explicit use of this notion. For an incremental elaboration of a correspondence mapping to be tractable, constraint must be exercised in placing aspects of a source and a target in correspondence. This constraint emerges both from the surrounding reasoning context and from specific knowledge of source and target situations. Finally, evaluation of an

extended mapping requires judicious use of interactions between knowledge of source and target situations. None of the issues raised by these component processes of analogical reasoning have answers which suggest general agreement in the current machine learning literature. Each of these components, then, presents open research problems for computational studies of reasoning and learning by analogy.

## LEARNING TO SOLVE ALGEBRA STORY PROBLEMS

The task domain for this thesis is learning to solve algebra story problems typical of instruction in secondary schooling. As an example, consider the following problem:

Two trains leave the same station at the same time. They travel in opposite directions. One train travels 64 km/hr and the other 104 km/hr. In how many hours will they be 1008 km apart?

These kinds of problems provide a constrained but sizable set of background knowledge sources which are necessary for problem solving. Thus learning can be examined in the context of multiple background knowledge sources. In addition, problem solving behavior in this domain has also interested psychologists, providing an opportunity for comparisons between computational and psychological approaches to learning. In brief, the problem solving expertise to be acquired consists of a **conceptual vocabulary** specific to the problem domain (e.g., a frame--like structure for simple motion events) and a set of **problem schemata** which suggest problem solving strategies on the basis of recognizing salient problem characteristics. For example, in the problem above, recognizing that an unknown quantity relates two co-linear motion events, a problem schema might suggest that total distance can be decomposed into the sum of distances from constituent motion events.

Problem solving in this domain amounts to the incremental construction of an increasingly constrained understanding of the unsolved problem. Analogical reasoning occurs as problem schemata are partially matched and must be further supported in order to apply known problem solving methods. Within the context of attempting to solve a given story problem, the system recognizes a "promising" problem schema on the basis of a partial match between enablement conditions in the schema and selected aspects of the current problem (e.g., unknown quantities). The problem solver then attempts to elaborate this match in a fashion which

further supports the partially enabled problem schema.

Tentatively proposed extensions to the correspondence mapping between source (the problem schema) and target (the current problem) situations must be evaluated in terms of knowledge specific to objects and relations in those situations. Thus in the course of confirming enablement conditions for one problem schema, it may be necessary to invoke a number of other knowledge sources, each subject to similar confirmation. Thus, analogical reasoning occurs in a context of connecting known problem solving methods with a novel problem description. However, within the same problem solving context, familiar problems can be solved by direct application of known problem solving strategies.

Analogical reasoning, as described, facilitates learning from problem solving experience in that existing solution strategies can be applied to a wider class of problems, providing it is possible to construct a supporting correspondence mapping between partially enabled schemata and novel problems. Problem solving is more robust and opportunities for learning from experience are more frequent when novel problems are "seen as" variants of more familiar problem classes. However, ambiguity in aggregation and clustering of problem solving experience places added burdens on the learning mechanism in terms of managing potentially errorful conceptualizations.

# **PLANNING AND LEARNING IN A DESIGN DOMAIN: THE PROBLEMS PLAN INTERACTIONS**

## **Kristian J. Hammond**

Department of Computer Science, Yale University  
P.O. Box 2158 Yale Station, New Haven CT, 06520

### **INTRODUCTION**

An important issue in planning is the problem of dealing with goal and plan interactions that arise when planning for multiple goals. The current solutions to this problem have all been within a paradigm of planning for individual goals separately, (using domain specific planning knowledge) and then dealing with whatever interactions arise upon merging them, (using domain independent knowledge of plan interactions). This paradigm, unfortunately, has resulted in planners that lack a vocabulary which would allow them make use of their abstract knowledge of plan interactions when planning for problematic multiple goal situations that have already been encountered and planned for.

My thesis, along with the CHEF program, (a planner in the domain of recipe design), is an attempt to integrate domain specific planning knowledge with abstract knowledge of plan interactions. The CHEF planner, by explaining failures due to plan interactions and storing them under the features implicated by this explanation, is able to anticipate and plan for these already encountered interactions before they occur again in a new situation.

### **BASIC GOALS**

The CHEF project has three theoretical goals:

The first is to make use of domain specific explanations of planning failures to index to abstract knowledge of how to deal with those failures. This is an attempt to provide a bridge between a planner's knowledge of the physics of its domain and its abstract knowledge of plan interactions. This is accomplished by building up an explanation of the causality underlying each planning failure which is encountered, using the causal rules of the domain. The general form of the explanation is then used to index to a knowledge structure, or TOP [319], that contains domain independent information about how to repair this type of failure in general.

For example, after designing a strawberry soufflé, (built by modifying

a vanilla soufflé in its plan library), the CHEF planner finds that the plan it has built results in a soufflé that doesn't rise. It explains this failure by chaining through its baking rules and finding that the amount of liquid produced by a plan step, pulping the strawberries it has added, is too much for the leavening material, the whipped egg whites and yolks, to bear. It also notes that the pulping step results in a state which is a precondition for mixing the strawberries with the other ingredients, in turn enabling the goal of the soufflé tasting like strawberries. Its explanation, in more general terms, is that a side effect of a required plan step, (pulping the strawberries), has disabled a condition, (a balance between liquid and whipped eggs), required for another step to satisfy another goal, (having the soufflé rise). This general explanation then, is used to index to the TOP SIDE-EFFECT:DISABLED-CONDITION:BALANCE, (henceforth SE:DC:B), which contains general information about *where* the failed plan can be modified as well as a set of strategies detailing *how* the situation can be repaired.

The second goal of the CHEF project is to find repair strategies for each of these interactions and devise a method for choosing among them while implementing them within a domain. These strategies are similar to Sussman's and Sacerdoti's *critics*, [365, 309], and Wilensky's *meta-planning* work, [392]. It differs from this work, however, in that the knowledge structures defined by different plan interactions are indexed by general characterizations of the interactions themselves and have a wider variety of possible replanning strategies associated with them. Further each of these strategies has the pragmatic considerations that make it appropriate to apply associated with it. The choice between strategies, then, is made on the basis of these pragmatic considerations. This choice requires a library of plans indexed by both domain specific goals, and pragmatic constraints determined by the existence of interactions with other plans.

Continuing with the soufflé example, the TOP SE:DC:B has five basic strategies associated with it:

- ALTERNATE-PLAN:PLAN1: Find an alternate plan for satisfying the initial goal that does not produce the undesirable (e.g., Use strawberry preserves.)
- ALTERNATE-PLAN:PLAN2: Find an alternate plan for satisfying the failed goal that does not require the now absent balance between states. (e.g., Make use of the more robust cake plan instead of a soufflé plan.)
- ADJUNCT-PLAN: Run an adjunct plan that allows the second

plan to succeed even in the presence of the imbalance. (e.g., Add flour to the soufflé mixture.)

- RECOVER-FROM-SIDE-EFFECT: Do something to get rid of the side effect before running the next step in the plan. (e.g.. Drain the strawberries after chopping.)
- ADJUST-BALANCE: Adjust the other state in the balance that has been put in jeopardy by the problematic side effect. (e.g.. Whip more egg white and egg yolk to compensate for the added liquid.)

The pragmatic considerations used in choosing between these strategies include the existence of the plans that are suggested, (e.g., Is there an alternate plan such as VANILLA-CAKE that can be used instead of the initial VANILLA-SOUFFLE?), the relative difficulty of implementing the strategies, (e.g., Is it harder to convert the entire recipe to a cake than it is to use strawberry preserves instead of fresh strawberries?), and the degree to which the goals of the overall plan are met by the modification, (e.g., Does a soufflé made with preserves taste as much like strawberries as one with fresh strawberries and added eggs?). These and other questions, then, are considered in deciding which of the competing strategies will be used to repair the plan failure.

The third goal of the project involves using the explanations of plan failures to learn the features of the new domain that are predictive of negative interactions between plans. Features implicated in the explanation of a failure are marked as predictive of that failure. These features are generalized to the most abstract level that still allows them to be explained by the rules used in the initial explanation of a failure. Learning these features allows the planner to anticipate plan interactions before they occur, by noticing the existence of features in a given input that predict them. Using this mechanism, then, the planner can avoid repeating those failures that it has encountered before.

In the soufflé example, the features that participated in the failure, (e.g., the violation of a balance relationship, the fact that it was making a soufflé, the pulping of fruit, and the presence of extra liquid) are marked as predictive of this failure and linked together by processing demons which search for the presence of the other contributing features. When confronted with the goal to make a Kirsch soufflé, which has a few tablespoons of liquor, the fact that there is a goal to make a soufflé, and there is a feature that is predictive of a particular kind of failure, (i.e., liquid), the planner is able to access its strawberry soufflé plan, (previously

repaired with the modification of extra egg whites), and use it rather than another soufflé plan that would lead to a repeat of a known failure. Further, because these features are predictive of a particular failure and a particular type of failure, even in those cases where the past plan is inappropriate, the planner still has access to the knowledge that there is a potential failure that has to be planned for, as well as access to the knowledge structure, the TOP SE:DC:B, that will enable that planning.

## CONCLUSION

The learning that is done by the CHEF planner is based on the notion that the significant aspects of a plan can best be identified through a causal analysis of how that plan is designed to run. In explaining failures, then, the planner is serving two functions. It is deciding, on the basis of its explanation, which repair rules can be reasonably applied to the failure and it is discovering the features of objects and actions in its domain that tend to interact negatively. By combining these functions, then, and storing the repaired versions of failed plans indexed under the features that originally lead to the failure it is able to effectively access positive solutions to problems that it has already encountered and planned for.

# **INFERENCE OF INCORRECT OPERATORS**

## **Haym Hirsh and Derek Sleeman**

Stanford University, Computer Science Department  
Stanford, CA 94305

### **INTRODUCTION**

Past work in Intelligent Computer-Aided Instruction has demonstrated that incorrect problem-solving performance can be represented as variations on correct problem-solving performance [63, 347, 42]. Subsequent work [43, 201, 348] attempts to automate the discovery of incorrect problem-solving methods when given correct methods. The work described here takes two different approaches to inferring incorrect problem solving operators. The first system, INFER\*, discovers new domain rules by applying known rules in reverse, proposing new rules to fill in missing steps. A second system, MALGEN, applies perturbation operators to known rules to create new rules.

### **INFER AND INFER\***

Given rules of algebra, both correct and incorrect ("mal-rules"), an initial problem, and an observed answer, LMS [349] attempts to find the sequence of rule applications that result in the answer. However, not all possible methods of obtaining incorrect results are foreseen and represented by mal-rules, and in such cases LMS fails to discover a path from problem to answer. These exceptions were then studied, and new mal-rules were suggested by the researcher to explain the behavior. INFER [348] was the first attempt to automate this process.

INFER applies rules of algebra in reverse to the observed answer until a form similar to the original equation is reached. Since some rules can be applied backwards in an infinite number of ways, heuristics are used to choose among them. For example, applying the ADD rule in reverse to  $X=7$  could result in  $X=5+2$ ,  $X=1+6$ ,  $X=1000+-993$ , and an infinite number of other possibilities. However, if it was known that  $X=7$  was the result of an initial problem  $X+3=4$ , only a small number of the possibilities become reasonable. Special heuristics are applied to suggest specific numbers to use in the reverse application of the arithmetic operation, using the initial equation as the source of the suggestions.

In the given case, a heuristic would result in  $X=4+3$ , seeing that the

initial equation had a 4 on the right hand side. More precisely, application of ADD in reverse to  $X=7$  results in  $X=N+(7-N)$ , and heuristics are then used to provide possible values for N. This separates the heuristics from the domain rules. ADD, MULT, or any other arithmetic operation applied in reverse results in an equation with a variable that needs instantiating, at which point heuristics are used to find possible values.

Further reverse applications of algebra rules are done to intermediary forms; the process is repeated until an intermediary form matches the form of the original equation, or until no rules apply. A new rule is then generated between the last intermediary step and the target equation. In the preceding example,  $X=4+3$  would next go to  $X-3=4$ . Since this is of the same form as  $X+3=4$ , a new rule is suggested, namely  $X+M=N \rightarrow X=N+M$ . A more detailed description of the techniques used is given in [348].

The INFER mechanism described above has the implicit assumption that mal-rules occur at the first step of a solution, since it only infers new mal-rules at the first step of a solution sequence, after applying rules in reverse from the observed answer to the desired equation. However, mal-rules can be used anywhere in the solution process, and the inference mechanism should discover them as well. An enhanced version, INFER\*, does this by repeatedly applying known rules to the initial equation in the forward direction, resulting in subsequent steps along the solution path. The INFER algorithm is then applied between each intermediate state and the observed answer.

## MALGEN

MALGEN [147] was conceived as an approach complementary to the INFER\* system. The heuristics used in INFER\* proved to be somewhat domain dependent, and an alternate method was suggested, motivated by the fact that the algebra mal-rules correspond to correct rules with minor deviations. MALGEN formalizes this, using a set of perturbations that modify rules in "reasonable" ways to form "reasonable" mal-rules. Given correct rules, MALGEN applies the perturbations to form new mal-rules. This is repeated on the mal-rules, generating more mal-rules, continuing the process as long as desired.

To enable perturbations to be less domain dependent, a more robust rule representation was designed. A problem solving operator is a rule with four parts: adequacy conditions, correctness conditions, actions, and results. The adequacy conditions are the minimal conditions for a rule to be relevant, specifying that the rule can be used on a given state, if it is

correct to do so. Correctness conditions, on the other hand, check if it is appropriate to apply the operator. The actions are local computations to be done if both the adequacy and correctness conditions are satisfied. Finally, the result is the new state in the solution space.

In the preceding description, the only part that does not encode specific domain knowledge is the adequacy conditions. They test whether an operator is valid in a given state. It is the other three parts, correctness conditions, actions, and result, that encode the operators knowledge, and are hence open to deviation. Incorrect versions of correct operators will have the same adequacy conditions and differ in one or more of the other three parts. Hence perturbations are used on these three parts.

### Perturbing Correctness Conditions

The first class of errors is in the correctness conditions. Incorrect operators that differ in this part cause the specified actions to occur at an incorrect time or place. For example, an incorrect application of ADD might be  $2+3\times 4 \rightarrow 5\times 4$ , ignoring precedence of arithmetic operators. Another example is  $-2+3 \rightarrow -5$ , in which the leading minus sign is not handled correctly. In both these examples the correct action is done to an incorrect state. The correct rule for ADD would check for these circumstances, and rules to represent the incorrect versions above would be similar to the correct ADD, however with correctness conditions that are deviations from the correct ones.

With this in mind, MALGEN perturbs the correctness conditions for all rules by negating them, using methods similar to DeMorgan's Laws. It makes copies of the original rule with new correctness conditions, each representing a different way the original correctness conditions may fail. For example, given a conjunction " $A_1 \wedge A_2 \wedge \dots \wedge A_n$ " the negation mechanism generates  $n$  new conditions, and hence  $n$  new incorrect rules, " $\neg A_1 \wedge A_2 \wedge \dots \wedge A_n$ ", " $A_1 \wedge \neg A_2 \wedge \dots \wedge A_n$ ", etc., through " $A_1 \wedge A_2 \wedge \dots \wedge \neg A_n$ ", each corresponding to a different way the original rule may fail. Note that conjuncts are not simply removed; rather, they are negated, so that all versions of the rule are mutually exclusive - if one mal-rule can be used, the correct version of the rule, as well as the other new mal-rules, will not be applicable.

## Perturbing Actions

Another class of mistakes are those in which an action is performed incorrectly. This is fairly broad, and examples are performing one action instead of another, performing the correct action on wrong arguments, or perhaps not doing the action at all. For example, instead of doing MULT, multiplying two numbers, the action ADD, adding two numbers, might be performed. Or perhaps instead of doing  $2-3=-1$ , the student may do  $3-2=1$ . A further example is the rule that subtracts a number from both sides of an equation. It can be viewed as moving a number across the " $=$ " and switching its sign. However, students sometimes forget to change the sign.

Three forms of perturbations of actions are used: changing actions, changing arguments, and removing subactions. The first, changing actions, uses a list associated with every primitive that contains other primitives that are similar to it. MALGEN then forms new rules by copying a rule, switching all primitives to similar ones, one at a time. Changing arguments, the second form of perturbation, simply takes existing actions and switches their arguments. The last, removing actions, takes primitives with one argument and replaces it with the IDENTITY primitive, one whose value is just the argument itself.

## Perturbing Results

The final type of incorrect operators are those in which the result is returned incorrectly. These are generated by perturbations that take the general form of the returned answer and modify it slightly. Its motivation comes directly from the observed performance of high school algebra students, in which the result is written incorrectly. For example, given the algebra problem  $4X=2$ , some students transform this to  $X=4/2$ . This class of perturbations are more domain specific, since general perturbations could yield far too many new operators with little significance.

## FUTURE WORK

INFER\* generates many possible mal-rules from a single instance of incorrect behavior, and likewise MALGEN has the ability to generate many mal-rules from a single rule. The use of a filter on the mal-rules generated will be explored. Such a filter is a necessity for INFER\*, in which some of the new mal-rules are obviously unreasonable. However, MALGEN creates reasonable mal-rules from existing rules, since the perturbations

used are reasonable. A filter would prove useful for MALGEN anyway, since less reasonable perturbations could be used, in the hopes of generating further mal-rules.

The filter is of necessity domain specific, and in algebra would use such knowledge as the fact that students never write  $X=4/$ , i.e., they have an understanding of the general form of answers. Another approach is to study the performance of teachers in "modeling" students, and use this to generate a filter of what the teacher does not even consider in evaluating the student's performance.

## ACKNOWLEDGMENTS

This research was funded by a grant from IBM. The work has been influenced by discussions with Bruce Buchanan, Pat Langley, and the GRAIL group at Stanford. Helpful suggestions on this paper were provided by Paul Rosenbloom. In-Yung Kim assisted in the development of INFER\*.

# A CONCEPTUAL FRAMEWORK FOR CONCEPT IDENTIFICATION

Robert C. Holte

Dept. of Electrical Engineering and Electronics, Brunel University  
Uxbridge England UB8 3PH

## ABSTRACT

This paper describes the status of a project [150] to develop a conceptual framework, called **CF**, which is applicable to all "practical" systems for a specific kind of learning, **concept identification**. Most of the concepts in CF have resulted from an analysis of the concept identification task and the influence of representation on the effectiveness of a learning system. The generality of CF is examined and possible applications are described.

## INTRODUCTION

A **conceptual framework** is a way of looking at and thinking about the phenomena (events, entities, etc.) in a given domain. It consists of a set of **concepts** and a set of precise **interpretation rules** which define the valid applications of the concepts to phenomena in the domain. A well-chosen conceptual framework exposes the "causal" underpinnings of a domain, and by doing so provides a firm foundation on which to build **explanatory principles** which can be used to explain, predict, and unify diverse phenomena in the domain.

## RELATED WORK

Related work includes proposals for "models" of learning [229, 352], taxonomies of learning systems [92], comparative analyses [50, 93], and the many explanations and claims about specific patterns of behavior in particular classes of learning systems found throughout the literature.

[16, 148, 215, 246, 248] have directly contributed to CF. Each introduces concepts which are indispensable in analyzing or explaining the behavior of a learning system and demonstrates their usefulness in a particular context. In CF these concepts have been extended, given precise definitions, and integrated with all the other concepts in CF.

## A TASK ANALYSIS OF CONCEPT IDENTIFICATION

Concept identification is a typical concept learning task (see [149]): labelled, noise-free examples are presented incrementally to a system which outputs a single concept, the **hypothesis**, whenever new examples become available. There are many possible **criteria of success** for a system operating in this kind of **informational milieu**. To be successful at concept identification, a system's output must converge to a hypothesis which correctly classifies all possible examples, even those not yet presented.

This choice of criterion of success is of enormous consequence. It requires the learning system to find or construct a concept which correctly classifies all possible examples. Because the learning system does not initially have sufficient information to determine whether or not a given concept satisfies this requirement, its primary concern is to gather information which will allow it to distinguish those concepts which correctly classify all possible examples from those which do not.

Concept identification may be cast as a search problem [248], but it is primarily a search for a correct characterization of a "solution set" in the space of candidate concepts, and not a search for a concept inside a known solution set. This shift of emphasis from the individual concept to sets of concepts permeates CF. Thus, the properties and behavior of a concept are of interest to a concept identification system only to the extent that they indicate trends in the behavior of a set of concepts: in [156] the concept which obtains the most useful information is one known to perform badly.

## THE REPRESENTATION OF CONCEPTS AND CONCEPT SPACE

A **representation** is a mapping between a concrete object, such as a string in a formal language, and an abstract entity. Both individual concepts and sets of concepts must be assigned representations in a practical learning system. The effectiveness of a learning system is heavily influenced by the choice of representation for individual concepts. The choice of representation for concept space has an equally strong influence deriving from a different source. Both "sources of power" have been included in CF.

The mapping between an individual concept and the string which represents it is decomposed in CF into a mapping from the concept to its **declarative aspect**, and a mapping from the declarative aspect to the appropriate string. A declarative aspect of a concept is anything which

uniquely identifies it; usually it is defined in terms of a small set of attributes of the concept.

The choice of declarative aspect is the choice of which properties of concepts will be accurately accessible through syntactic operations on the strings which represent them. If the right choice is made strings which are syntactically similar will correspond to candidates which share properties or stand in relations which play an important role in learning.

The set of candidates is usually represented by a procedure which generates the strings representing the individual concepts. The strings will be generated in a structured manner, e.g. all strings of "size M" may be generated before any of "size M+1". The structure of the generation process defines a **generative structure** of candidate space.

The choice of generative structure determines how effectively a learning system will be able to exploit information indicating a strong trend in the behavior of a particular set of concepts. The behavior of some systems, especially enumerative systems (e.g. [389]), can be analyzed to a considerable degree on the basis of generative structure alone.

## THE GENERALITY OF THIS CONCEPTUAL FRAMEWORK

CF is intended to provide an analytically useful account of the operation of all practical concept identification systems. But the usefulness of its concepts has typically been demonstrated for just one kind of learning system: will they be equally useful for analyzing other kinds of learning system?

Knowing that these concepts are useful for one kind of system, one could produce evidence for their general usefulness if one could demonstrate that, for the purpose of analysis, many learning systems are of this "kind". With this reasoning in mind, the notion of "enumerative system" has been extended to encompass several kinds of systems. It is thus becoming apparent that many of the distinctions commonly used to define "kinds" of learning system have little analytical significance.

Alternatively, one might be able to produce evidence of the general usefulness of CF's concepts by arguing that these concepts involve abstractions of systems which transcend the boundaries between "kinds" of systems. This source of evidence has produced encouraging preliminary results.

## APPLYING THE CONCEPTUAL FRAMEWORK

The next step is to test CF by using it to assess specific claims and anomalies in the literature, and by performing experiments which test its predictions. Many of CF's concepts are expected to prove usefully applicable to learning tasks other than concept identification. CF will be applied to two learning-related projects at Brunel: the application of genetic algorithms to VLSI chip design [125], and the writing of a process scheduler for a VLSI wafer fabrication plant.

## ACKNOWLEDGMENTS

I gratefully acknowledge the financial support which made this work possible: a War Memorial Scholarship from the Imperial Order of the Daughters of the Empire (Canada); Postgraduate Scholarships from the Natural Sciences and Engineering Research Council (Canada); and Overseas Research Student Scholarships from the Committee of Vice-Chancellors and Principals (UK).

# **NEURAL MODELING AS ONE APPROACH TO MACHINE LEARNING**

**Greg Hood**

Carnegie-Mellon University, Department of Computer Science  
Pittsburgh, PA 15213

## **ABSTRACT**

In this paper I propose that a neural modeling approach is reasonable for investigating certain low-level learning processes such as are exhibited by invertebrates. These include habituation, sensitization, classical conditioning, and operant conditioning. Recent work in invertebrate neurophysiology has begun to provide much knowledge about the underlying mechanisms of learning in these animals. Guided by these findings, I am constructing simulated organisms which will display these basic forms of learning.

## **WHY NEURAL MODELING?**

Neural modeling, to most AI researchers, brings to mind the largely unsuccessful efforts in the early sixties to build networks capable of learning. Most of these models were based on idealized units such as the perceptron [239]. Those units bear little resemblance to present-day models of how neurons actually function. A common belief was that by simply combining large numbers of those units, intelligent behavior would naturally follow. This was based on the notion that since the connectivity of the neurons in the brain appeared quite random, connectivity was relatively unimportant in constructing neural networks. In the two decades since that time, much information has been gathered about the details of neural functioning, the ways in which neurons may be combined to produce behavior, and the basis of learning in these systems. With this new information from neurobiology I believe that it is time to reexamine neuron-like systems as a means of exploring machine learning.

Many of the studies of the neurophysiology of learning have been performed on invertebrates. This is due to the relative simplicity (several orders of magnitude, in cases) of their nervous systems as compared with those of vertebrates. Another reason is that because of the large size of the major neurons, it becomes fairly easy to identify certain neurons across individuals within a species and even across closely related species. This

facilitates the investigation of the effect of different environmental conditions on a particular learning task. These studies include those of Kandel and coworkers on the marine snail *Aplysia californica* [162], Alkon on the marine snail *Hermisenda crassicornis* [2], and Krasne on the crayfish [187].

Why should invertebrate studies be relevant to understanding learning in humans? It appears that in the evolutionary process processes and structures are typically modified rather than replaced with totally different processes and structures. If such is true for the nervous system, we should expect that human neurons would share some of the properties of invertebrate neurons. Each one may of course have developed specializations which are not present in the other, but nevertheless it is likely that similar types of learning in each organism should have similar underlying physical processes. So it is reasonable to ask if there are aspects of learning which are common to both humans (or vertebrates) and invertebrates. It has been found [163] that marine snails are able to exhibit three fundamental types of learning which psychologists have studied in vertebrates for many years: habituation, sensitization, and classical conditioning. Operant conditioning has also been found to occur in several types of insects including locusts [153], grasshoppers [153], and fruit flies [105]. Understanding these forms of learning in invertebrates would be a first step toward understanding similar forms in higher animals and man.

One approach to machine learning is then to mimic the identified processes of learning which take place in the nervous systems of invertebrates. To do this requires a fairly accurate simulation of the functioning (i.e., performance without learning) system upon which learning mechanisms may be imposed. This underscores the importance of using "real" neural elements instead of idealized ones. For if we develop systems which bear little relation to the understood natural systems, then we cannot use experimental results to guide in the design process. The knowledge gained by performing accurate simulations, moreover, will in turn assist in corroborating the correctness of existing neurophysiological theories of learning and may also guide in the selection of novel experiments to extend these theories.

One may legitimately wonder whether working at a low level will yield benefits not present at a more abstract symbolic level. My feeling is that there are insights to be gained from the low-level study of learning, if only to reinforce existing ideas within the field of machine learning such as generalization and discrimination. Low-level processes may, however, turn out to be very important in some areas such as recognition since they seem more compatible with the notion of partial matching than symbolic

processing. These are open questions.

## A SIMULATED ORGANISM

To test the approach advocated above, an organism is being developed which is controlled by a network of units with properties similar to the known properties of real neurons. The simulated environment in which this organism may move and function is the World Modeling System, which is described in detail elsewhere [Carbonell (this volume)]. The body of the organism is a simple cylinder which may slide around on the floor of the simulated world, in which there may be obstacles and other organisms. The interface between the neural network and the environment is handled by a sensory-effector interface (SEI) which controls the frequency of firing of certain neurons (sensory neurons) to match the strength of associated stimuli from the tactile and visual senses. The SEI also translates the frequency of firing of other neurons (motor neurons) into forces which are applied to the body of the organism.

Performing experiments on such a simulated organism within a simulated world is much easier than performing experiments on natural organisms such as the marine snail. First, it is possible to make arbitrary internal measurements without disturbance to the organism. For example, one can record the synaptic transmissions (the generated postsynaptic potentials) without isolating segments of the nervous system, and so an "intact" organism can be studied in the midst of some activity. A second advantage is that the experiments are repeatable, and can be rerun under exactly the same conditions. This means that the effect of a certain change can be determined without uncertainty over whether the effect was caused by a small change in environmental conditions.

It is difficult to decide exactly how faithful one should be in the simulation of known neural functioning. I have argued above that it is advantageous to closely emulate the natural mechanisms of learning. However, there are several factors which limit the level of detail at which the simulation may occur. The computational power of available computers puts a large constraint on the simulation detail since if we are to investigate networks of any reasonable size (perhaps 100 neurons), then the time to do the simulation must not be excessive. Another limiting factor is the current knowledge about the underlying biochemical processes of neural functioning, especially those which produce learning. Since these processes are not fully known, we can at best mimic their measurable consequences. Even if these processes were completely understood at a single site, we still do not have the same level of knowledge for all parts

of an invertebrate nervous system. This makes the accurate duplication of an entire system impossible. Finally, it appears that below some level the details are uninteresting from the perspective of machine learning.

The level at which I have chosen to build a neural simulator is the voltage level. In the simulation the "voltage" of a neuron roughly corresponds to that found at the axon hillock of a real neuron, which is the usual site at which neural firing is initiated. If the voltage level at this point in the simulated neuron exceeds a certain critical value, the *threshold*, the neuron is considered to fire. The firing of a simulated presynaptic cell causes several current channels to open in the receptor of the postsynaptic cell. Each of these current channels loosely corresponds to a type of ion channel in a real neuron.<sup>1</sup> These current channels close with a characteristic half-life so the current pulse injected into the postsynaptic neuron is a pulse with very fast rise time and exponential decay. These current pulses are summed spatially and temporally in the postsynaptic cell to determine the net current influx or efflux, which is used together with the previous voltage to determine the new voltage. The voltages of all neurons are updated at fixed intervals of time (on the order of 1 millisecond<sup>2</sup>) so the simulation is done synchronously and is easily suited for implementation on a highly parallel computer.

There are many difficulties in studying large networks of simulated neurons. One is the length of time required to perform simulations on current computers. Certain techniques have been found to permit faster simulation. A key idea is that, with not unreasonable assumptions, the entire history of a neuron's synapses (the state of all receptive current channels) can be represented by a short vector, whose length is proportional to the number of *types* of current channels instead of the number of channels themselves. Another problem is that the specification of a large network becomes very difficult because great number of parameters which must be specified for each neuron and for each synapse. As a partial solution to this problem, a specialized language for describing neural networks has been developed which allows one to describe a network compactly in a hierarchical manner much as a complex serial process can often be decomposed into a small set of procedures which can be called

<sup>1</sup>Ion channels in real neurons are small areas of membrane which selectively permit the flow of certain types of ions across the membrane. The gating mechanisms of an ion channel may be voltage-sensitive (as in the case of propagation of the action potential) or chemo-sensitive (as in the case of postsynaptic receptors).

<sup>2</sup>These timesteps are unrelated to and usually much smaller than the timesteps of the world simulation.

repeatedly.

Several behavioral properties are desired of the organisms which are being constructed. One is that the organism should be long-lived as opposed to being only capable of a one-shot learning task. An organism which has been interacting with other objects in a simulated world should become better able to function in it due to its accumulated experience. The organism should also demonstrate the capacity for several fundamental types of learning: habituation, sensitization, classical conditioning, and operant conditioning. The important point here is that a single organism should be able pass tests for all of these types of learning, not just that it is possible to build one organism capable of habituation, another capable of sensitization, etc. This is not a trivial problem because of the probable interactions among these several types of learning. As an example of a conditioning test, consider an organism which initially has a reflex in which the activation of certain "pain" receptors by sudden pressure triggers an escape response. A classical conditioning test would be to repeatedly present the organism with a previously neutral stimulus such as a particular sound immediately prior to being struck by another object. After a period of time the sole presentation of the sound should elicit the escape response.

Certain characteristics are frequently observed in the nervous systems of natural organisms, and are probably necessary for interesting learning behavior. The organism should have a repertoire of built-in reflexes and behaviors upon which to build. These must be designed into the nervous system from the start. It does not appear likely that a *tabula rasa* approach will work here any more than it did for the early neural network research. Another design constraint is the need for building in pain and pleasure circuitry for the reinforcement of behavior. Clearly an organism without this will not have any motivation to avoid adverse situations and to attempt to attain pleasant ones. These appear to be among the ultimate goals of an organism from which most short-term goals are derived directly or indirectly. Another characteristic sought is a predictive capacity so that the organism may anticipate sensory information before it is actually received. With this capacity, incoming sensory data may be matched at a low-level to the expectations, and any discrepancy may be relayed onto higher levels to redirect the focus of attention so as to make corrections in either its behavior or its predictive apparatus. This permits the organism to extract interesting events from the vast amount of incoming sensory data.

## CONCLUSION

Is there hope of building an organism whose behavioral versatility and robustness are comparable to that which has evolved in even a simple natural organism such as the marine snail? There are two reasons why I am hopeful. First, although our knowledge of any nervous system is far from complete, many patterns are beginning to emerge. These include the ways in which neurons may be combined to yield more complex behavior, and specific mechanisms of learning whereby synaptic effectiveness is modified. The second reason for hope is that in our simulated world an organism does not face same evolutionary pressure that occurs in the real world. It does not have to feed, grow, and reproduce as fast as possible in the midst of predators in order to preserve the species. Thus, our hand-constructed organisms do not have to possess all of the redundancy or behavioral nuances, and so may need far fewer neurons and synapses than their natural counterparts. These simulated organisms, although certainly not able to compete with natural organisms, may help elucidate the mechanisms of learning which are possible in a neural system and provide insights on how these processes work in nature.

## ACKNOWLEDGMENTS

I thank Keith Barnett, Jaime Carbonell, Klaus Gross, Mike Mozer, Alain Rappaport, and Hans Tallis for their valuable comments on this paper and in the earlier discussions of these ideas. This research was sponsored in part by ONR grants N00014-79-C-0661 and N00014-82-C-50767, ARPA grant F33615-84-K-1520, and an NSF graduate fellowship.

# STEPS TOWARD BUILDING A DYNAMIC MEMORY

Larry Hunter

Department of Computer Science, Yale University  
New Haven, CT 06520

## INTRODUCTION

AI researchers have come to precious few agreements about their field, but there is at least one area where we have reached consensus: AI programs require vast amounts of knowledge, organized in a manner that facilitates its use. Not only must such a memory be able to retrieve relevant information for a wide variety of tasks, but it must be able to add to and modify itself: no useful body of knowledge is ever complete and unchanging. An expanding system of knowledge, organized to facilitate retrieval for a variety of disparate tasks will be referred to here as a *dynamic memory* [319]. The diversity of recall strategies is an important constraint on the organization of knowledge in the system. The categories that are used to identify similar memories may differ from task to task. New connections between categories or between primitives and categories may be required for different strategies. It is an important goal of the memory building system to prepare for different kinds of recall. A good design for one type of recall may interfere with the design for another. Designing for many types of recall tasks can place very strong demands on a storage system, thereby directing the evolution of the system.

Another important source of direction in good design of a dynamic memory system is the size of the memory. Previous implementations of such systems have remembered relatively few 'experiences' [206, 177]. Using relatively small memories removes a great deal of the pressure on the organization scheme, since relatively simple search techniques can be used to find relevant memories.

The goal of this project is to design, build and test a dynamic memory system that uses multiple recall strategies to access a large body of knowledge. The domain chosen to build the system in is lung tumor pathology. The memory system is to eventually encompass a collection of 9255 images stored on a videodisk, brief diagnosis and comment on each image, broader descriptions (including histologic and cytologic characterizations) of a significant percentage of the images, abstracted case histories of the patient when available (approximately 20%) and a base of

'textbook' knowledge of pathology and histology necessary to represent the images. Initially the project has taken a small sample of these, begun with the background knowledge, added the image descriptions, and then had the system organize the new material and recall images using three different recall strategies.

## DOMAIN DESCRIPTION

The domain of lung tumor pathology was chosen, principally because the image collection and expertise of Dr. Raymond Yesner. The images, which illustrate various aspects of lung tumor pathology, have been pressed onto a videodisk which has been integrated with an Apollo workstation. Sketchy image descriptions and patient case history information are being entered. Protocols of diagnostic consultations have been analysed yielding important information on the memory processes of pathologists at various levels of expertise. These protocols have also been used to generate detailed descriptions of images, including all the features that expert pathologists attend to during understanding. The sample used in the initial program consists of 40 images, 14 with detailed descriptions, 9 case histories and enough background knowledge to characterize some of its important aspects.

Pathologists classify abnormal tissue on the basis of functional and morphological changes. Biology is subject to much inherent variation, and the pathology of a poorly understood biological process like neoplastic change is even more subjective. Experts call upon their broad experience to identify many exceptional cases; one ultimate goal of this project is to make this kind of expertise available to nonexpert pathologists. The kinds of questions that expert pathologists find themselves answering (as uncovered in our protocol analysis) fall into four categories.

## RECALL STRATEGIES

In the most common questions, a pathologist specifies a category of lung tumor that he would like to know about. He may specify a variety of specific subcategorizations or particular features or combinations desired. An expert clearly must have his memory organized to facilitate this most basic recall strategy, which can be thought of as a top-down traversal to a memory.

A second recall strategy is needed to cope with biological variation, which nearly always produces phenomena different from its classical description. Uncertainty in a pathologic diagnosis may be due to this

variation, so a pathologist may want to know the range of permissible variation within a given class of tumor. These questions generally follow the expert's description of a prototype (eg. "Is it possible for these kind of cells to show more abundant cytoplasm?") and can be metaphorically thought of as a horizontal traversal. The goal of this recall strategy is to recover memories that differ from an existing description along a particular dimension.

A third traversal type is called for when the pathologist encounters a tumor with distinctive features, but does not recognize it. He would like to tell the system the features that he sees and have the system recognize the entity. The system should respond with a prototypical instance of the disease and a description (including the name, of course). This kind of traversal is a form of the classic AI problem of recognition.

Finally, if the system recognizes several diagnostic possibilities or the pathologist is unsure or confused about what diagnostic category an entity belongs to, the system should be able to provide additional criteria upon which the the various possibilities could be differentiated. This kind of traversal requires finding the features of both (all) the possibilities and picking features that distinguish between them. This strategy is often identified with differential diagnosis, but also includes the identification and avoidance of confusions.

## MEMORY MODIFICATION

There are two major forms of memory modification in the current program: adding new experiences (predominantly an issue of indexing them appropriately) and modifying (or adding or deleting) categories. Since both processes are intricate, they will only be described briefly here.

The indexing system is driven by a set of indexing rules that range greatly in complexity and generality. Each recall strategy dictates a particular set of indices be built. Some indexing rules are directly associated with a particular strategy (eg. building back pointers in top-down descriptions to facilitate recognition). Other indexing rules are useful for more than one strategy (eg. noting features where a case differs from a prototype is useful in both top-down and horizontal retrieval.) Indexing rules may have to invoke complex processes in order to accomplish their tasks. For example, when a case has been misclassified, the indexing rules that attempt to handle possible confusions must find a set of features that *predictively* distinguish between the categories [321]. If this data were present in the initial descriptions, the mistake would not have been made, so the program may have to reexamine previous cases or even ask

questions to find appropriate criteria.

Many indexing rules require less processing. While some kinds of mundane indexing are straightforward (eg. building back pointers so that a categorical description can be used for recognition), we have developed a growing list of mundane indexing rules crucial for recovering knowledge in a form different from the way it was originally stored. One important mundane indexing rule builds indirect indices. These are links not from memory to memory, but from a memory to a set of indices. Indirect indices may be thought of as classifying other indices, making it possible for traversal schemes to refer to the class of indices it wants to traverse instead of the names of particular indices. This capability is particularly useful in recognition.

Attempting to handle large numbers of memories and indices had led to some useful heuristics. For example, the number of indices between memories tends to grow faster than the number memories, yet traversal time remains constant, at least in experts. The data structures used to represent indices must therefore have the quality that the time to traverse a known index must not depend on the number of indices at the node. There are several data structures that meet this criterion (eg. hash tables), but many commonly used association techniques (eg. lists) do not.

Indexing rules are a way to make memory organization principles explicit. Since memory organization is provisional and based on incomplete information, there are times when the rules will be wrong. As the system evolves, it will become important to detect and correct indexing failures (i.e., misclassifications). Not only might the particular failure be remedied, but the rule that build the inappropriate index might itself be modified. There are only two possible kinds of indexing failure: either an irrelevant memory is found or a relevant memory is missed. Of the two, only the irrelevant recall is possible to correct. If a system could detect somehow that it had relevant knowledge that it wasn't using, it could, by the same mechanism, find that knowledge and use it. Since recovery is possible from irrelevant recall (caused by overindexing), but not from missed recall (underindexing), it seems reasonable that indexing rules should trigger liberally. Future work includes automatically modifying incorrect indexing rules.

Some categories index to many cases. As the number of examples grows, the category may become overloaded. When a numeric threshold is crossed, the program tries to split the category into meaningful subcategories. This process has been attempted using common feature analysis to determine the subcategories [206], but there are problems with that approach that make some causal analysis of the subcategorization seem inevitable [321]. This, along with building new categories based on

confusions or failures is an important direction for future work.

## CONCLUSION

The attempt to build useful dynamic memories is usefully constrained by requiring the system to support flexible recall and large numbers of experiences. The rules for building up such a system are best made explicit, and they vary in complexity and computational requirements. There are likely to many indexing rules, and it is possible that there will be indexing rule failures. There must also be mechanisms for modifying the classification scheme used as new memories are added. These issues are being explored in a domain that demands flexible access to a large amount of information.

# **LEARNING BY COMPOSITION**

**Glenn A. Iba**

GTE Laboratories  
Waltham, MA. 02254

## **ABSTRACT**

An approach to learning based on structural composition provides the basis for much of my current research work. The focus is on how existing structural elements can be composed to form larger structural elements. The importance of *uniformity of representation under composition* is stressed. A general learning framework is discussed which includes mechanisms for automatically proposing new compositions. Filters are used to select only the best of the proposed compositions. This framework is currently being explored in the domain of puzzle solving, where operators are the elements being composed to form macro-operators. This work should have important implications for automated mathematical discovery, automatic programming, and concept learning.

## **INTRODUCTION**

Many domains share the property of possessing a compositional structure. They have elements which can be composed to form macro-elements. These macro-elements may in turn be composed to form still larger macro-elements, resulting in a compositional hierarchy. Example domains include concept learning, problem solving, mathematical theorem proving, and computer programming. The respective elements for composition are concepts, operators, theorems, and programs. The natural learning problem is that of discovering and utilizing new composite structures. This short summary paper provides an overview of the learning framework being explored for these areas. Current work is focused on the discovery of macro-operators in problem solving, using a variety of puzzles as the vehicle for exploration.

## **THE GENERAL LEARNING FRAMEWORK**

The general learning framework consists of three parts:

1. Performance system

2. Set of structural elements

3. Learning system.

The performance system makes use of the currently available set of structural elements to perform its task. The learning system creates new structural elements by composing members of the existing set of elements, and makes the new elements available to the performance system by adding them to the existing set. In order for the performance system to utilize these composite structures, they must be represented in the same format as the previous structural elements. I term this requirement *uniformity of representation under composition*. This property is necessary in order to permit the separation of the learning and performance systems. It should also be mentioned that, although the collection of structural elements is referred to as a set, nothing precludes this set from having structure imposed on it. For example, concepts could be structured into a generality hierarchy, and operators could be incorporated into a difference-operator table. All that is required is that the learning system be able to add new elements in such a way that the performance system can make use of them.

The learning system itself consists of three components: structure proposer, static filter, and dynamic filter. The learning system is periodically activated to propose new composite structures according to various heuristics. The proposed structures are then analyzed by a static filter, which makes an estimate of the utility of the newly proposed structure. If the estimate surpasses a given threshold, then the new structure is added to the set available to the performance system. Performance statistics are maintained on new elements as they are used by the performance system. Finally, a dynamic filter is periodically called to eliminate those elements which do not prove themselves in actual practice. The importance of the filters is that they prevent the performance system from becoming bogged down under the weight of an ever-growing set of structural elements. A new element must prove itself to be sufficiently useful to compensate for the additional cost to the performance system entailed by its presence.

## DISCOVERY OF MACROS IN PROBLEM SOLVING

The initial work in this area, which forms the topic of a Ph.D. dissertation in progress, is concentrated on the discovery of new macro-operators in the domain of puzzles. This work provides an alternative to the approach of Korf [181] whose macros are more specific than those considered here. Within the learning framework discussed above, two

performance systems are currently under study: best-first search, and goal-directed search. Puzzles that have been used for testing include Hi-Q peg solitaire, and Tower of Hanoi. Extensions to slide-jump, Rubik's Cube, and the 15 puzzle are being considered. The primary heuristics for the macro proposer are peak-to-peak (with best-first search), and subgoal completion (goal-directed search). The static filter is based on general estimates of simplicity, power, and applicability. The dynamic filter utilizes statistics on frequency of applicability, frequency of applications leading to progress, and frequency of occurrence in solution sequences. These techniques have led to some demonstrations of simple learning in variations of the combinatorially difficult Hi-Q peg solitaire puzzle. Some of that work is to be reported in the current IJCAI-85 proceedings [157].

# **KNOWLEDGE ACQUISITION: INVESTIGATIONS AND GENERAL PRINCIPLES**

**Gary S. Kahn**

Department of Computer Science, Carnegie-Mellon University,  
Pittsburgh, Pa. 15213

## **ABSTRACT**

After describing research goals in knowledge acquisition, a summary of the MORE system is provided. MORE interviews domain experts for information of diagnostic significance, and provides a shell for building diagnostic systems. This is followed by a summary of current research issues.

## **RESEARCH GOALS**

An obstacle to the development of knowledge-based systems is the cost of acquiring and representing domain specific information. I am exploring the design of automated systems for knowledge acquisition. Such systems must have the ability to represent domain knowledge and to use that knowledge in the service of both solving problems and acquiring more knowledge. Investigations to date have focused on diagnostic domains and the development of MORE [160], a system that actively interviews domain experts for information of diagnostic significance.

## **MORE: A Knowledge Acquisition Tool**

Like TEIRESIAS [83] and the EXPERTISE TRANSFER SYSTEM [29], MORE is designed to facilitate interviews with domain experts. It differs from these systems both in the strategies it uses, as well as in its reliance on a model-theoretic approach to the acquisition of diagnostic knowledge. The underlying model is characteristic of that presumed to play a role in everyday diagnostic reasoning. It involves the classical concepts of causal and consequent events, as well as prior, posterior, and conditional likelihoods. It entails the ideas of background conditions which effect these likelihoods, expressed qualitatively in terms of the diagnostic significance of observing or failing to observe symptoms associated with hypothesized events.

MORE's general understanding of the factors which contribute to the diagnostic significance of symptoms is used to seek out information which can give leverage in a diagnostic task. Inquiries range from superficial attempts to determine the nature of background conditions believed to make the occurrence of a particular cause more likely, to more sophisticated probes -- for example, ones that determine if a symptom which may result from one of several causes does so as a consequence of distinguishable underlying causal paths. MORE's strategies for acquiring knowledge are more completely described elsewhere [161]. MORE's selection of strategies is guided by an analysis of the information already acquired. As new information is entered, it is integrated into MORE's gradually evolving domain model.

MORE has a procedure for mapping domain knowledge of the kind described above into diagnostic rules. The domain expert is asked to associate confidence factors with each rule. MORE's domain model supports qualitative expectations on the relative significance of rules. If users violate these expectations, MORE issues warnings. Once a rule set has been created, MORE's diagnostic interpreter can be used on sample cases to test the current knowledge base. The interpreter is designed to do both forward and backward chaining.

A MORE prototype is currently being used to build diagnostic systems that respectively analyze disk faults, computer network failures, and defective circuit boards. Initial indications are that MORE provides a reasonable tool for building diagnostic systems in these domains.

## CURRENT RESEARCH

The initial MORE system was limited in a number of respects and research into its improvement continues as a collaborative effort between myself, John McDermott, and Larry Eshelman. In particular, we are enhancing MORE with the goal of giving it a role in analyzing its diagnostic failures. The result of such an analysis would be either a recommendation to the user to focus on providing more information of a specific kind, or a modification of MORE's internal domain representation. The latter might involve altering confidence factors, or removing independence assumptions.

Secondly, we are improving MORE's underlying representation in order to support more selective and pointed probing during diagnostic sessions. The original MORE was much more intelligent as a "knowledge engineer" than as a diagnostician. It failed to use its domain knowledge to guide its search for significant diagnostic information, exhaustively asking relevant questions.

Thirdly, we continue to work on improving the user interfaces to MORE. Although our research is essentially involved with underlying representations and control strategies, weak interfaces try the patience of the most dedicated domain experts. Finally, we are broadening the system to correct for a number of original simplifying assumptions. In particular MORE is being provided with a richer set of mechanisms for inferring and acquiring data. Among these are database interfaces and more sophisticated computational capabilities.

## CONCLUSION

MORE's power comes from understanding the nature of the considerations which affect assessments of diagnostic significance. By pursuing interview strategies which focus on such knowledge, MORE makes effective use of a domain expert's time. Unlike human knowledge engineers, MORE can rapidly access and interpret a knowledge base of new and unfamiliar information.

MORE demonstrates a potential path for the development of knowledge acquisition systems. Such systems need to structure domain knowledge in such a way that it can be used to both solve problems and acquire more knowledge relevant to the tasks at hand.

# **PURPOSE-DIRECTED ANALOGY: A SUMMARY OF CURRENT RESEARCH**

**Smadar Kedar-Cabelli**

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

## **ABSTRACT**

Existing techniques for analogical reasoning are based on mapping some underlying causal network of relations between analogous situations. However, causal relations relevant for the purpose of one analogy may be irrelevant for another. We describe here a technique which uses an explicit representation of the purpose of the analogy to automatically create the relevant causal network.

## **RESEARCH PROBLEM**

A fundamental open problem in artificial intelligence research on analogy is to determine which aspects of two situations are relevant to forming analogies between them. The traditional approach to analogy [110] was to examine surface features of two situations, and compute a partial match between them: the closer the match, the closer the analogy.

This approach did not satisfactorily model analogy, however, because situations which are quite dissimilar in their surface features can be made analogous at a deeper level of meaning: they may share some underlying causal network of relations [399, 129, 52, 57]. Consider the following analogy from [129]: "the hydrogen atom is like our solar system". To understand this analogy, the network describing the physical mechanisms that cause the planets to revolve around the sun is mapped to the atom to explain why the electrons revolve around the nucleus of the atom. Note that the analogy is not intended to teach us that the nucleus of the atom is 'yellow, hot or massive' like the sun. These 'surface' attributes are not involved in the causal network, and are therefore considered irrelevant to the analogy.

Often, however, there are numerous causal networks describing the situations. Which are the relevant ones to map when performing a particular analogy? Causal relations relevant for the purpose of one analogy may be irrelevant for another! Consider, for example, a different analogy with the sun: the metaphor "Juliet is the sun", derived from

Shakespere's *Romeo and Juliet* (also described in [129]). We know the context in which this metaphor is conveyed; that Juliet is a woman and Romeo loves her. The purpose of this metaphor is to analogically convey positive qualities about Juliet, not to convey anything about physical mechanisms! Thus the causal network about the sun which was supplied for the previous analogy is no longer relevant.

## APPROACH AND RELATED WORK

A more robust model of analogical reasoning needs the ability to automatically focus on the relevant causal network. Our thesis is that explicit knowledge of the purpose of the analogy can provide this focusing ability. The aim of the research is to develop a novel technique, Purpose-Directed Analogy, to demonstrate the thesis. The research described here adapts techniques for performing goal-directed and explanation-based generalization developed recently (e.g. [249, 166, 401]). One key feature of these techniques is that the relevant aspects of a *single* example can be extracted by generating an explanation of how the example satisfies a particular goal, or purpose. Because only one example is analyzed, alternate ways of satisfying a particular purpose cannot be learned. Our work on analogy addresses this limitation by extending the approach to *multiple* examples.

## RESULTS

We have designed and begun implementation of a system that incorporates Purpose-Directed Analogy (see [164] for more details). In particular, we are developing a system to learn concepts of everyday artifacts by analogy to known examples of the artifacts. The system uses a specialized notion of 'purpose': the purpose for which these artifacts will be used. Two examples are considered *analogous* if they share the network of relations which demonstrate that both of them can be used for the same purpose. For example, assume an agent wants to learn the concept HOT-CUP: objects that can be used for the purpose of drinking hot liquids. One way to learn the concept is to be able to determine if a new example (a styrofoam cup, say), is analogous to a known, prototypical example (a ceramic mug) in ways relevant to the purpose of a HOT-CUP. Given a different purpose (ornamental or religious, say), a different network of relations would be relevant.

The work described here is most closely related to Winston's [401], where the relevant *structural* features of an example of an artifact are

extracted by explaining how the example satisfies some pre-defined *functional* features. We extend this work by providing the ability to automatically derive relevant *structural and functional features* from an explicitly given *purpose*.

A system for performing Purpose-Directed Analogy would take as input the name of the concept, its purpose, a new example, and a domain theory<sup>1</sup>. Then, by analogical reasoning to a known example of the concept, the system would determine whether the new example is a member of the concept.

The system would first *retrieve* a known example of HOT-CUP (a ceramic mug). The system would then *explain* to itself how this ceramic mug satisfies the purpose of a HOT-CUP. Artifacts can be viewed as objects designed to enable people to perform certain actions (chairs to sit on, pens to write with, etc.). More precisely, using AI planning terminology, if the purpose of an artifact is to *enable an agent to perform a goal action*, then the artifact will satisfy the purpose if its structural features *enable a plan of actions* leading to the goal. It will enable a plan of actions if it *satisfies the preconditions of the actions* in which it is involved. So in this example, a ceramic mug will enable an agent to drink hot liquids if it enables the preconditions of actions in a plan leading to DRINK: that is, if it enables the agent to PUTIN the hot liquids (i.e. pour), KEEP the hot liquid in the cup for some interval of time, GRASP the cup with the hot liquids in order to PICKUP, and finally if it enables the agent to DRINK the hot liquids. The prototypical ceramic mug clearly satisfies these preconditions with its open concavity, its non-porous, insulating material, its flat bottom, handle, and light weight.

The styrofoam cup will be considered analogous to the ceramic mug if it too can be used for the stated purpose. To show that, the system would *map* the explanation generated for the ceramic mug<sup>2</sup> and attempt to *justify* that it is satisfied by this example. The styrofoam cup satisfies the explanation, although with slightly different structural characteristics. It differs structurally in that the styrofoam, not ceramic material, provides insulation; and the conical shape, rather than the handle, makes it graspable.

We are also applying Purpose-Directed Analogy to a more complex

<sup>1</sup>The domain theory consists of typical actions an agent can perform, and a structural and functional model of the artifact.

<sup>2</sup>The explanation can be viewed as a causal network of relations: it is a collection of relations R, related by a higher-order relation 'enable(ri,rj)'.

case study. Given the legal statute 'a vehicle is prohibited in a public park', the task is to learn the concept DISTURBING-VEHICLE, an object which enables driving but interferes with park use. (This work was initiated within the TAXMAN II project [225, 265].)

## FUTURE WORK

We plan to complete the implementation of the system in the near future. In addition, we plan to experiment with the system using case studies of increasing complexity. Further, several major theoretical issues still need to be addressed. One is the problem of automatically inferring the purpose of the artifact from some problem-solving context. Previous work has demonstrated a scenario for this in the context of heuristic search [166]. Secondly, we are exploring techniques to deal with the inexactness and incompleteness of the theory in commonsense domains, when learning concepts in these domains.

## ACKNOWLEDGMENTS

Thanks go to my advisor Tom Mitchell, to Rich Keller, Jack Mostow, Prasad Tadepalli and others who have commented on drafts of this paper. This research is supported by GTE Laboratories, under Contract No. GTE840917.

# **DEVELOPMENT OF A FRAMEWORK FOR CONTEXTUAL CONCEPT LEARNING**

**Richard M. Keller**

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

## **ABSTRACT**

This paper shows how several limitations of inductive concept learning methods can be overcome by giving the learning system explicit knowledge of the context in which learning occurs, including knowledge of 1) the kind of concept to be learned, 2) the performance system to be improved by learning and 3) an underlying plan linking the learned concept with expected performance improvement.

## **RESEARCH PROBLEM**

In recent years, several machine learning researchers [249, 320, 166] have suggested that the inductive concept learning systems developed in the 1970s exhibit some fundamental limitations that cannot be addressed from within the established paradigm. Inductive systems (e.g., Winston's ARCH system [396]) form concepts by generalizing from positive and negative instances based on syntactic similarities among the instances. A pre-defined generalization hierarchy specifies permissible generalizations of the observed syntactic features. Critics of the inductive systems characterize them as:

1. *Syntactic and Empirical*: Inductive systems perform generalization based on syntactic similarities which may be irrelevant to the concept being learned. Generalization is accomplished almost exclusively by data-driven pattern-matching techniques. The use of more powerful analytical, deductive learning techniques is not possible because these techniques generally require knowledge that is not represented in the inductive system's generalization hierarchy.
2. *Unresponsive*: Most inductive systems do not monitor their effect on the performance system being improved by learning. Without feedback on performance, the learning system cannot verify its

effectiveness and modify its behavior accordingly.

3. *Non-self-directed*: Inductive systems lack the important capability of deciding which concepts are worth learning. Consequently, these systems are unable to pose learning problems to themselves.
4. *Non-adaptable*: Modifying an existing inductive system to learn a different type of concept than originally intended requires manual intervention. New syntactic features and a new generalization hierarchy must be designed for each substantially different concept being learned.

The next generation of learning systems will need to overcome these deficiencies to solve increasingly difficult learning problems in the future. The aim of my research is 1) to design a new learning system framework that addresses the above difficulties and 2) to provide initial solutions to some problems arising within this new framework.

#### **APPROACH: CONCEPT LEARNING AS AN EMBEDDED PROCESS**

The key ingredient missing from inductive concept learning systems is *contextual meta-knowledge*: knowledge pertaining to the context in which learning transpires. Contextual meta-knowledge includes knowledge about 1) Why learning is taking place, 2) What concept is being learned and 3) How the learned concept will be used. With the inclusion of contextual meta-knowledge, a learning system has a more global perspective on the learning process. This perspective enables the system to take a more active role in reasoning about, monitoring and controlling its own behavior. In particular, addition of contextual meta-knowledge enables construction of a concept learning system that is:

1. *Analytical*: Takes advantage of more deductive, knowledge-based learning techniques,
2. *Responsive*: Monitors and responds to feedback from the performance system,
3. *Self-Directed*: Automatically formulates concept learning tasks, and
4. *Adaptable*: Easily adapts itself to learn different concepts.

My approach is to provide contextual meta-knowledge to the concept learning element by viewing concept learning in context -- as a sub-problem

embedded within the primary problem of improving the performance element. A **contextual concept learning system** is provided with a representation of the *performance system* to be improved and a specification of *performance objectives* for system behavior. The learning system initiates a *performance improvement planning process* which formulates a plan to achieve the stated objectives. In some cases, the *performance improvement plan* involves a sub-task of acquiring a specified *target concept*. Concept learning is then initiated from within this performance improvement planning context.

## RESULTS

My research is focused on recasting the LEX system [247], an existing inductive concept learning system, within the contextual framework described above. LEX is a learning system that improves the performance of an integral calculus problem solving system by learning the target concept 'useful-problem-solving-step'. LEX is designed specifically to learn this single target concept. To support the feasibility of a contextual framework, I have developed a hand trace of the performance improvement planning process by which a contextual learning system could actively choose to learn about 'useful-problem-solving-steps.' I have also implemented a concept learning system that learns this concept using very different techniques than the original LEX system. The performance improvement planning scenario and the implementation are described in the following paragraphs.

As input to the performance improvement planning process, the learning system would be provided with a declarative representation of the calculus problem solving system and the specific performance objective of improving efficiency while maintaining correctness. Then, using standard planning and constraint propagation techniques, the system would develop a plan to achieve the stated performance objective. For example, one possible plan involves improving efficiency by modifying the problem solver to prune the search space. The pruning criterion is developed as a by-product of the planning process: "execute only those problem solving steps that lead to a solution", i.e. only 'useful-problem-solving-steps.' This pruning criterion, although correct, is not very efficient to use because the problem solver cannot determine whether a step "leads to a solution" without first applying that step and then searching for a solution. As a subtask, therefore, this plan involves re-expressing the initial, inefficient target concept definition in a more efficient form. I call this subtask a *concept operationalization task* [165].

I have also implemented a concept learning system, named METALEX, that accomplishes concept operationalization. METALEX takes as input 1) the initial definition of the 'useful-problem-solving-step' target concept, 2) the calculus problem solving procedure, 3) a criterion for evaluating the efficiency/correctness performance objective and 4) knowledge about the performance improvement plan which involves the target concept. METALEX outputs a re-expression of the initial target concept definition that can be used, in accordance with the performance improvement plan, to achieve performance objectives. The learning algorithm works by empirically monitoring execution of the performance improvement plan and modifying the target concept definition in response to observed plan failures.

Specifically, METALEX starts with a version of the problem solver that has been modified according to the performance improvement plan. In particular, the modified solver uses the initial target concept definition to prune away all but the 'useful' problem solving steps. The modified problem solver is executed on a set of selected training problems, and a number of empirical, plan-monitoring measurements are gathered during execution, including overall execution time, pruning time, number of 'useful' and 'useless' problem solving steps executed, and various other measurements. If performance objectives are not satisfied by the initial target concept definition, the definition is modified based on data provided by the empirical plan-monitoring measurements. For example, the initial definition, although correct, is so inefficient to evaluate that it cancels out any performance improvement that results from pruning. The pruning time measurement reveals this inefficiency and the concept definition is modified to improve evaluation efficiency. Continuing with the cycle of execute-monitor-modify, METALEX eventually re-expresses the definition of 'useful-problem-solving-step' into a form that improves the problem solver's performance on problems in the training set and other similar problems.

Currently, LEX out performs METALEX when comparing extent of performance improvement achieved by each concept learning system. However, METALEX holds considerable promise for significant improvement with further development. Moreover, METALEX's novel architecture squarely addresses the deficiencies cited above for inductive systems.

## RELATED WORK

This research is an outgrowth of my work with Tom Mitchell on LEX2 [165, 249] and Explanation-Based Generalization [255]. Recent work by Kedar-Cabelli on Purpose-Directed Analogy [164] is based on a framework similar to the contextual framework described in this summary.

## ACKNOWLEDGMENTS

My thanks to Tom Mitchell and Smadar Kedar-Cabelli for their help in refining the formulation and presentation of the ideas expressed above. Thanks also to Jack Mostow for comments on an early draft of this paper. Research support has been provided by a Rutgers University Graduate Fellowship, NIH Grant No. RR-64309 and NSF Grant No. DCS83-51523.

# ON SAFELY IGNORING HYPOTHESES

Kevin T. Kelly

University of Pittsburgh, Department of History and Philosophy of Science  
Pittsburgh, PA

## ABSTRACT

Inductive intelligence is a matter of inductive generality and computational feasibility. Efficiency can be improved with no cost in generality by a device that carefully ignores only unsuitable or redundant hypotheses. An example of this approach in the inductive inference of clausal theories is described.

## INTRODUCTION AND MOTIVATION

There are two goals for designers of inductive inference devices. The first is to design machines that are inductively *general*-- i.e. that can converge to strong, true hypotheses in many different possible worlds. The second is to design machines that are *feasible*. The difficult task is to achieve both goals at once. Hence, there are some proposals for procedures that run on real computers but whose inductive generality is questionable and difficult to investigate [155, 396] and there are other proposals for "enumeration" or "generate-and-test" procedures that are easily seen to be general [131, 27, 116] but that are computationally hopeless.

Some research has been directed toward retaining the generality of enumeration methods while chipping away at their enumerations so as to increase their feasibility [152, 272, 330]. Careful examination reveals that a lingering difficulty apparent in each of these attempts is the repeated consideration of hypotheses whose inadequacy is implied by the inadequacy of hypotheses previously tested. This note describes a strategy for alleviating this difficulty for clausal hypothesis languages.

## THE INDUCTIVE PROBLEM

The problem to be solved assumes evidence to be given as finite, consistent sets of atoms or negated atoms. Hypotheses are assumed to be finite sets of function-free clauses. What is sought is a strongest suitable

hypothesis for any given evidence. A suitability relation is assumed to have the property that (A) any hypothesis refuted by the evidence is unsuitable and that any logical consequence of a suitable hypothesis is also suitable (B) any hypothesis entailing an unsuitable hypothesis is also unsuitable, and (C) A set of clauses is suitable only if each clause in the set is suitable. An aim for intelligent inductive inference, then, is to find the strongest suitable hypothesis for any evidence  $e$ , without considering any hypothesis weaker than an hypothesis whose suitability has been discovered or stronger than an hypothesis whose unsuitability has been discovered.

## CANONICAL STRINGS

Let  $U$  be any alphabet. Consider the structure  $\langle U^k, \leq \rangle$ . For any  $\sigma, \sigma'$  in  $U$ ,  $\sigma \leq \sigma'$  exactly if there is a substitution  $\eta$  such that  $\sigma = \sigma' \eta$ . Define  $\tilde{\sigma} \sim \sigma'$  exactly if  $\sigma \leq \sigma'$  and  $\sigma \geq \sigma'$ .  $\sim$  is the relation of string-isomorphism over  $U^k$ . Let  $\sigma$  be in  $U^k$ , and let  $\text{FIRSTOCC}(\sigma, i)$  be that element  $u$  of  $U$  such that exactly  $i-1$  elements of  $U$  occur in  $\sigma$  before the first occurrence of  $u$  in  $\sigma$ . Then define  $\text{CANON}(\sigma) = \sigma \eta_\sigma$ , such that the substitution  $\eta_\sigma = \{1/\text{FIRSTOCC}(\sigma, 1), \dots, n/\text{FIRSTOCC}(\sigma, n)\}$ . For example,  $\text{CANON}([x, y, z, x, x, y]) = [1, 2, 3, 1, 1, 2]$ . For any subset  $S$  of  $U^k$ ,  $\text{CANON}(S) = \{\text{CANON}(\sigma) : \sigma \in S\}$ .  $\text{CANON}(S)$  has the property that for any  $\sigma, \sigma' \in S$ ,  $[\sigma] \geq [\sigma']$  exactly if  $\text{CANON}(\sigma) \geq \text{CANON}(\sigma')$ . So the  $U^k$  structure is homomorphic to the  $\text{CANON}$  structure, which in turn is isomorphic to the quotient structure  $U^k / \sim$ . A string  $\tau$  is canonical if there is a string  $\sigma$  such that  $\tau = \text{CANON}(\sigma)$ . Equivalently,  $\sigma$  is canonical exactly if  $\sigma$  is a string of positive integers such that 1 occurs first, and for  $n > 1$ , if  $n$  occurs in  $\sigma$  then  $n-1$  occurs in  $\sigma$  before  $n$  does.  $\text{CANON}(n)$  is the set of all canonical strings of length  $n$ . And finally,  $\text{LEVEL}(k, n)$  is the set of all canonical strings of length  $k$  such that  $n$  is the greatest positive integer occurring in each. So the set  $\text{CANON}(k)$  of canonical strings is a compact way to represent the structure of substitution over a set of strings of length  $k$ . It is easy to verify that  $\text{CANON}(k)$  is a lattice, and in the Hasse diagram of this lattice, each  $\text{LEVEL}(k, n)$  is the  $n$ th level of the diagram.

## APPLICATION TO CLAUSES

Consider an arbitrary function-free, relational clause like

*Clause C:*  $-G(x,y)$  or  $-G(y,z)$  or  $G(x,z)$

which says that  $G$  is transitive. None of Shapiro's implemented model inference systems described in [330] is capable of discovering this simple clause, because they ignore the difficult task of searching the different patterns of variables that can arise in a clause of form  $[-G(*,*)$  or  $-G(*,*)$  or  $G(*,*)]$  pp.23-26. To concentrate on this combinatorially difficult aspect of clausal search,  $C$  can be uniquely coded as the pair  $\langle \text{PREDLIST}(C), \text{VARLIST}(C) \rangle$ , such that  $\text{PREDLIST}(C) = [-P,-P,P]$ , and  $\text{VARLIST}(C) = [x,y,y,z,x,z]$ . That is, the  $\text{VARLIST}(C)$  is exactly the sequence of all occurrences of variables in  $C$ , in order of occurrence, and  $\text{PREDLIST}(C)$  is the corresponding list of occurrences of predicates or their negations in  $C$ . A PREDLIST is  $n$ -ary if a list of  $n$  variables is required to fully specify it as a clause. Let  $P$  be an  $n$ -ary PREDLIST. Let  $\text{SPEC}(P)$  be the set of all ways of specifying  $P$  as a clause over variable alphabet  $\text{VAR}$ . So each element of  $\text{SPEC}(P)$  can be thought of as an element of  $\text{VAR}^n$ . Moreover, the structure  $\langle \text{SPEC}(P), \models \rangle$  (in which  $\models$  is semantic entailment) is isomorphic to  $\langle \text{VAR}^n, \geq \rangle$ . So we can preserve all the logical structure of  $\text{SPEC}(P)$  by representing the infinite search space  $\langle \text{SPEC}(P), \models \rangle$  by the finite structure  $\langle \text{CANON}(\text{VAR}^n), \geq \rangle$ .

## THE SEARCH

Each PREDLIST can be represented as a vector of positive integers such that each entry stands for the number of predicates or negated predicates occurring in the corresponding PREDLIST. Vector  $v$  precedes  $v'$  exactly if  $v' = \langle k_1+v_1, \dots, k_n+v_n \rangle$ . All of the PREDLISTS immediately preceded by  $v$  can be generated by successively adding 1 to each entry of  $v$ . Starting with the  $\mathbf{0}$  vector, we search the SPEC lattice of each vector  $v$ , which is represented simply as the lattice  $\text{CANON}(k)$ , such that  $k$  is the arity of  $v$ . From the completed searches of the parents of  $v$  in the precedence ordering we can easily compute a set of SEEDS-- canonical tuples that represent, along with  $v$ , the weakest clauses not entailed by any suitable clause in  $\text{SPEC}(v')$  for any  $v'$  preceding  $v$ .

The SEEDS for  $v$  mark off the upper portion of  $\text{SPEC}(v)$  that need not be searched because its vertices represent clauses entailed by previously confirmed clauses. So if  $\text{SPEC}(v)$  is searched "breadth first" from weaker to stronger clauses, canonical strings above the SEEDS in the corresponding canonical lattice can be ignored by beginning the search with the SEEDS rather than at the top of the lattice.

Moreover, any daughter of an unsuitable clause in  $\text{SPEC}(v)$  is also

unsuitable and hence can be ignored by not generating any canonical daughter of a canonical string corresponding to an unsuitable clause. But since a daughter of an unsuitable clause can also be a daughter of a suitable clause, an important question is how to generate the clauses at a level  $L$  of a SPEC lattice that are *not* daughters of a set of unsuitable clauses at a previous level  $L'$  without performing  $\leq$ -checks between all the vertices of the level and all the failed vertices at the prior level. This is accomplished by algorithm CDS (Complement-Descendant-Set). CDS represents the canonical lists at Level  $L$  as a connected tree of integers, for each canonical list shares some prefix with any other. CDS has the property that some branches of the tree of level  $L$  that are daughters of failures at level  $L'$  are not considered or generated. So typically, only a few vertices in the "middle" of a SPEC lattice need ever be generated or considered. After  $SPEC(v)$  is searched, those clauses that suitable but not entailed by any other suitable clauses in  $SPEC(v)$  are placed in a global set HYP.

Finally, whenever the bottom of a lattice  $SPEC(v)$  is suitable, none of the vectors  $v'$  preceded by  $v$  is searched. And when no vectors  $v$  remain to be searched, HYP is returned as the procedure's conjecture. The procedure is guaranteed to find the strongest suitable hypothesis for the given evidence, if there is one.

## OBSERVATIONS

I have implemented the SPEC search (with CDS as a subroutine) and the SEED generator in MACLISP. A remaining difficulty with this procedure is that when a clause is logically equivalent to a clause shorter than itself, it will be considered twice, in different SPEC lattices. It has proven difficult to design these redundancies out of the system in an elegant fashion. Another project is the extension of this procedure to clauses with function symbols.

## CONCLUSION

Since inductive intelligence is composed of feasibility *and* generality, one way to pursue it is to seek methods that ignore hypotheses without compromising inductive generality. The above procedure, based on the heuristic search of sets of hypotheses represented as trees of canonical strings, is a step in this direction.

# A MODEL OF ACQUIRING PROBLEM SOLVING EXPERTISE

Dennis Kibler and Rogers P. Hall

Irvine Computational Intelligence Project

Department of Information and Computer Science, University of California  
Irvine, California 92717

## ABSTRACT

This is a shortened version of a paper describing a computational model of acquiring problem solving expertise through a combination of instruction, analogical reasoning between similar problem types, and problem solving experience. In keeping with psychological studies of problem solving, expertise is defined as the possession of problem schemata which associate solution strategies with problem descriptions, supported by a specialized conceptual vocabulary for describing problems. Problem solving is described as a process of problem understanding which makes liberal use of analogical reasoning in an effort to "connect" novel problems to known solution strategies. Problem schemata and the conceptual vocabulary of which they are constructed are learned in a manner which is incremental, pragmatic, and strongly connected to existing knowledge sources.

## INTRODUCTION

As a goal for a computational model of acquiring problem solving skills, expertise in problem solving can be viewed as the possession of an abstract **conceptual vocabulary** tailored to the particular problem solving domain and an assortment of **problem schemata** reflecting problem categories. These schemata provide a mechanism for retrieving appropriate problem solving strategies when a problem from a particular category is encountered. Retrieval is based on conceptual cues which must be inferred from the problem statement. Acquisition of problem solving expertise in some domain then amounts to learning these problem schemata and the conceptual vocabulary out of which they are constructed.

We will present a computational model of learning problem solving skills in the domain of algebra story problems. Problem solving is cast as a process of problem understanding in which previously acquired knowledge is applied to achieve a unified representation of a novel problem. This aspect of problem solving will be described as a form of analogical

reasoning: new problems are "viewed as" variants of familiar problem categories with the aid of inferences provided by background knowledge sources. Learning in the context of such knowledge sources is primarily a process of modifying (extending, reorganizing or correcting) problem-specific knowledge for the purpose of widening problem solving skills.

## A PROBLEM SOLVING ARCHITECTURE

Our model of solving algebra story problems depends upon the cooperation of multiple knowledge sources operating in an asynchronous fashion through a globally accessible problem description. The initial problem description is a jumbled set of propositions with relatively little structure. From an abstract perspective, problem solving proceeds by using existing knowledge sources to manipulate this problem description until sufficient quantitative constraints are available to allow calculation of a solution. A solution path in this space of manipulated problem descriptions consists of increasingly coherent descriptions of the current problem generated by the actions of applicable knowledge sources.

Although the problem solver may utilize a large number of knowledge sources, we can divide these various knowledge sources into four functional types. These include: **augmenters**, **organizers**, **asserters**, and **decomposers**. The problem solver schedules activities of the various knowledge sources by a competitive process which attempts to expend the least effort while still progressing towards a goal state. Augmentation and organization are least costly, but do not generate any equations, and therefore may not necessarily lead closer to a problem solution. Instead their contributions must be focused towards enabling and confirming other knowledge sources. Assertion and decomposition, although more complex, are the only operations that generate equations. They must be applied to achieve a solution. Since enabling conditions for any of the four types of knowledge sources may be only partially matched by a current problem description, we must often treat a knowledge source proposing a modification to the problem description as a hypothesis which is subject to confirmation. This is precisely the role of analogical reasoning in this model of problem solving.

## LEARNING MECHANISMS

The goals of learning, as described earlier, are the acquisition of problem schemata and the specialized conceptual vocabulary of which they are constructed. Broadly speaking, the learner's conceptual vocabulary (i.e.,

frame-like descriptions of events) gradually changes as the result of problem solving experience and instruction. Direct instruction will be involved in the extension or correction of some knowledge sources, while learning when to apply background knowledge (i.e., acquisition of problem schemata) is the learner's responsibility. Hence our computational model of learning to solve algebra story problems involves a variety of forms of learning including learning by being told, learning by taking advice and learning from examples.

The learning mechanisms we propose satisfy a number of constraints. First, they are all incremental. Thus, problem solving expertise is acquired gradually through active experience with a succession of problems and instruction presented by a teacher. Second, learning is tolerant of errors. Correctness of acquired concepts or schemata is relative to problem solving experience rather than being defined in absolute. Concepts constantly change and evolve according to their problem solving utility. Third, newly acquired knowledge is connected to old knowledge by the recognition, elaboration and confirmation of analogies. Thus learning does not occur in isolation from existing knowledge of the task domain.

We briefly consider some examples of learning processes, each example based on simple problems drawn from Mayer's taxonomy [223]. These examples demonstrate direct interaction of instruction, inferences based on background knowledge, and problem solving experience. The first example shows how the learner refines initially naive concepts so that they are more appropriate for problem solving. The second example shows how analogy can drive the transformation of previously learned concepts and schemata into new problem solving knowledge.

## CONCLUSIONS

The full version of this paper illustrates how a conceptual vocabulary and problem schemata can be learned through a combination of instruction, analogical reasoning between similar problem types, and problem solving experience. Acquisition of problem solving skills is incremental, pragmatic, and strongly connected to existing knowledge sources. Problem solving is modelled as a process of problem understanding. Through inferences supplied by processes of augmentation, organization, assertion and decomposition, problem descriptions become increasingly coherent and constrained. Analogical reasoning allows tentative application of known solution methods, subject to verification by surrounding knowledge sources.

## **ACKNOWLEDGMENTS**

This work was supported by a gift from the Hughes AI Research Laboratory, a division of Hughes Aircraft.

# ANOTHER LEARNING PROBLEM: SYMBOLIC PROCESS PREDICTION

Heedong Ko

Artificial Intelligence Laboratory, Department of Computer Science  
University of Illinois at Urbana-Champaign  
Urbana, Illinois 61801

## INTRODUCTION

Inductive inference is a mode of reasoning that starts from particular facts and concludes a general hypothesis that explains the observed facts and hopefully predicts some unobserved facts correctly. What is given and assumed for the facts and what types of explanation and prediction are intended from the hypothesis warrant different solutions to the problem of how to derive the hypothesis from facts.

Facts are given as **independent** positive and negative instances of some generic object such as an arch [396] or a soybean disease [70]. The hypothesis subsumes all the positive and none of the negative instances (here, I consider subsumption as a type of explanation). We call such learning problem, **instance-to-class** problem.

Instead, we can show parts of the object to be learned and the description of the whole object is derived as an explanation for how the parts are related. We call such learning problem, **part-to-whole** problem. We can refine this problem further depending on the type of the object to be learned. When the object is a sequence of letters, this learning problem is traditionally called, **letter sequence extrapolation problem** [387]. That is, the letters are part of the sequence and implicitly related by **next-to** relations and the hypothesis, sequence generating rule, explains how the letters are related and predicts the future continuation of the sequence.

## SYMBOLIC PROCESS PREDICTION

**Symbolic process prediction** substantially generalizes letter sequence extrapolation problem by allowing elements of the sequence to be any symbolic description, called **snapshots**. Initially, Dietterich [96] investigated symbolic process prediction under Eleusis card game and developed SPARC/E program where each snapshot is described by a set of card attributes such as suit, rank, and more and the program discovered the card sequence generating rules. Michalski, Ko and Chen generalized

SPARC/E and developed SPARC/G [235] to deal with arbitrary set of attributes. SPARC/G has shown that many important problems can be formulated in the context of symbolic process prediction.

One of them is to draw **causal relations** between adjacent snapshots in a physical world. The spring is attached to the wall and a mass is attached to the other end. Each snapshot is described with a set of attributes such as direction of movement of the mass, the length of the spring (extended, rest or compressed) and more. SPARC/G hypothesized that the length of the spring determines the direction of the movement of the mass, a shallow causal analysis of the nature of motion. Deep causal account of the observed facts in terms of the restoring force and newton's second law can be obtained by proving the inductive hypothesis using standard deductive inference techniques but the focus of the research was not to conquer the domain, the physical world but to understand the generality of symbolic process prediction view.

## MODEL DRIVEN LEARNING

In symbolic process prediction problem, the rule space for inductive inference is too large for strict forward application of inductive inference rules on the fact database so a metalevel control [171] of this process is necessary and provided for by models. A model prescribes the syntactic form of the inductive hypothesis. Model does not manipulate the control structure of the inductive inference rules because model does not choose inductive inference rules but rather, model constrains how to view the fact database. Three models are identified for symbolic process prediction:

1. disjunctive model ignores the sequential information in the sequence and treats each snapshot as an instance of the object to be learned, instance-to-class problem. Therefore, symbolic process prediction subsumes instance-to-class problems.
2. decompositional model is used to discover if-then hypotheses where the objects mentioned in the if-part temporally precedes those of the then-part. The spring example was run under this model.
3. periodic model is used to discover recurring regularities in the sequence, loops. The hypotheses discovered from decompositional model are used as fact database for periodic model. Consider each hypothesis as a node of a graph and draw an edge from A to B iff the then-part of A unifies with if-part

of B. Now, a standard back-edge detection algorithm is used to detect the cycle in the graph. Once a cycle is detected, all the nodes in the path form the periodic rule.

There is a hierarchical relationship between models: disjunctive model is used by decompositional model and and decompositional model is used by the periodic model. The program should use such hierarchical relationship to avoid redundant effort. Unfortunately, the actual implementation is less general than ideal models so that only partial hierarchical relationships exist between decompositional and periodic model.

Decompositional model chooses an attribute of the if-part object and partition the sequence for each value of the attribute into a set of snapshots whose previous snapshot is covered by the if-part. For each partition, disjunctive model can be used for the description of the then-part by considering the snapshots for the then-part as examples of some class and then, standard instance-to-class algorithms such as Aq11 [230] can be used.

Since each snapshot can be associated with arbitrary number of attributes, it is combinatorially explosive to consider the attributes of the if-part exhaustively. Probability theory tells us that when  $P(A)=P(A|B)$ , A and B are independent. Then, the attribute for the if-part can be chosen heuristically using the fact that if the conditional probability distribution for the partition according to the chosen attribute does not change from the probability distribution of the original sequence, the attribute is an implausible choice.

Periodic model using just the graph search technique does not capture all the possible looping structures because decompositional model is not fully general. Since an edge can be drawn only when then-part of a rule matches with if-part of another rule and if-part has restricted syntax, namely only single attribute, from decompositional model, graph search will discover only certain attribute repeats its value. So, current implementation takes a phase of the periodic rule as an advice and segment the sequence accordingly and treat each segmented sequences as another full-fledged sequence and recursively invokes the entire program with each of them.

## FUTURE RESEARCH

Symbolic process prediction addresses a class of learning problems that discovers the sequence generating rules from a sequence of snapshots that are described by many attributes. There are other types of part-to-whole learning problems. For example, you can show a temporally adjacent scenes of the world before and after an action and learns the precondition

and effects (add and delete list) of STRIPS style operators. Note that this is different from LEX style of operator learning [256] where only the preconditions are learned. Implementation is underway.

Previously, learning structural descriptions from examples [396] are considered to be an instance-to-class problems where each examples are considered independent. But this is just like teaching a child what an arch is by showing example of it at one time and years later, show another example of it. Instead, examples are given in sequence in a short period of time. We should take advantage of such temporal information between descriptions. This constraint demolishes the combinatorial problem because most of object correspondences are given by temporal adjacency. Disjunctive model is in this spirit but ignores temporal adjacency and descriptions of snapshots are not structured objects. The remedy is on the way.

Dufay and Latombe [106] reports synthesizing a manipulator level program from traces of execution of robot motions in the context of mechanical assembly tasks. The trace is represented by a linear graph where the nodes are motions and the links are the state descriptions including geometric relationships of the parts of the assembly. This graph is translated into loops and conditional statements. Loop detection is similar to periodic model and conditional statement detection is to decompositional model. Automatic synthesis of robot programs for assembly task are being considered as an application domain for symbolic process prediction.

## ACKNOWLEDGMENTS

Author's view of considering different types of learning problem was originally inspired by part-to-whole learning problem posed by R. S. Michalski. The author also thanks Kaihu Chen for valuable discussions.

This work was supported in part by the National Science Foundation under grant NSF DCR 84-06801 and by the Office of Naval Research under grant no. N00014-82-K-0186.

# **LEARNING AT LRI ORSAY**

## **Yves Kodratoff**

UA 410 du CNRS et Laboratoire de Recherche en Informatique  
Université de Paris-Sud F-91405 ORSAY CEDEX France

### **ABSTRACT**

We shall give a hint of the 8 following sub-fields of Learning by which the "Inference and Learning" group of LRI is concerned.

1 - theory of the "best" generalization. 2 - use knowledge about the universe (generalization hierarchies, theorems, logical & idempotence). 3 - variable bindings. 4 - theorems. 5 - multiple hierarchies. 6 - change representation language. 7 - build new concept hierarchies. 8 - applications: Plant pathology, rocket control, archaeological axes, comparison with human learning, help to expert questionnaires design, vision.

### **THEORY OF THE "BEST" GENERALIZATION**

We are attempting to find a complete and comprehensive theory within the framework of first order logic (with the least possible involvement with the second one when a new function or predicate must be "invented") on the way of comparing two generalizations drawn from the same set of examples and counter-examples.

Very generally, one can describe our approach as following the classical one that uses already stored knowledge about the universe in which learning is taking place. We are trying to systematize this approach, and are therefore confronted to the problem of the combinatorial explosion of the possible deductions that can be made from a big knowledge base.

We shall not describe our formal results here but rather illustrate them by a partial description of the implementations that are done or under way at LRI.

### **USE KNOWLEDGE ABOUT THE UNIVERSE. VARIABLE BINDINGS. THEOREMS.**

We have tried to combine knowledge about the predicates that describe the examples, generalization hierarchies and theorems [229, 256] knowledge about logical connectors [143, 172, 382].

As a negative feature, our approach very much increases the combinatory complexity of the generalization. As a positive feature, it allows to find generalizations that have a better fit with the examples.

We shall now illustrate both improvement and difficulties on a trivial example.

### *Example*

*Imagine that we want to learn what is contained in the two following sentences illustrating a concept one is supposed to find.*

1 - *A is a square and B is red.* 2 - *C is a red square and D is red.*

*Suppose that the knowledge about the universe can be summarized by*

*T1: different names are given to different objects*

*T2: all objects have a color*

*T3: the "and" in the above sentences has the properties of the logical &.*

*The two above sentences can be given the form*

*E1:  $\text{SQUARE}(A) \ \& \ \text{RED}(B)$ . E2:  $\text{SQUARE}(C) \ \& \ \text{RED}(C) \ \& \ \text{RED}(D)$*

*Without using knowledge about the universe, one would find generalizations that forget some information about the examples like*

*G1:  $\text{SQUARE}(x) \ \& \ \text{RED}(y)$*

*Understanding that two different variables can always receive the same constant unless specifically forbidden, G1 says that there are at most (at most means that the two objects may merge into only one) two objects, one is a square, the other one is red.*

*Using T2, we could improve somewhat G1 by adding that x must have some color u, obtaining*

*G2:  $\text{SQUARE}(x) \ \& \ \text{COLOR}(u,x) \ \& \ \text{COLOR}(\text{RED},y)$*

*where RED is now used as a constant rather than a predicate. Otherwise stated, one has "climbed the generalization tree" as Michalski's APC would say [22, 229]. G2 says that there are at most two objects, one being a coloured square, the other one being red.*

*Now directly applying knowledge about the universe to E1 and E2, one can improve this result.*

*First way.*

*One can use T2 to transform E1 and E2 into equivalent forms E1'*

and  $E2'$ .

$E1': \text{SQUARE}(A) \ \& \ \text{COLOR}(u,A) \ \& \ \text{COLOR}(\text{RED},B)$

$E2': \text{SQUARE}(C) \ \& \ \text{COLOUR}(\text{RED},C) \ \& \ \text{COLOR}(\text{RED},D)$

A possible generalization of  $E1'$  and  $E2'$  is

$G': \text{SQUARE}(x) \ \& \ \text{COLOR}(w,x) \ \& \ \text{COLOR}(\text{RED},y) \ \& \ \text{DIFFERENT}(x,y)$

where  $T1$  has been used in order to mark that  $x$  and  $y$  have different instances in all examples.

Introducing such a new predicate has been called "detecting descriptor interdependence" in APC.

$G'$  states that there are exactly two different objects, one is a colored square, the other is red, while  $G2$  above was forgetting the information about the difference of the instances of  $x$  and  $y$ . Therefore  $G'$  is better than  $G2$ .

#### Second way

One can use  $T3$ , particularly the fact that  $A$  is equivalent to  $A \ \& \ A$  for any predicate  $A$ , in order to transform  $E1$  into  $E1''$ , the examples become

$E1'': \text{SQUARE}(A) \ \& \ \text{RED}(B) \ \& \ \text{RED}(B)$

$E2: \text{SQUARE}(C) \ \& \ \text{RED}(C) \ \& \ \text{RED}(D)$

and generalize to

$G'': \text{SQUARE}(x) \ \& \ \text{RED}(y) \ \& \ \text{RED}(z) \ \& \ \text{DIFFERENT}(x,z)$

$G''$  says that there are at most three objects two of them being red and one being square.

Using  $T2$  in  $G''$ , one obtains

$G''': \text{SQUARE}(x) \ \& \ \text{COLOR}(w,x) \ \& \ \text{RED}(y) \ \& \ \text{RED}(z) \ \& \ \text{DIFFERENT}(x,z)$

which keeps all information of  $G'$  plus the fact that there are two red objects. Therefore  $G'''$  is better than  $G'$ .

More complicated examples would unfortunately show that there are cases were the generalizations obtained as the above  $G'$  and  $G'''$  are not always comparable.

## MULTIPLE HIERARCHIES

We have recognized that multiple (or "tangled") hierarchies are necessary to express complete knowledge. The following example will show how several hierarchies may allow to refine the heuristics that lead rule application. It shows also how "graded concepts" can be described and used.

*Suppose that we want to find the best possible rule that can be induced from the two following examples.*

$$\text{E1: NEED(FRANCE, MAGNETOSCOPES) \&} \\ \text{PRODUCE(JAPAN, MAGNETOSCOPES)} \Rightarrow \\ \text{MAYBUY(FRANCE, MAGNETOSCOPES, JAPAN)}$$

*which is short for saying that if France needs magnetoscopes and that Japan produces them, then France may buy them from Japan.*

$$\text{E2: NEED(BELGIUM, COMPUTERS) \& PRODUCE(USA, COMPUTERS)} \Rightarrow \\ \text{MAYBUY(BELGIUM, COMPUTERS, USA)}$$

*The usual way of generalizing these two rules is to remark that France, Belgium, USA and Japan are countries, that magnetoscopes and computers are goods and to write the generalization*

$$R: NEED(x,u) \& PRODUCE(y,u) \Rightarrow MAYBUY(x,u,y)$$

*Rule R cannot be applied blindly since x, u and y cannot take any value (for instance, u cannot be a country).*

*We have found here a problem comparable to the one solved by LEX, where rule R is given (it is, for instance, "APPLY rule OP<sub>2</sub>") and one has to find the heuristics that lead the application of R.*

*Suppose that one disposes of the following taxonomies, relative to goods.*

GOODS

electronic

....

computers	magnetoscopes	....
-----------	---------------	------

*relative to commercial treaties,*

## COMMERCIAL TREATIES

FREE ENTERPRISE

COMMUNIST

.... EEC

COMECON ....

France Belgium RFA....

USSR ....

*and relative to industrial development in electronic industries.*

## ELECTRONIC DVPT

FIRST RATE

SECOND RATE

....

*The two examples E1 and E2 are not enough to decide to which taxonomy "x" belongs.*

*Suppose that one has a third example E3 in which RFA buys also some electronic goods, one could then infer a condition for applying rule R.*

*C: x is an EEC country, and y is a first rate electronic industry country, and u is an electronic good.*

*The complete learned rule would be: IF C THEN APPLY R.*

**In this rule, RFA is a graded concept that has been used here as a first rate electronic industry country, but could also, in other contexts, be an EEC country.**

A given constant may belong to several hierarchies and one has to choose the good one in order to find the correct generalization formula. An implementation of the above hinted at algorithm is under way at LR1.

## BUILD NEW CONCEPT HIERARCHIES [28, 73]

There are many ways of generalizing a set of examples and there are many ways to cluster examples in order to obtain intermediate concepts.

We are currently developing an algorithm that clusters examples by using a distance that evaluates the future result of their generalization.

Examples that will give very general expressions are said to be far from each other, examples that generalize less are said to be closer.

A kind of symbolic distance is thus defined, this distance depends

very much on the relative importance (or weight) one gives to each descriptor.

To each ordering of the importance of the descriptors is associated a different concept hierarchy.

Building such intermediate concept hierarchies is clearly important in Robotics where subtasks must concur with the main task.

We are currently studying how to apply this non-numerical approach to uncertain or blurred data.

# **COPER: A METHODOLOGY FOR LEARNING INVARIANT FUNCTIONAL DESCRIPTIONS**

## **Mieczyslaw M. Kokar**

Northeastern University

Department of Industrial Engineering and Information Systems  
360 Huntington Avenue, Boston, MA 02115

### **ABSTRACT**

Functional descriptions constitute a significant class of goals for machine learning systems. Physical laws are one example of this kind of description. In this paper the COPER methodology is described, which allows discovery of functional descriptions from incomplete observational data. COPER eliminates irrelevant arguments, generates additional relevant argument descriptors (if some are missing), and generates a functional formula. The important feature of this methodology is that it allows testing of relevance for some of the attribute descriptors without varying their values throughout the training events. To do this, it utilizes the property of invariance of meaningful functional descriptions. One of the examples of COPER's rediscoveries is Bernoulli's law.

### **FUNCTIONAL DESCRIPTIONS**

A learning system is usually presented with a set of events, each of which is described as a sequence of attribute descriptors characterizing a given event. When dealing with functional descriptions one of the attributes is treated as a value attribute, while the others as argument attributes. A functional description gives a rule for ascribing value attributes to argument attributes. Such a rule constitutes a generalization of the given set of events. After learning this rule the system does not need to keep a table of events.

This kind of descriptions plays a very significant role in many applications. In problem solving an algorithm is a function whose argument is the starting point in the problem space, while the goal state is the value of the function. The function is described as a sequence of rules from the rule space. The rule for classifying poker 'hands' into concepts (flush, poker) and ordering them can serve as another example.

Physical laws represent another example of functional descriptions. For instance, Bernoulli's law describes the flow of a fluid through a pipe:

$$p_1 + \rho g h_1 + \rho V_1^2/2 = p_2 + \rho g h_2 + \rho V_2^2/2$$

where  $p_1$ ,  $p_2$  are pressures,  $h_1$ ,  $h_2$  - heights,  $V_1$ ,  $V_2$  - flow velocities,  $\rho$  - density,  $g$  - acceleration of gravity. A computer system, in order to generate a database for the process described by Bernoulli's law could collect the information about this process in a table (one entry is a sequence of values of the attributes  $p_1$ ,  $p_2$ , ...,  $\rho$ ,  $g$ ) and keep in memory. But this is not an efficient method. It is much better to keep just the rule of assigning values to the arguments and use it whenever needed.

## INVARIANCE

A functional formula can be generated only if the condition of functionality of the presented events is fulfilled (i.e., when all the value descriptors are constant for a subset of events for which the arguments are the same). It is quite easy to test this condition. There are two possible outcomes, positive and negative. In both cases it is not clear what conclusion should be drawn. If the condition of functionality is fulfilled then it could be the consequence of a relevant (not presented to the system) variable being constant. If the condition is not fulfilled, then what is the preferable direction of search for an additional descriptor. These difficulties can be overcome by utilizing some properties of functional descriptions.

Meaningful functions are functions which are expressed solely in terms of the descriptors' language (in other words, in terms of the operations generating the descriptors language) [218]. Such functions can be shown to be invariant [174]. It means that if the arguments of the function are changed according to the rules of generation of the description space then the values also follow the same rules. Physical laws, for instance, are invariant under transformations called 'dimensional transformations' [64, 175, 218].

It is possible to show that if a function is invariant, and if we know all of its arguments (complete relevance) and its value for one point (event), then it is possible to predict its value for a class of events (an orbit) [64, 175, 218]. If the predicted values do not agree with the observed ones then it means that the condition of complete relevance (Michalski [233]) is not fulfilled (some arguments are missing). The important feature of invariant functional descriptions is that to move from one point of an orbit to another not all arguments of the function need to be changed. This gives the possibility of testing relevance of argument

descriptors without varying their values. For instance, in the case of Bernoulli's law, we can test relevance of the argument g, acceleration of gravity, without varying its value. The methodology for learning invariant functional descriptions is called COPER, a detailed description of this methodology can be found in [174].

## COPER

COPER is a system for discovering invariant functional descriptions. The existing implementations of COPER are oriented toward discovery of physical laws. Research is underway for extending COPER to other nonphysical, nonnumerical domains.

The input information for COPER is of two kinds: syntactical - describing the syntax of argument descriptors (this is information about the system of units used), and experimental - results of measurements of some of the arguments of a particular physical process.

The goal is to decide which of the argument descriptors are irrelevant (those are eliminated from considerations), which of the arguments are missing (these should be added). The second part of the goal is to generate a formula describing the given events.

As an example, take the process described by Bernoulli's law. Assume, the information COPER is provided with consists of the value descriptor p2, the argument descriptors: p1, h1, h2, V1, V2, G, S (where G and S are some irrelevant descriptors, we assume here that they were held constant), rules of generating new descriptors (these are operations of multiplication and exponentiation to a real number), a set of training events (these is a measurement table, one entry is a sequence of argument descriptors of one event and a respective value descriptor (p2) ). Note, that at that point COPER is not given any information about two missing arguments, acceleration of gravity g, and density ro. The training events can be generated out of the formula for Bernoulli's law, or can be obtained from observation (in the simulations both g and ro were held constant).

The result is such that COPER discovers that for this data, the condition of complete relevance is not fulfilled, in other words, some argument descriptors are missing. COPER starts searching for additional descriptors. In the current implementation the information about an additional parameter is given by an expert. Suppose the first argument given to COPER is acceleration of gravity g. COPER accepts it as a plausible one, but requests more. When presented with the second missing argument, density ro, COPER decides that the condition of complete relevance is fulfilled, and starts searching for a function. The resulting

formula is identical with formula representing Bernoulli's law.

The goals of COPER are similar to those of BACON [37] and ABACUS [112]. The methodological approach is different. The main differences are that COPER makes use of some syntactic knowledge of descriptors space; in case of physical laws these are units of measurement and operations of multiplication and exponentiation, and that it utilizes the concept of invariance of functional descriptions.

The use of invariance for the purpose of discovery gives a significant power to the system: it can reason about those argument descriptors whose values were not changed throughout the experiments, it can decide whether the condition of complete relevance is fulfilled (whether all relevant arguments are included), and it can generate the form of the missing argument descriptors (i.e., how the descriptors are expressed in terms of the system of measurement units). These features of COPER distinguish it from other systems for discovering functional descriptors. Neither BACON nor ABACUS can perform this kind of reasoning.

## FUTURE RESEARCH

1. Research is underway to extend COPER towards nonphysical nonnumerical functional descriptors. It should be able to discover a complete set of concepts in the game of poker, etc.
2. The concept of invariance is a base for a theory of similarity which is utilized in COPER. The concept of similarity is related to the concept of analogy. The next step of development is to extend the reasoning engine of COPER to incorporate reasoning by analogy.
3. Attempts should be undertaken to apply this system for discovering unknown laws. The promising fields are engineering sciences, e.g., software engineering, robotics, and nonengineering sciences, as social sciences, where the quantitative descriptors have not been as widely established.

# **USING EXPERIENCE AS A GUIDE FOR PROBLEM SOLVING**

**Janet L. Kolodner and Robert L. Simpson**

School of Information and Computer Science  
Georgia Institute of Technology  
Atlanta, Georgia 30332

## **ABSTRACT**

Much of the problem solving people do involves consideration of previous similar situations. Such *natural problem solving* integrates learning and analogy with problem solving itself. The transfer of knowledge from a previous case to a current one is guided by demands made by the problem solving process. Access to previous experiences keeps the problem solver from avoiding past mistakes and helps the problem solver specialize plans to particular types of situations. Both success and failure contribute to learning.

## **USING EXPERIENCE IN PROBLEM SOLVING**

We call the kind of problem solving that makes use of experience *natural problem solving*. Natural problem solving integrates learning and analogy with problem solving itself. Learning occurs as a natural consequence of problem solving. If a novel procedure is derived in the course of solving a complex problem and all goes well in its execution, then a new procedure is learned for dealing with this new class of problems. If problems are encountered applying known knowledge to a new case, the results of analyzing and fixing the problems result in refinement or modification of problem solving knowledge. Because general knowledge is not always adequate to deal with novel situations, analogy to previous similar novel situations is necessary to deal with their complexities. Thus, remembering a previous case to use in later problem solving is one of the necessary learning processes. Because descriptions of problems may be incomplete, problem understanding is a necessary prerequisite to coming up with a solution and is part of the natural problem solving framework. Feedback and analysis of feedback through follow-up procedures are also part of natural problem solving. Without evaluation processes based on feedback, learning could not happen and analogy to previous experience would be unreliable.

Within that framework, our approach has two components [176, 178]: *experience-driven learning* and *case-based problem solving*. In experience-driven learning, *experience contributes to the refinement and modification of reasoning processes and knowledge*. Experience drives the learning process. As problem solving is happening, the new experience is added to memory. Similarities between cases lead memory to make generalizations to use in later analysis. Successful experiences reinforce already-known rules or previous hypotheses, while failures trigger analysis of the reasoning and knowledge used originally, and modification of faulty rules and knowledge. Experience thus plays a major role in enhancing problem solving knowledge, in the process turning novices into relatively more expert reasoners.

*Case-based reasoning* uses previous experiences to suggest means of solving new problems. Thus, *individual experiences act as exemplars upon which to base later decisions*. Recall of a previous experience can aid in understanding the intricacies and focus of a new problem, generating a plan for resolution of the problem, and in case of failure, in explaining and remedying the failure and re-evaluating the case, resulting in avoidance of previous mistakes and shortcuts in problem solving.

The following example illustrates our framework. A failed mediation attempt triggers a need to explain the failure. A later episode, in a different domain, but with the same goal structure, causes reminding of the first episode, and through case-based reasoning, a prediction and advice about a proposed solution are given.

Two sisters want the same orange. Their mother proposes that each sister take half of it. One of the sisters states that she wants the whole peel for baking. The mother suggests that they divide it agreeably: one taking the fruit to eat, the other taking the peel for baking.

While the mother thought that both sisters had the same goal, she was mistaken. Though their sub-goals were in conflict, their goals were not. Stepping back and considering the real goals rather than the manifest ones resulted in a goal concordance. The following shows how this analysis is transferred in understanding and making a prediction about another situation. Here, we imagine the mother reading the following story in the paper:

Egypt and Israel both want possession of the Sinai. The US suggests they cut it down the middle. Both Egypt and Israel complain.

Analogy to the orange dispute allows her to conclude that possession of the Sinai is merely a subgoal, that the real goals of the two countries should be considered, and that a mutually-agreeable split based on those goals be sought.

## THE MEDIATOR

Our MEDIATOR project [178, 179, 344] resolves common sense disputes based on experience solving previous similar problems. Common sense disputes are the kinds people run into from day to day, e.g., children quarrelling over possession of objects, colleagues needing the same resource at the same time, and disputes described in the newspapers. The MEDIATOR program, developed by Bob Simpson, begins with a semantic memory detailing the kinds of disputes it might encounter (e.g., physical, economic, and political) and a set of common mediation plans (e.g., one cuts the other chooses, split the difference, divide by parts). As it resolves disputes, it builds up an episodic memory organized by the concepts in its semantic memory. During processing, it first attempts recourse to previous experience to resolve a problem, and if no applicable experience is available, it uses default means to resolve the problem. It learns based on feedback about the decisions it has made. If feedback is positive, it reinforces its belief that a particular type of plan is appropriate to a particular problem by storing the case and the plan used to resolve it. When it encounters later problems with features similar to one it has stored in memory, it will be reminded of that case and check to see if the plan used there was appropriate to its new problem. A positive experience may thus provide a shortcut in later problem solving. If feedback is negative, the MEDIATOR tracks down its error, fixes the knowledge that was responsible, and attempts resolution of the problem a second time based on the new knowledge learned during feedback and the corrected knowledge that caused the previous error. When it finally resolves the problem satisfactorily, and stores the entire case in memory, later reminding of that case will (1) allow the problem solver to resolve a later similar problem without making the same mistakes a second time or (2) help the problem solver to figure out what went wrong when a similar failure occurs in the future.

There are several novel aspects to the MEDIATOR project. First, its model of problem solving includes not only the planning part of problem solving, but also problem understanding and failure resolution based on feedback. Case-based reasoning facilitates reasoning during all of these phases of problem solving.

Second, the analogical transfer process is "demand driven", where demand is provided by the task the problem solver is carrying out. When the problem solver is trying to classify a problem, it is the problem classification of the previous case that it investigates for applicability to the new problem. When it is attempting to derive a skeletal plan, it is the abstract plan from the previous case that it checks for applicability.

Third, the MEDIATOR has a well-articulated long term memory for experience. Problem solving experiences presented to the MEDIATOR are indexed in memory by those features which differentiate them from other experiences represented in similar ways. The memory organization is based on MOPs [177, 319].

A fourth novel feature of the MEDIATOR is in its use of the same problem solving model to both solve domain problems (resolving disputes) and to track down and fix failures in reasoning. It is able to do this because it treats both types of problems as first, classification problems, and then, plan instantiation problems. In solving domain problems, it thus seeks to classify disputes it encounters according to whether they are physical, economic, or political disputes, each of which "knows" which types of plans are commonly useful to its resolution. Thus, classification allows pointers to potentially applicable skeletal plans, which are then refined for the particular problem.

During failure resolution, the MEDIATOR treats the failure it has encountered as its new problem. During the understanding phase of failure resolution (explaining the failure), it attempts to classify the error (as, e.g., a classification error, an elaboration error, a particular kind of elaboration error, a plan refinement error). Each of those error classifications has remediation plans associated with it to fix the faulty knowledge or faulty reasoning rule. It fixes its errors by instantiating and refining a plan appropriate to the kind of error it encountered (e.g., one can fix elaboration errors by using an alternate inference rule or by asking the value of a feature from the user). In the same way previous experiences can provide shortcuts in problem solving, previous failures can provide shortcuts in error recovery (e.g., the orange dispute above).

## OTHER PROJECTS

While the MEDIATOR explores the framework for integrating learning, problem solving, and analogy, there are details that it does not address. One of those is determining the level of abstraction at which an analogical transfer should be made. We are addressing that problem in the domain of trouble shooting (and fixing) breakdowns in household appliances.

Another topic not addressed is control of the simultaneous processes of problem solving and memory traversal. In our newest project, we are attempting to develop an architecture for problem solvers which use and learn from experience. We are concentrating on the interactions (via a blackboard) between three processes: the memory traversal process, the problem solver, and the interrupter. Our emphasis right now is in determining what the interrupter needs to know in order to decide that it is appropriate to interrupt the problem solver and present it with a case (found by the memory traverser).

## **ACKNOWLEDGMENTS**

This research has been supported in part by NSF Grant No. IST-8317711, in part by the Air Force Institute of Technology, and in part by ARO Grant No. DAAG29-85-K-0023. Robert Simpson's current address is ARPA/IPTO, 1400 Wilson Blvd., Arlington, VA 22209.

# **HEURISTICS AS INVARIANTS AND ITS APPLICATION TO LEARNING**

**Richard E. Korf**

Computer Science Department, University of California  
Los Angeles, Ca. 90024

## **ABSTRACT**

We present a characterization of heuristic static evaluation functions which unifies their treatment in single-agent problems and two-person games. The central thesis is that a useful heuristic function is one which is invariant along a solution path. This local characterization of heuristics can be used to predict the effectiveness of given heuristics and to automatically learn useful heuristic functions for problems.

## **INTRODUCTION**

Heuristic search is generally used in one of two types of applications: single-person problems such as the Eight Puzzle, and two-person games such as chess. Surprisingly, the treatment in the literature of heuristic search in these two different domains has little in common. In single-agent searches, a heuristic evaluation function is viewed as an estimate of the cost of the remainder of the solution path. In two-person games, however, a heuristic function is vaguely characterized as a measure of the "strength" of a board position for one player versus the other. Since these characterizations of heuristic functions are based on global information in the problem space, it is difficult to choose between candidate functions or to learn new ones automatically.

## **A UNIFIED THEORY OF HEURISTIC EVALUATION FUNCTIONS**

We propose a new theory of heuristic evaluation functions which unifies their treatment in single-person problems and two-person games. The theory characterizes successful heuristics in terms of local information which makes it an effective basis for evaluating given heuristics and for learning new heuristic functions. In this brief exposition we shall simply present the theory and indicate one application.

Simply stated, we claim that an ideal heuristic evaluation function has two properties: 1) when applied to a goal state, it returns the *outcome* of the search; and 2) the value of the function is invariant along an *optimal solution path*. The quality of a real evaluation function depends upon how closely it approximates this ideal. The *outcome* of a search is the criterion against which success is measured. For example, in a single-person search where the task is to find a lowest cost path to a goal state, the outcome would be the actual cost of the solution path found. In a two-person game, the outcome is either win, lose, or draw for a particular player. By *optimal solution path* we mean the actual sequence of moves made in going from the initial state to a goal state assuming perfect problem solving or play. In a single-person problem the term takes on its familiar meaning as a lowest cost path from initial to goal state, whereas for a two-person game, the optimal solution path is the sequence of moves that would be made by two players playing perfectly. Taken together, these two properties ensure a function which is a perfect predictor of the outcome of pursuing any given node in the problem space.

We claim that any successful evaluation function will satisfy these properties to some extent. For example, the evaluation function for the A\* algorithm [139] is  $f(n)=g(n)+h(n)$  where  $g(n)$  is the cost of the best path from the initial state to the node  $n$  and  $h(n)$  is an estimate of the cost of the best path from node  $n$  to a goal state. Typically the  $h$  term is called the heuristic in this function, but for our purposes we will refer to the entire function  $f$  as the heuristic evaluation function. When this function is applied to a goal node, the  $h$  term is zero, the  $g$  term represents the cost of reaching the goal from the initial state, and hence  $f$  returns the cost of the path or the outcome of the search. If  $h$  is a perfect estimator, then as we move along an optimal path to a goal state, each move increases  $g$  by the cost of the move and decreases  $h$  by the same value. Thus, the value of  $f$  remains invariant along an optimal path. If  $h$  is not a perfect estimator,  $f$  will vary somewhat depending upon the amount of error in  $h$ . Thus, a good evaluation function for an algorithm such as A\* will determine the outcome of the search and is relatively invariant over single moves.

Now consider a two-person game using minimax search and a static evaluation function. The static evaluation reflects the strength of a given board position. When applied to a state where the game is over, the function determines the outcome of the game, or which player won. This is often added as a special case to an evaluation function, typically returning positive and negative infinity for winning positions for MAX and MIN, respectively. When applied to a non-goal node, the function is supposed to return a value which predicts what the ultimate outcome of the game will

be. To the extent that the evaluation is an accurate predictor, its value should not change as the anticipated moves are made. Thus, a good evaluation function should be invariant over the actual sequence of moves made in the game. Therefore, in both examples we see that a good evaluation function should have the properties of determining outcome and invariance over single moves.

## PREDICTING HEURISTIC PERFORMANCE

The main theoretical contribution of this work is to unify the treatment of heuristic evaluation functions in single-person problems and two-person games. Its practical applications include predicting the performance of particular heuristics for particular problems, and learning heuristic functions. As examples of the former we will consider Manhattan distance for the Fifteen Puzzle and Rubik's Cube, and material count for checkers and Othello.

The Manhattan distance heuristic for the Fifteen Puzzle is computed by measuring the distance along the two-dimensional grid of each tile from its current position to its goal position, and summing these values for each tile. Manhattan distance is a very effective heuristic function for solving the Fifteen Puzzle [182]. A completely analogous heuristic can be defined in three dimensions for Rubik's Cube: for each individual movable piece of the cube, count the number of twists required to bring it to its goal position and orientation, and sum the values for each component. Even though Rubik's Cube is similar to the Fifteen Puzzle and the two heuristics are virtually identical, three dimensional Manhattan distance is effectively worthless as a heuristic function for Rubik's Cube [183].

As another anomalous example, consider the games of checkers and Othello with material count as an evaluation function. Othello is a game played on an eight by eight square grid with pieces which are white on one side and black on the other. Each player alternately places pieces with his color showing on empty squares. Whenever a player places his pieces at both ends of a line of his opponent's pieces, the opponent's pieces are flipped over and become the property of the original player. The winner is the player whose color shows on the majority of the pieces at the end of the game. Material count is an evaluation function which sums the number of pieces belonging to one player and subtracts the total material of the other player. It turns out that material count is a fairly successful evaluation function for checkers but relatively ineffective for Othello, even though both games have similar goals.

Our theory explains both of these anomalies. The explanation is that

for both evaluation functions the invariance property is to a large extent satisfied for one of the tasks but not for the other. In the case of Manhattan distance for the Fifteen Puzzle, a single move changes the Manhattan distance by a single unit whereas for Rubik's Cube a single twist can change the Manhattan distance by as much as eight units (eight pieces move at once). Similarly, the material count in checkers rarely changes by more than a single piece during one move, but in Othello it can change a large number of pieces. Our theory predicts that evaluation functions which are relatively invariant over single moves will be more effective, as is the case in these examples.

## LEARNING EVALUATION FUNCTIONS

In addition to making qualitative predictions about the performance of given evaluation functions for given problems, our theory can be used as the basis of a method for learning heuristic functions. The basic idea is that since the characterization of a successful evaluation function is in terms of invariance over single moves, candidate evaluation functions can be optimized based on local information in a problem space. In particular, one can search for a function which is invariant over moves along a solution path. This technique was implicitly used by Samuel's [312] pioneering experiments on learning checkers evaluation functions, and by Rendell's [292] more recent work on learning heuristics for the Fifteen Puzzle.

Christensen (this volume) presents some experiments in applying this idea to learning the coefficients of a linear static evaluation function for chess based on material. This is accomplished by using linear regression to modify the coefficients based on the difference between the static evaluation of a state and the value returned by a mini-max look-ahead search.

## CONCLUSIONS

We have presented a theory which unifies the treatment of heuristic evaluation functions in single-person problems and two-person games. The theory characterizes a successful heuristic in terms of invariance over single moves. This local characterization is useful for making qualitative predictions about the performance of given heuristics, and for the automatic learning of heuristic functions.

## **ACKNOWLEDGMENTS**

I would like to thank Jens Christensen for discussions which contributed to this work. This research is supported by the National Science Foundation under grant IST-85-15302.

# **COMPONENTS OF LEARNING IN A REACTIVE ENVIRONMENT**

**Pat Langley, Dennis Kibler, and Richard Granger**

Irvine Computational Intelligence Project  
Department of Information and Computer Science  
University of California, Irvine, California 92717

## **INTRODUCTION**

The goal of the C-MU/UCI World Modelers Project [Carbonell (this volume)] is to develop an integrated model of learning in a complex, reactive environment -- in particular, a simulated physical world that contains three-dimensional objects. Within this framework, we are designing cognitive architectures within which to cast our theories of learning, and in this paper we outline one such architecture. The paper is organized around four types of learning that we believe are central to dealing with reactive environments -- the formation of object concepts, the acquisition of procedures, the construction of cognitive maps, and the discovery of problem solving heuristics. In each case, we discuss issues of representation and performance in addition to problems of learning.

## **THE FORMATION OF OBJECT CONCEPTS**

Much of our knowledge takes the form of concepts for describing objects that exist in the world, such as "chair" or "person". Previous research in Machine Learning has represented concepts in terms of necessary and sufficient conditions, and relied on tutors to provide positive and negative instances of the concept being learned. We propose instead to represent object concepts as prototypes based largely on Binford's [23] generalized cylinder notation, and to model concept formation in the absence of a tutor. Example objects are divided into positive and negative instances based on their ability to satisfy the agent's goals, and the resulting concept descriptions are indexed by these goals as well.

Each concept is stored in terms of the basic shapes involved, together with numeric parameters such as the length, radius, and orientation of each component object. Each parameter has an average (default) value and an associated variance (inversely related to its criteriality). An incremental "averaging" process revises these descriptions as new instances are encountered. Recognition involves a partial matching process based on the

criteriality of each numeric parameter in the description. Though this process is expensive, indexing by goals allows one to selectively apply the matcher to promising concepts. Easterlin (this volume) discusses the role of goals in concept formation at greater length.

## THE ACQUISITION OF PROCEDURES

In addition to object concepts, an agent must have some procedures for achieving its goals. We plan to represent procedures as sequences of abstract state descriptions, connected by one or more operators. E.g., the procedure for throwing would contain a number of "snapshots" describing successive positions of the agent's arm and the object being thrown. Some operators can be decomposed into procedures themselves, giving a hierarchical representation. Procedural concepts can be used in two modes -- for recognizing instances of a procedure in the behavior of other agents, and for applying a procedure to achieve one's own goals. Recognition relies upon the predicted state descriptions, while application also requires knowledge of the operators. Both processes assume that operators are applied in a "repeat--until" fashion until the next predicted state has been achieved.

Procedures can be applied in two modes. In *closed loop* mode, the resulting state is compared to the predicted state description each time an operator is applied, to see if progress has been made. In *open loop* mode, one operator is applied a specified length of time, then the next is applied, and so forth; the resulting states are compared to the predicted states only after much of the procedure has been carried out. Open loop behavior can be carried out more quickly than closed loop behavior, since continual matches against the predicted state are not required. However, open loop behavior requires accurate estimates of the length of time each operator in the procedure should be applied. At any point in development, one can run a procedure in open loop mode, but behavior will be inaccurate until the optimal timing parameters have been established.

We believe that many procedural concepts are initially learned through imitation, and that this provides the agent with a basic "plan" which can be run in closed loop mode. However, before the procedure can be used efficiently, it must be "fine--tuned" through a process of trial and error, in which slightly different state descriptions are tried (such as moving the arm farther back). This fine tuning occurs only in closed loop mode, since one requires detailed trace information about body positions to modify the state descriptions appropriately. Simultaneously, the timing parameters necessary for open loop behavior are estimated, but these estimates will not become

stable until the state descriptions themselves stabilize. This causes open loop learning to lag behind closed loop learning.

## THE CONSTRUCTION OF COGNITIVE MAPS

Before an intelligent agent can interact with a complex environment, it must have some model of that environment. We plan to represent such knowledge using "cognitive maps", in which the positions and orientations of local objects are specified in terms of a three-dimensional coordinate system. Like object concepts, these cognitive maps summarize experience over long periods of time, and will be "fuzzy" in cases where the component objects do not have constant locations. For example, the walls and windows in a room seldom move, while the furniture may well change over time.

While local maps are similar to object concepts, global maps are analogous to procedural concepts. Thus, global maps store route information in terms of the operators required to traverse those routes, and they refer to local maps in their state descriptions; the latter can be used as landmarks to check one's progress towards the goal state. Just as procedures are stored hierarchically, so are global maps stored at multiple levels of aggregation, with lower level routes being treated as operators by higher levels. Local maps are used extensively in manipulation, while global maps must be accessed for navigational planning.

In our framework, local maps are acquired through a process similar to that used to form object concepts, but which operates in a "piecemeal" fashion. Since the visual field cannot take in the whole surroundings at once, the learner must look in different directions to construct a model of his local surroundings. Moreover, he may acquire knowledge of object positions by different senses (such as sight and touch), which are then combined into a coherent whole. Although local maps can be formed simply by observing one's immediate surroundings, global maps require the exploration of adjacent local environments. Just as the agent stores sequences of actions that lead to a successfully thrown ball, so he stores the sequences of steps that successfully lead between two locations. In this framework, local maps will be acquired before higher level structures, and this agrees with our intuitions about the development of spatial knowledge.

## THE DISCOVERY OF PROBLEM SOLVING HEURISTICS

In addition to employing stored procedures, an intelligent agent must be able to generate plans dynamically in response to new situations. This

requires some form of problem solving, and we plan to use a version of means--ends analysis [268] to satisfy this need. This approach provides methods for dynamically generating subgoals as "stepping stones" to accomplishing higher level goals. Goals are represented as (desired) state descriptions similar to cognitive maps; they can be more or less abstract.

Our approach to means--ends analysis requires transparent descriptions of operators (in terms of their preconditions and expected results), so these must be learned from experience. We believe that many instances of an operator's application must be observed before a general description of its effect can be learned, and that this process is very similar to that used in forming object concepts. In addition, means--ends analysis does not eliminate search, and heuristics must be discovered to constrain the search process. We plan to adapt methods for assigning credit based on solution paths [346], so that heuristic conditions can be learned from subpaths whenever a subgoal is achieved.

## DISCUSSION

Although we have described the components of our architecture, we are not yet committed to an overall control structure, and this is clearly a direction for future research. However, even the components indicate that our approach to modeling learning differs significantly from earlier Machine Learning work, and we should briefly consider these differences. First, our object concepts and procedures are closely linked to the physical world, and this has led us to represent them using a hybrid numeric--symbolic scheme, rather than the purely symbolic approach used in most AI systems. This representation also lets us account for the "fuzziness" inherent in many human concepts.

Second, our representations for object concepts, procedures, cognitive maps, and goals are all very similar to each other, and all are closely related to the descriptions generated by the agent's sensory interface. This should help us achieve our long--term goal of an integrated model of learning, since different components should be able to communicate easily. Finally, we have emphasized the importance of goals in both indexing concepts and directing the learning process. We believe that such goal-based mechanisms are essential in dealing with a complex environment, since they impose the additional structure required to organize observed objects and events. In our future work, we plan to use these constraints to direct our search for a viable model of learning in complex, reactive environments.

**ACKNOWLEDGMENTS**

We would like to thank the members of the UCI World Modelers Project -- David Benjamin, Dan Easterlin, Howard Henry, and Don Rose -- for useful ideas that we have included in the paper. This research was supported by a gift from Hughes Aircraft Company.

# THE DEVELOPMENT OF STRUCTURES THROUGH INTERACTION

Robert W. Lawler

GTE Laboratories, FRL  
Waltham, Ma 02254

Vygotsky asserts that the internalization of socially rooted and historically developed activities is the feature distinguishing between human and subhuman psychology<sup>1</sup>. Piaget claims that the development of knowledge can only be understood through appreciating the history of structural changes deriving from experience<sup>2</sup>. Lewin argues that a central task for the cognitive sciences is going beyond the search for statistical norms of behavior to the detailed consideration of particular cases of individuals interacting with their milieu<sup>3</sup>. I have studied the learning of one child in breadth and in detail through a period of six months and more to trace the development of her cognitive structures. [203] The general objective of THIS effort is to use detail of THAT study to address some of the cited themes in a more computational fashion than was attempted in [203]. Some progress has been made to date, reported in "Learning Concrete Strategies Through Interaction" and "The Internalization of External Processes"<sup>4</sup>.

## THE IMPORTANCE OF STRUCTURE

The structuralist assumption is important in explaining saltations of performance. An outstanding example comes from recent work in modelling the evolution of vertebrate flight. Caple, Balda, and Willis argue in

---

<sup>1</sup>See especially chapter 4 of [383].

<sup>2</sup>This position, which permeates Piaget's work, is encapsulated in his paradoxical slogan: There can be no development without structures, nor any structures without development. See [275] for an introduction to his ideas.

<sup>3</sup>See "The Conflict between Aristotelian and Galilean Modes of Thought in Contemporary Psychology" in [216].

<sup>4</sup>These essays are to be subsumed in a more ambitious work now underway with the working title "The Development of Structures through Interaction".

[55] that insectivorous dinosaurs' use of forelimbs to stabilize their movement while running and jumping to snap at bugs can explain, in terms of positive feedback in a physical model, the development of those forelimbs into wings capable of the power stroke of flight. The theory measures good adaptedness with the criterion of "foraging volume", the space within which the creature might capture bugs in a given time. A fast jumper that can stand on two feet will be better able to capture bugs from a larger space than will a slowpoke with four feet solidly on the ground.

Three different physical changes led to increases in foraging volume: bipedalism, increased running speed, and controlled jumping. Because of the physics of the world, limbs extended to control the trajectory during a high-speed jump experience lift. Since this lift increases with increased speed and since it also increases stability of the running creature, a positive feedback condition develops where more speed produces more lift and this in turn permits more speed. The creature takes off. Flattening of the forelimbs would increase lift, as would the effective extension of the proto-wing surface by the transformation of scales into longer feathers. Once in the air, those very limb movements which facilitate trajectory control, when coordinated, serve as the power stroke of flight. This story exemplifies the crossover in application of a structure with adaptive value for one function to some new function which provides a different adaptive advantage.<sup>5</sup>

## THE SEARCH FOR ANTERIOR STRUCTURES

Since structures exist in evolved physical things and the idea of co-adaptive evolution of structures is an effective a form of explanation, it is not bizarre to explore the extent to which information structures may be argued to exist in the human mind and to inquire about how they might prove useful in mechanical minds. For those who may hold suspect such analyses as cognitive studies permit, I note that unanticipated structures have been found even in physiological experiments. Satinoff argues in [315] that the smooth temperature control performance of mammals is based upon the multiple integration of disparate temperature control systems characteristic of earlier developed life forms. I have argued that mature skills can arise from such small but significant changes in the

---

<sup>5</sup>This is co-adaptive evolution of structures; "functional lability" names this re-applicability.

organization of pre-existing, fragmentary bodies of knowledge<sup>6</sup> which represent the things of everyday experience and operations on them. Piaget's "conservation" experiments are strong evidence that there exist organizations of knowledge about the world quite different from the collections of ideas held by most adults.<sup>7</sup> Such studies raise to salience the issue of what precisely the structural precursors of mature performance might be. If only we could specify the character and function of antecedent structures, we could explain large scale behavior changes as saltations emergent from minimal internal organizational changes. Even if one does not believe in the essential reality of large-scale information structures in the human mind, the precise articulation of problems of learning (both natural and artificial) from a structuralist position gives us a language for discussing issues which may be beyond the reach of more atomistic descriptions.

## FROM NEOPHYTE TO MASTER

Within a function-oriented structuralist view of human learning, a central problem is explaining the development of a person from neophyte to master. The specific case on which I am now working can characterize the general problem. A child has been observed beginning to play tictactoe strategically<sup>8</sup> by imitating a three move plan for establishing a fork another child performed. The characteristics of her knowledge at that time were egocentricity and particularity. EGOCENTRICITY: she did not attend to the moves of her opponent unless they directly interfered with her single plan. PARTICULARITY: when her sole plan was blocked, she was unable to develop any alternative. How, from such a beginning, is it possible for a mind to develop powerful notions, such as the offensive use of forcing in achieving forks, and methods of mastery, such as a scheme of games classification which permits immediate prediction of victory for over half the games possible to play once the first two moves are specified ? In this specific case, I believe the collection of processes for a computational model of such a path of development must involve at least these:

- data structures which embody the minimum encoding of

<sup>6</sup>Chapter 2 of [203] sets out such an argument.

<sup>7</sup>See, for example, [273].

<sup>8</sup>The study on which this description is based appears as Chapter 4 of [203].

knowledge needed to perform **actions learned from imitating others** to achieve a recognizable goal:

- interaction-driven **reflexive construction**, which includes the concrete learning of:
  - specific constraints upon egocentric plans
  - new goals recognized when surprises occur
  - new plans constructed as variations of prior plans to achieve newly recognized goals
- **redescription of goal elements of concrete data structures** based upon inversion of bug descriptions: the essential use of such processes is the introduction of new terms into the descriptions of goals and plans
- **representing the role of “society” in the internalization of external processes**: the constriction of the context of action, coupled with interior goals to continue activity, leads to functional re-application of structures developed for some other purpose to meet the challenge of the context-constriction; the importance of this process is in permitting the development of “virtual experience”; this amounts to the introduction of a Vygotsky-like view into modelling of mind.
- **reflexive abstraction**, a functional analysis of genesis, as exemplified in Bourbaki’s description of the generality of axiomatic systems:

A mathematician who tries to carry out a proof thinks of a well-defined mathematical object, which he is studying just at this moment. If he now believes that he has found a proof, he notices then, as he carefully examines all the sequences of inference, that only very few of the special properties in the object at issue have really played any significant role in the proof. It is consequently possible to carry out the same proof also for other objects possessing only those properties which had to be used. Here lies the simple idea of the axiomatic method: instead of explaining

which objects should be examined, one has to specify only the properties of the objects which are to be used. These properties are placed as axioms at the start. It is no longer necessary to explain what the objects that should be studied really are.... N. Bourbaki<sup>9</sup>

One use of such a process would be to explain the recognition that symmetry applies with power in the domain of tictactoe.

- **reflexive classification, a feature-oriented organization of things:** Piaget contrasts the former, reflexive abstraction, with classificatory or Aristotelian abstraction,<sup>10</sup> demeaning the latter somewhat by referring to it as "simple"; to the extent that classification is an automatic process, perhaps an outgrowth of such mechanisms of memory organizations as Schank [319] and Kolodner [177], have proposed, it must be distinguished from a consciously articulated application of such classification processes on a more abstract level. For example, a pattern-oriented classification of tictactoe games into types based on the first two moves permits an easy to use scheme by which masterful play can be achieved without forward play in the tree of possible games. The challenge is to imagine how a consciously articulated process of such classificatory abstraction could arise from an "automatic" side effect of memory organization; Piaget's answer would be, of course, by reflexive abstraction. The attempt to build a model with such a set of characteristics is underway.

---

<sup>9</sup>Cited in [113]. p.69.

<sup>10</sup>In [274]. p. 320.

# **COMPLEX LEARNING ENVIRONMENTS: HIERARCHIES AND THE USE OF EXPLANATION**

## **Michael Lebowitz**

Department of Computer Science, Columbia University  
New York, NY 10027

### **ABSTRACT**

At Columbia, our work on concept learning covers a wide range of topics involving complex learning domains. In this summary, we briefly describe three areas of recent research: generalizing hierarchically structured representations; an experiment in explanation-based learning; and first steps at integrating explanation and similarity-based methods.

### **GENERALIZING HIERARCHICAL STRUCTURES**

RESEARCHER is a system designed to read, remember and learn from patent abstracts. As many device patents are best represented as hierarchies of parts, RESEARCHER has as one of its goals the incremental generalization of hierarchical descriptions. This is done by finding similar examples in memory, comparing them with the new example, and abstracting out the similarities. Typical problems in generalizing hierarchical descriptions include: deciding how the components in the objects being compared correspond; dealing with differing levels of description of objects; and structuring memory so that maximal inheritance of the sort used in semantic networks and frame systems, which implies minimum space utilization, can be achieved automatically. The details of how we address these problems can be found in [385].

Since the examples given to RESEARCHER are not expressly designed for learning specific concepts (as they would be for a system being taught concepts), the program must decide which examples to compare for the purpose of generalization. This is done using a generalization-based memory (GBM) of the sort shown in Figure 1 and described in [207, 208]. A hierarchy of concepts is created in memory that organizes specific examples (although GBM requires a few twists to handle inheritance properly). Figure 2 shows a possible, simple GBM for disk drives. Note that: 1) the generalization hierarchy is automatically created by RESEARCHER, not provided to the system in advance, and 2) each node in GBM represents a complete hierarchical description of the kind we have been looking for (in

effect, RESEARCHER's memory is a hierarchy of hierarchies).

```
generalized object----->specific instances
| 
| > more specific generalized object----->specific instances
| 
| > still more specific generalized object->specific instances
| 
| > still more specific generalized object->specific instances
| 
| > more specific generalized object----->specific instances
```

**Figure 1:** Schematic illustration of generalization-based memory

```
disk-drive# -----> patent A
| 
| > floppy-disk-drive# -----> patents B, C
| 
| > single-sided-floppy-disk-drive# ----> patents E, F
| 
| > double-sided-floppy-disk-drive# ----> patents G, H, I
| 
| > hard-disk-drive# -----> patents I, J, K
```

**Figure 2:** Possible generalization-based memory for disk drives

The current implementation of RESEARCHER's generalization scheme works quite well on modest-sized patent examples, including the insertion of implicit levels in hierarchies to improve matches. A modified version of the program, CORPORATE-RESEARCHER, has been tested on hierarchical descriptions of corporate organizations. In the future, we plan to: address some of the combinatoric problems that arise for large examples; consider whether our approach of abstracting out all possible similarities is too extreme (in particular, determining exactly what the generalized concepts signify); apply confidence evaluation methods of the sort used in our other work to refine initial generalizations; and look at the startup problems in generating a generalization-based memory.

## EXPLANATION-BASED LEARNING

In the last few years, a new approach has become popular in the machine learning field — *explanation-based learning*. This line of research views learning as a knowledge-intensive activity, much like other tasks in AI. The basic idea is that a program takes a single example, builds up an explanation of how the various components relate to each other using

traditional, domain-dependent AI understanding or planning methods, and then generalizes the properties of various components of the instance as long as the explanation remains valid. What is left is then viewed as a generalized description of the instance that can be applied to further examples. This kind of learning is tremendously useful, as it allows generalized concepts to be determined on the basis of a single example, and involves explanations that are already needed for important roles in understanding.

We are currently applying the idea of explanation-based learning to the design of logic circuits [108]. This is a domain where we have a relatively complete domain model, and hence can build up a good explanation with which to start the explanation-based learning process. The idea is that we take a set of device specifications along with a circuit that implements those specifications and build up a proof as to why the circuit satisfies the specifications. Then, we generalize the specifications and circuit such that the proof still holds. This allows us, for example, to generalize a shift register into a circuit schema that will implement arbitrary permutations of a series of input bits. In theory, this schema would then be available for later designs, and would be more widely applicable than a simple shift register. The task and general method we are using is related to that in [250], although we focus on developing circuit schemata, rather than designing by analogy.

## INTEGRATED LEARNING

UNIMEM [208] is a program that can accept a large quantity of relatively unstructured facts about a domain, use generalization techniques to determine important concepts, and use these concepts to organize the information in a fashion that allows further generalization and intelligent question answering. For example, if information about Congressional voting records is given to UNIMEM, the program might recognize that Congressmen who vote for the MX missile also usually vote for the draft. UNIMEM makes use of generalization-based memory of the sort described above. UNIMEM allows us to focus on the problems of learning without worrying about natural language input or hierarchical representations. Some of the problems in forming concepts from complex input data involved in our research with UNIMEM include: the impact of domain-dependent knowledge on concept learning; categorizing numeric input information so that generalization is possible; concept evaluation and refinement from further examples; using concepts that very slightly contradict new input items; dealing with concepts that change over time; and question answering

based on generalization-based memory.

As we mentioned above, explanation-based learning is providing an interesting alternative to learning by comparing similarities. As an adjunct to our work on UNIMEM, we are looking at how these two methods might be combined — applying explanation-based techniques during the course of similarity-based learning. For domains lacking detailed explanatory rules, this combination can achieve the power of explanation-based learning without some of the computational problems that can otherwise arise. We have looked at how the ideas of *interest* and *predictability* [207] can be particularly valuable in this context. Such work is necessary because the building and analysis of explanations does require extremely detailed knowledge of the domain, which may not always be available. In addition, virtually all current explanation-based learning work is in the "perfect learner" paradigm that assumes that all input is noise-free and fits the correct final generalization.

We have begun work involving a three step plan for integrated learning: applying explanation-based methods to generalizations derived by noticing similarities, instead of to individual examples; using *interest* to determine when to learn; using *predictability* to help control an otherwise unmanageable explanation process. This plan is more fully developed in [209]. It is intended to allow us to process large numbers of examples and deal with realistic, noisy data.

## ACKNOWLEDGMENTS

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165. A number of students have contributed to this work. In particular, the work on generalizing hierarchies was done by Kenneth Wasserman and on explanation-based learning by Tom Ellman.

# PREDICTION AND CONTROL IN AN ACTIVE ENVIRONMENT

Alan J. MacDonald

Kobler Unit, Dept Computing, Imperial College  
London, England, SW7 2AZ

## ABSTRACT

The definition of a mechanism which learns to predict the consequences of events and actions in an environment is in progress. Salient features of this research include an attempt to separate prior assumptions (bias) from the learning algorithm proper and, as far as possible, to make them explicit. The emphasis is on events rather than objects - the input from the environment is not restricted to be a succession of object and relation oriented descriptions of environmental state. Of particular interest is the acquisition of such object based representations. The task domain (learning to predict in a particular class of environments) is defined with these aims in mind.

## INTRODUCTION

The approach to machine learning outlined here was originally stimulated by George Kelly, who proposed [168] that it would be profitable for psychologists to emphasise man's scientist-like characteristics (as opposed to, say, man's rat-like characteristics). Kelly, in his Personal Construct Theory (PCT), abstracts man as an active exploratory agent whose only goal is to better anticipate events: for Kelly *all* psychological functioning is a matter of learning.

While this research is derivative of PCT, it has become progressively less recognisable as such as the analysis and formalisation [219] of Kelly's basic notions have proceeded. Nevertheless, many of Kelly's major concerns remain central. Thus the objects of study, like the persons of PCT, are systems which continue indefinitely to improve their predictive ability in an environment by "identifying **replications** in events". A replication for Kelly is "anything which repeats" but otherwise this central concept is undefined. While a fundamental assumption of PCT is the view that structure is attributed to events rather than intrinsic to them, resulting in a strong emphasis on the individual nature of each person's "theory of the world" and on the consequently relative nature of his perception, replication itself

seems to be assumed as an absolute.

A system, in order to identify a replication, requires some predefined notion of similarity. Thus replication is not absolute - it is relative to a set of criteria which constitute a major source of prior assumptions or "bias" in a system. Bias in learning systems is often procedural and implicit, with consequences in the task domain which may be difficult to interpret or predict. The intention here is to gain flexibility and "transparency" by maximising the proportion of bias expressed in this explicit declarative form relative to other implicit sources of bias.

Given this emphasis a predictive system therefore requires an initial set of criteria by which events may be judged to be similar, a means of utilising these judgements for predicting events, and a method for the construction and evaluation of new criteria. The task domain is a set of simple environments broadly embodying some general characteristics of the physical world. One of the longer term aims of this research is to obtain some sort of characterisation of the minimal assumptions required for successful prediction in such environments.

## THE DOMAIN: ENVIRONMENTS

A suitable class of environment should capture some very general properties of the real world yet, at the same time, a typical member should be relatively easy to define for instantiation by a tractable computational method. With the latter requirement in mind it has been decided to consider the implementation of only environments in which time, space, and variables are discreet. An attempt has been made to ensure that it is natural to attribute the following (not entirely independent) properties to a typical environment [219]:

- **Geometrical structure** - a notion of *place* or *locality*: If  $t(A,B)$ , the minimum time before events in locality A can have any consequences in locality B, is a metric on localities then locality broadly corresponds to the intuitive idea of place.
- **Limited local complexity**: Both the complexity of the internal structure of each point in the environment and of its relationships with other points should be limited.
- **Decomposability**: An incomplete theory of the environment should be predictive. Satisfaction of the previous two conditions makes this likely.

- **Uniformity:** There should exist general laws which remain predictive over time and in a diversity of situations. These should, in principle, be identifiable by some learning mechanism.
- **Richness:** An environment should allow improvement in predictivity to continue indefinitely - there should always be something unpredictable about an environment which may later become predictable. Ideally an environment should also appear to be unbounded or infinite.

Such discreet environments as above can be viewed as generalised cellular automata: whereas the common conception of cellular automaton is of a regular array of identical finite state machines, here the neighbourhood relation and the association of next state functions with points are restricted only by the listed conditions and their consequences. The static structure of an environment can be represented as an infinite weakly connected labelled graph where the set of edges is the neighbourhood relation and vertex labels determine the distribution of next state functions. Thus one way of specifying an environment for simulation is to define the associated graph. Okabe [270] shows how an infinite labelled graph can be defined by a finite set of languages and his method is being adapted for use in this context.

An environment of this type might appear closer to a quantum mechanical world than that of everyday experience, but a variety of relevant "macroscopic" phenomena can appear at a fairly small scale in a cellular automaton. A learning mechanism interacts with an environment through an interface to a locality of bounded volume. Only a portion of an environment is actually constructed around a learning mechanism as it moves around and thus in general the specified environment will not be simulated exactly, but only approximated.

## AN ARCHITECTURE FOR A PREDICTIVE MECHANISM

The system under development (ETP) interacts with its environment through a **boundary** which is a vector whose successive values represent successive states of the system's sensors and effectors. The sequence of values of the boundary is stored, along with internally generated judgements of similarity as the **event history** of ETP. ETP assumes that the components of the boundary are individually predictive - that there is a helpful mapping between components of the boundary and aspects of events in the environment.

The major components of the system are:

1. *An initial set of replicative criteria (RCs):* these are fixed criteria by which events are judged similar. Components of the boundary are the most basic RCs, since each partitions the set of possible boundary values and is used to partition events. The boundary is memoriless - it reflects only the current state of the environment. Also it embodies only weak assumptions about the environment. ETP may therefore be equipped with arbitrarily complex criteria operating over the event history whose judgements are added to it and become inputs to the prediction algorithm. For example, if it were required that the system a priori treat all palindromes of greater than some specified minimum length as similar, then this feature would appear in the event history where appropriate.
2. *A basic algorithm for producing predictions:* This is a generalisation of a simple incremental algorithm [219] which, given a sequence of symbols as an input stream produces, on receipt of each symbol, a set of possible next symbols. The current algorithm takes a sequence of sets (the event history) as input, and interprets each element as signifying the occurrence of an event satisfying an external criterion of similarity. Thus each symbol records the the identification of a feature of an event. The algorithm also requires an external function which calculates the intersection of the extension of a pair of RCs and assigns new tokens to these intersections as necessary. Given these data, the algorithm incrementally constructs a partially ordered data structure in which is represented every replicated sequence of features which makes a non-redundant prediction. The basic algorithm can be very space hungry, but each node can be evaluated independently for predictive performance, and in consequence fairly straightforward heuristics for controlling space consumption can be defined.
3. *A method for identifying new RCs:* This arises from the highly restricted nature of the regularities that the prediction method can identify. The task of acquiring new RCs resembles a standard concept learning task: Given a performance history of the prediction method, the object is to characterise sets of events which have similar consequences (relative to current judgements of similarity). The use of current RCs (including those in the boundary) may also be abandoned and thus cease to be added to the event history. This component will exhibit

bias and, as noted previously, the precise nature of the relationship of this bias to performance in different domains may be obscure. It is intended initially to use an existing concept learning method (eg version spaces) in which bias is made fairly explicit.

## DISCUSSION

Obviously the development of this system is in its early stages and the stated aims have only been partly realised. There are actually three sources of bias in the system, only one of which appears the form of replicative criteria. The heuristics by which the prediction algorithm controls its space consumption, the means by which useful new RCs are constructed, and the interaction of these two processes are likely to produce biases which will be difficult to relate to the task domain. The use of action and experimentation for accelerating learning remains to be considered. This imposes more stringent constraints on the specificity of the predictions produced - currently a large set of predictions are generated, each consisting of a set of events. A third issue is that of efficiency: the concern at present is to achieve "epistemological adequacy" while satisfying only gross tractability constraints.

## ACKNOWLEDGMENTS

This research is supported by the Science and Engineering Research Council under IT fellowship B/ITF/64.

# BETTER INFORMATION RETRIEVAL THROUGH LINGUISTIC SOPHISTICATION

Michael L. Mauldin

Computer Science Department, Carnegie-Mellon University  
Pittsburgh, PA 15213

## ABSTRACT

An algorithm is suggested for learning sketchy scripts for a skimming parser. The parser is to be part of a knowledge-based information retrieval system. A case frame matcher will match an abstract of the user's query against abstracts of the texts, and should provide significantly better "recall" than keyword based IR systems.

## INTRODUCTION

Recent research has shown that although keyword-based information retrieval techniques are reasonably successful for moderate databases, they fail on larger databases. There are two measures of IR system performance: *recall* measures how many relevant documents are found, and *precision* measures how many irrelevant documents are found. While keyword based systems have good precision, they sometimes retrieve as little as 20% of the relevant documents from a large database [25]. Blair and Maron list many problems with textual databases, and most of their findings fall into two categories:

- Phraseology: synonyms, slang and jargon terms obscure the meaning of the text from keyword approaches.
- Granularity: as database size increases, increasingly fine-grained searches are necessary. Retrieval techniques that only consider the presence or absence of words cannot distinguish different relationships between the same words, and retrieve far more documents than the user considers relevant.

The real answer to both of these problems is to bite the bullet and really read the text. By understanding what a document is about, we can better determine whether it is relevant to the user's query.

## TEXT SKIMMING

DeJong's Frump [85] successfully analyzed 10% of the news stories on the UPI wire, requiring an average of 20 seconds per story. Frump's understanding of a story is based on "sketchy scripts," case frame structures which encode knowledge about the roles of common actions and events. Frump is composed of two main modules: the *predictor* which uses the script to constrain the meaning of the input, and the *substantiator*, which uses those constraints to analyze the text. Although Frump is fundamentally top-down (predict then substantiate) there is also a bottom-up component to the analysis (mainly for proper script selection).

DeJong claims that about half of the UPI news stories are analyzable with sketchy scripts, and lists three limitations of the original program:

- Lack of a necessary script.
- Undefined vocabulary words.
- Unknown/complex sentence structure.

This work focuses on learning new scripts, although it is hoped that the mechanism used will also be applicable to learning vocabulary at some future date. If such an algorithm can be developed, then text skimming can become a useful tool for IR systems.

## INFORMATION RETRIEVAL BASED ON TEXT SKIMMING

Frump's output was one or more instantiated sketchy scripts. We can think of these as *abstracts* of the text. Using a skimming parser to build abstracts, the following steps are used to retrieve documents:

1. Parse the user's query into an abstract.
2. Parse each document into an abstract (this can be done off-line).
3. Use breadth-first case frame matching to determine those abstracts most closely matching the abstract of the query.
4. Display the selections, and have the user pick from a menu those which are relevant (providing feedback to the learner).

DeJong only counted a parse as correct when the resulting frame contained enough information to produce a natural language summary of the story. For our purposes, the abstract need only tell enough about the text to properly retrieve it: a less stringent criterion.

## LEARNING SKETCHY SCRIPTS

The challenge is for the system to learn scripts from analysis of the textual data with only limited feedback. The bottom-up nature of initial script selection suggests one possible mechanism. The parse fails when several non-terminals are present on the stack without a corresponding higher level non-terminal in the grammar.<sup>1</sup> A "sketchy script learner" could easily hypothesize a new script with some combination of these non-terminals as slot fillers. Figure 1 illustrates how such a system would operate. The algorithm proceeds as follows:

1. Parse user's query.
2. If there are unparsed fragments, hypothesize several new scripts using the the fragments.
3. Retrieve the documents matching the parse (using unparsed fragments but not the new scripts).
4. The user selects the relevant documents, and the system remembers that set.
5. (Off-line) reparse all retrieved texts (and optionally some or all of the database).
6. Retrieve a new set of texts for each hypothesized script.
7. Retain those new scripts which have a better precision value without discarding any documents the user considers relevant.
8. Periodically reparse the entire database to incorporate the new knowledge.

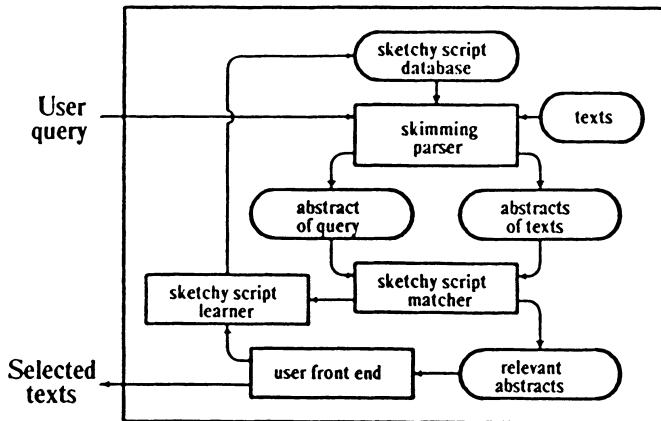
This method is modeled after so-called "Genetic Algorithms" which learn by hypothesizing new constructs and then scoring and retaining the best performers (see [148, 222]). GAs depend on two mechanisms for success:

- A means of suggesting plausible new structures.
- A means of ranking competing structures.

Since we can do both in the IR domain, there is a good chance that the resulting system will be successful.

---

<sup>1</sup>In Frump these non-terminals are represented by Conceptual Dependency graphs.



**Figure 1:** Knowledge-based Information Retrieval

## SUMMARY

By determining the content of a document rather than merely analyzing key words, I hope to produce a full-text IR system with improved recall performance. For this system to be attractive, it must learn new scripts to keep up with a large and changing database. Using failed parses to drive a hypothesizer, and credit assignment based on user feedback, a method akin to genetic search will be used to provide the new scripts necessary.

# **MACHINE LEARNING RESEARCH IN THE ARTIFICIAL INTELLIGENCE LABORATORY AT ILLINOIS**

**Ryszard S. Michalski**

University of Illinois at Urbana-Champaign

This note provides a brief account of major projects on Machine Learning done in the Artificial Intelligence Laboratory at the Department of Computer Science of the University of Illinois. Some of the projects are done in collaboration with research groups from other organizations. A few projects mentioned here are described in more detail as separate reports in the book. Research conducted by Professor G. DeJong and his group at the Coordinated Science Laboratory is reported separately. References contain publications describing details on projects described below and closely related work. Due to the space limitations these references include only a selection of papers from those written in the last year.

Our research has several objectives: to develop unifying principles of learning, to investigate relationships in learning and inference methods employed by people and machines, to develop experimental learning systems, and to test the methods developed on selected practical problems in various domains. Within the spectrum of scientific approaches to machine learning one can distinguish two extremes: general-purpose knowledge-free, and domain-specific knowledge-intensive. The approach taken by our research group represents a middle range between the two; it can be called domain-generic knowledge-modular. Within the large space of different learning processes one searches for generic domains, which represent classes of conceptually related tasks. A research goal is then to develop methods and programs for chosen generic domains. As examples of generic domains consider learning structure-based descriptions from examples, conjunctive conceptual clustering or predicting sequence of events characterized by attributes. Solving any specific task from a generic domain requires that the program for this domain be supplied with a module of knowledge relevant to the given task. For different tasks different modules are developed. The next task is to develop an integrated learning system that includes various learning subsystems, each of which serving as an expert on problems within a given generic domain. Several projects conducted by our group relate to selected generic domains. Below is a brief description of major current projects.

## INFERENCE-BASED THEORY OF LEARNING

A learning process can be viewed as a transformation of information provided by a source of information (a teacher or environment) into a more efficient and/or more general form in which it is stored for future use. This transformation involves the background knowledge of the learner (that includes goals of learning, concepts learned from the past experience, hypothesis evaluation criteria, etc.), and uses various types of inference. The type of inference employed in any given act of learning depends on the source of information and the goal of learning.

The inference type defines the learning strategy. For example, when learning a concept from examples, explaining the behavior of a system, or discovering an equation that binds specific numeric values, one employs inductive inference. Therefore these are forms of inductive learning. The inductive learning strategy is complex and inherently fallible, but it frees the teacher from spoon-feeding the student. The other extreme is rote learning which requires little or no inference from the student, but requires that the teacher provide all the information in a form directly absorbable by the student. This project investigates the trade-offs between various learning strategies, and attempts to develop a unifying theory of learning.

## INDUCTIVE LEARNING OF CONCEPTS, RULES AND PROCEDURES

Following earlier work on the AQ11 and INDUCE concept learning programs, this project is concerned with the development of advanced multi-purpose learning systems for various classes of problems, and their application to different practical problems. The knowledge representation formalism is based on various forms of variable-valued logic and the annotated predicate calculus (a typed predicate calculus with additional operators plus annotations). Recently studied topics include methods for incremental learning and refinement of rule bases, automated generation of relevant attributes and terms, learning from incomplete and/or noisy examples, learning rules with exceptions, and inferring causal models from facts. Selected results are described in more detail in separate reports in this book: Igor Mozetic describes how one of our inductive learning programs refined and simplified by two orders of magnitude a knowledge base for diagnosing cardiac arrhythmias, and Kaihu Chen describes the work on learning procedures from examples. Results on learning with exceptions and from noisy data are described by Becker. Other results are described in reports listed in references.

## PROBABILISTIC LEARNING

This project, directed by Professor Larry Rendell, investigates probabilistic learning systems capable of forming concepts and optimizing problem solving strategies using a form of conceptual clustering. Such systems can automatically split a problem solving state space into equivalence classes, which greatly improves the efficiency of a search process. They can handle noisy data and improve incrementally. Current research includes development of general principles and models for efficient induction, and the investigation of the role of probabilistic methods in learning. More details on this work are in a separate report by Rendell.

## CONCEPTUAL CLUSTERING AND AUTOMATED DESIGN OF CLASSIFICATIONS

This project, co-directed by Professor Robert Stepp and the author is concerned with problems of dividing a collection of observations into separate classes via conceptual clustering. Classes are created not because objects are similar according to some predefined numerical measure of similarity, but because they relate to each other in such a way that together they constitute (or can be explained by) a simple concept. We have developed two conceptual clustering programs, CLUSTER/2 for building classifications using attribute-based descriptions, and CLUSTER/S for building classifications using structure-based descriptions. Current research involves problems of building goal-oriented classifications and automated generation of relevant descriptors. A separate report by Stepp gives details on some of the recent work.

## LEARNING TO PROGRAM BY ANALOGY

Learning by analogy can be viewed as inductive and deductive learning combined. Through induction the system finds a common substructure in the entities being compared, and then by deduction uses this substructure for analogical mapping. This project, led by Professor Dershowitz, uses analogy as a tool in automatic programming. To determine a program for a new problem, the specification of the problem is compared with the specification of an existing program. Through analogy, the existing program is modified to meet the new specification.

## QUALITATIVE PREDICTION

This research aims at developing methods for discovering rules and relationships characterizing discrete processes and then using these rules and relationships to predict the continuation of the process. As examples of such processes consider a sequence of states of a patient under treatment, or stages in the development of a crop disease in a field. This research started with a successful development of a system SPARC/E (sequential pattern recognition) for discovering rules in the card game Eleusis. This game provides a microcosm modeling the process of scientific discovery. Recent work involves the development of a program SPARC/G for solving a general class of non-deterministic prediction problems, and an application of the program to various domains, such as robotics and biology. More details on this project are provided in the references. Some aspects this of research are described by Heedong Ko and Kaihu Chen in separate reports in the book.

## QUANTITATIVE DISCOVERY SYSTEMS

In many areas of science, there occurs the problem of discovering a numerical law explaining or characterizing a given set of observations. Attempts to automate some parts of this process were made in pioneering work by Simon, Langley and collaborators on the development of various versions of the BACON system. Our research strives to extend the work on BACON and combine it with our concept learning systems. The system ABACUS, recently developed by Falkenhainer in his M.S. thesis, can not only to discover qualitative laws that characterize given sets of observations, but also formulate the areas of applicability of these laws. When different subsets of data are governed by different laws, the program formulates the laws relevant for each subset, and provides conceptual descriptions defining preconditions for application of each law. A new project, done in collaboration with Nayel El-Shafei from MIT, attempts to develop a system integrating methods for quantitative and qualitative discovery.

## KNOWLEDGE ACQUISITION FOR EXPERT SYSTEMS THROUGH INDUCTIVE LEARNING

In the context of research on the meta-expert system ADVISE, directed by Baskin, Michalski and Stepp, we are investigating methods for building expert systems with inductive learning abilities. ADVISE is an

integrated set of tools for designing and experimenting with expert systems with multiple control strategies, different methods for propagating uncertainties, several knowledge representations, and learning capabilities. A module of ADVISE, called QUIN (for query and inference), provides a set of learning and inference operators that apply to data in relational table format. These operators can generate decision rules from examples, formulate classifications, select relevant attributes or determine representative learning events.

## **VARIABLE-PRECISION LOGIC**

The concept of variable-precision logic was introduced by the author and Patrick Winston from MIT's Artificial Intelligence Laboratory. This logic is concerned with problems of reasoning that represents different trade-offs between certainty of conclusions, their specificity, and computational resources involved in deriving these conclusions. A paper finished recently on this subject uses censored production rules as a representational and computational mechanism for handling some of these trade-offs and learning with exceptions. Some aspects of VPL were recently implemented by Becker in his M.S. thesis work.

## **HUMAN PLAUSIBLE REASONING AND CONCEPT FORMATION**

This research involves two projects, one in collaboration with Alan Collins from Bolt, Beranek and Newman, Inc., and the other with Doug Medin from the Department of Psychology at the University of Illinois. The goal of this research is to develop a theory explaining how humans conduct plausible reasoning, how they learn concepts, and how they generate classifications. A paper describing the core of a perturbation theory of human plausible reasoning is in its final stages. Recent research involved experiments comparing human and machine performance on selected problems of concept formation and classification development. A paper describing the results is in preparation.

## **ACKNOWLEDGMENTS**

The support of the research done in the Laboratory is provided by the National Science Foundation under grants DCR 84-06801, MCS 83-07755, and by the Office of Naval Research under grant No. N00014-82-K-0186.

# OVERVIEW OF THE PRODIGY LEARNING APPRENTICE

## Steven Minton

Carnegie-Mellon University, Department of Computer Science  
Pittsburgh, PA, 15213

### ABSTRACT

This paper briefly describes the PRODIGY system<sup>1</sup>, a learning apprentice for robot construction tasks currently being developed at Carnegie-Mellon University. After solving a problem, PRODIGY re-examines the search tree and analyzes its mistakes. By doing so, PRODIGY can often find efficient tests for determining if a problem solving method is applicable. If adequate performance cannot be achieved through analysis alone, PRODIGY can initiate a focused dialogue with a teacher to learn the circumstances under which a problem solving method is appropriate.

### INTRODUCTION

The purpose of this paper is to briefly outline current research on the PRODIGY system at Carnegie-Mellon University. PRODIGY is a member of the class of *Learning Apprentice Systems*, which Mitchell, Mahadevan & Steinberg [254] have defined to be "interactive knowledge-based consultants that directly assimilate new knowledge by observing and analyzing the problem solving steps contributed by their users". Although the learning and problem-solving components of PRODIGY are intended to be domain-independent, the program is initially targeted for a "robot construction" domain. This domain involves complex blocks-world tasks such as building bridges, building towers that require scaffolding, etc. Ten years ago, Fahlman [111] characterized the learning problem for this domain as "beyond the state of the art".

PRODIGY's problem solving component is an advanced STRIPS-like planning system. After solving a problem, PRODIGY generalizes the solution by analyzing the constraints inherent in the solution. This approach has come to be known as *explanation-based learning* [271]. The next two

---

<sup>1</sup>PRODIGY is being jointly developed by the author, Jaime Carbonell, Craig Knoblock, and Daniel Kuokka.

sections briefly describe some of the major differences between PRODIGY and other explanation-based learning systems. In particular, we will describe how PRODIGY focuses its learning so that effective control knowledge is learned, and how PRODIGY augments its knowledge base by interacting with a human expert. The reader is cautioned that the system is currently under development, and at present only the basic problem solver and the rudimentary generalization mechanisms have been implemented.

## LEARNING EFFECTIVE CONTROL KNOWLEDGE

A PRODIGY domain consists of a set of STRIPS-like operators [119], each comprised of an add-list, delete-list and precondition-list, and a set of inference rules. Inference rules are used to prove that the preconditions of an operator are satisfied in a particular state. PRODIGY solves problems using a simple heuristic search. A complete solution consists of a sequence of operators and inference rules which "proves" that the solution actually solves the problem at hand.

It has been demonstrated that solution sequences of this type can be generalized by replacing problem-specific terms with problem-independent terms while maintaining the dependencies among the steps in the solution [243, 119, 256]. These generalized solution sequences can then be used as macro-operators during subsequent problem solving. One drawback to this approach is that the store of macro-operators may increase rapidly as the problem-solver gains experience. This is certainly true in the STRIPS paradigm, where subsequences of a single solution may themselves be used as macro-operators. Under such circumstances, searching through the space of stored macro-operators to find one that is best-suited to the current problem can eventually become very expensive.

One method of coping with this problem is to selectively learn only the most valuable aspects of a new problem solving method. An earlier paper [242] described a program called MORRIS that only generalized and saved solution sequences if they were frequently used, or if they were particularly useful for solving difficult problems. With PRODIGY, we are exploring a stronger, more analytic approach for extracting control knowledge from a problem-solving episode. In situations where the program finds that it has spent much of its time exploring dead-end paths in the search tree, it can re-examine expensive portions of the tree and analyze why each subtree succeeded or failed. Working from the leaves inward, the success and failure conditions for each subtree are generalized (by constraint back-propagation [371]), combined, and simplified by a theorem prover. In this way, applicability conditions can be produced that describe

when a problem-solving method is appropriate (or inappropriate).

In general, there is no guarantee that *concise* applicability conditions for a problem-solving method can be found, even if they exist. Simplification may require an arbitrary amount of theorem proving. However, it is often possible to detect cases when concise descriptions are likely to be found. For example, all paths in an entire subtree may depend on a single condition that is impossible to achieve under certain circumstances. Such cases can be detected relatively easily. In practice, problems involving interacting goals often show such a pattern.

By focusing the learning process on those cases where the payoff is likely to be greatest, PRODIGY can maximize the effective compression of the search space. However, if PRODIGY cannot find efficient applicability conditions on its own, it can ask for help. Because the search tree analysis pinpoints the difficult part of the problem, PRODIGY can initiate a focused interaction with an expert so that the expert is not forced to guess where the difficulty lies.

## FOCUSED HUMAN-COMPUTER INTERACTION

In the traditional "learning by examples" paradigm, communication between the teacher and the machine is extremely constrained, in that the teacher simply presents a sequence of examples to the program. While this form of communication is simple, it does not necessarily imply that the teacher's task is an easy one. One of the goals of the PRODIGY project is to increase the bandwidth between the teacher and the machine. When the machine has a problem it can indicate to the teacher exactly where the difficulty lies. The teacher may then communicate to the machine, in a natural manner, whatever domain specific knowledge is necessary for the machine to adequately perform its task.

One circumstance that requires the teacher's help occurs when PRODIGY has spent an inordinate amount of time solving a subproblem. In the simplest case, if PRODIGY is not able to solve the subproblem itself, it can simply ask for the solution (a sequence of operators that solve the subproblem). Unfortunately, in some circumstances generalizing the answer may not provide sufficient information to enable PRODIGY to avoid a prolonged search during subsequent problem solving, since (as described in the last section) analysis of the solution may fail to produce efficient control knowledge. Therefore, PRODIGY may also ask the teacher "why" his method is more appropriate than other methods which might appear to be applicable. The teacher can respond by describing the circumstances that make his method more appropriate than the others. Additionally, an

explanation can be provided - a proof that under these circumstances the new method satisfies some constraints that the alternative methods do not. (The teacher's descriptions and explanations must be expressed in the program's representation language -- a variation of first-order predicate calculus). Because the methods are "discussed" in the context of the actual task, the teacher's job is relatively straightforward.

This type of dialogue is a natural method for communicating an explanation to a learning program, and we expect it to be useful in a variety of circumstances (not just for acquiring control knowledge). Whenever the program's performance is unsatisfactory (detected by either the program or the teacher), the teacher can present the correct solution, and the program can isolate the rules that lead it astray. Assuming the teacher can explain why his solution is more appropriate, the program will be able to generalize the explanation, and correct or refine its knowledge base.

## CONCLUDING REMARKS

We have presented a brief overview of the PRODIGY system, an explanation-based learning system currently being developed at Carnegie-Mellon University. The two primary research objectives motivating this work are (1) to explore methods for analyzing search trees in order to extract effective control knowledge, and (2) to demonstrate that an analytic learning system can focus its interaction with an expert, enabling the expert to communicate explanations to a program in a natural, case-oriented manner.

# A LEARNING APPRENTICE SYSTEM FOR VLSI DESIGN

**Tom M. Mitchell, Sridhar Mahadevan, and Louis I. Steinberg**

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

## ABSTRACT

Learning Apprentice Systems are interactive knowledge-based consultant systems that directly confront the knowledge-acquisition bottleneck by learning from their users. This paper discusses the notion of Learning Apprentice Systems in general, and summarizes our efforts to develop a specific Learning Apprentice in the domain of VLSI design.

## INTRODUCTION AND MOTIVATION

It is by now well-recognized that a major impediment to developing knowledge-based systems is the knowledge acquisition bottleneck: the task of building up a complete enough and correct enough knowledge base to provide high-level performance. A number of researchers have attempted to develop automated learning methods to address this knowledge-acquisition bottleneck, but with very few exceptions (perhaps [46, 228]) machine learning methods have not yet produced useful knowledge acquisition tools.

This paper summarizes our research toward a new class of knowledge-based consultant systems designed to overcome the knowledge acquisition bottleneck by allowing the system to learn from its users without an explicit training mode. We define this class of *Learning Apprentice Systems* as follows:

*Learning Apprentice System:* an *interactive* knowledge-based consultant that directly assimilates new problem-solving knowledge by observing and analyzing the problem solving steps contributed by its users through their *normal* use of the system<sup>1</sup>.

---

<sup>1</sup>This notion of a Learning Apprentice System was suggested originally in [249].

Learning Apprentice Systems are thus intended to possess certain characteristics of human apprentices: they initially provide assistance in solving some class of problems, and by assisting in solving problems they gain experience from which they build expertise. They learn through the *normal* course of assisting in solving problems, and therefore do not impose on the user any additional burden for "training" the system. The user's view of the system is simply that it is assisting in solving his or her problem.

## LEAP: A LEARNING APPRENTICE FOR VLSI DESIGN

We are currently developing a Learning Apprentice System, called LEAP, that assists in (and learns about) the design of VLSI circuits. LEAP is being developed as an augmentation to a knowledge-based VLSI design assistant called VEXED [252]. VEXED provides interactive aid to the user in implementing a circuit given its functional specifications, by suggesting and carrying out possible refinements to the design. A large part of its knowledge about circuit design is composed of a set of *implementation rules*, each of which suggests some legal method for refining a given function. For example, one implementation rule states that *IF the required function is to convert a parallel signal to a serial signal, THEN one possible implementation is to use a shift register*. It is this kind of rule that LEAP is initially intended to learn.

The user interface provided by VEXED is well-suited to collecting training examples from which such rules can be inferred. In particular, given some circuit submodule with its desired functional specifications, VEXED uses its existing implementation rules to suggest possible refinements to the design of this submodule. The user may accept the advice (in which case VEXED refines the submodule accordingly), or he may override the advice and choose to manually refine the submodule. In those cases where the user intervenes, VEXED records the function of the submodule to be implemented along with the user's solution to this subproblem. This information provides a training example from which a general implementation rule is to be inferred.

LEAP uses an explanation-based generalization method to produce general rules from such specific training examples. By using its knowledge of digital circuits to verify that the user's circuit implements the desired function, the generalizer is able to determine which features of the given circuit are essential in the general rule, and which are not. The use of analytical, explanation-based generalization in this system offers two significant advantages over more empirical generalization methods: (1) it

allows forming a justifiable generalization from a single training example, and (2) it allows the system to screen incorrect training examples because it generalizes only from those training examples that it can explain/verify. LEAP and its learning methods are described in greater detail in [254].

## DISCUSSION

Given the previous lack of success of machine learning methods in addressing the knowledge-acquisition bottleneck, why should one believe that LEAP may succeed? There are three significant architectural features of LEAP that combine to make the learning task more tractable than in previous efforts:

- *The interactive nature of the apprentice consultant allows capturing training examples closely suited to refining the knowledge base.* In particular, training examples correspond to single problem-solving steps which are to be covered by a single rule (rather than a chain of rules). As a result, there is no credit assignment problem. Furthermore, the training examples acquired focus only on knowledge that is missing from the system, since this is the only time the user need intervene in the problem solving.
- *Analytical, explanation-based generalization methods are more powerful than previously employed empirical methods.* In particular, LEAP is able to produce justifiable generalizations of single training examples, and to screen errorful data. The circuit design domain allows for the system to have a sufficiently strong domain theory to guide this analytical method.
- *Because the knowledge base involves rules that are logically independent, the growth of the knowledge base is monotonic, and there is no need to update old rules when new rules are added.* Because the implementation rules in VEXED/LEAP suggest possible (not necessarily preferable) implementations of a given function, they are logically independent, and the addition of a new rule does not require reassessing the validity of existing rules.

In addition to LEAP, Learning Apprentice Systems are currently being developed in the domains of well-log interpretation [Winston (this volume)], robot construction tasks [Minton (this volume)], and medical diagnosis

[Wilkins (this volume)]. In task domains for which Learning Apprentice Systems are feasible, we expect that they will offer strong advantages over present architectures for knowledge-based systems. Many copies of a Learning Apprentice System distributed to a broad community of users could produce a base of problem-solving experience very large compared to the experience from which a human expert learns. For example, by distributing copies of LEAP to hundreds of circuit designers, the system (collection) would quickly be exposed to more example circuit designs than a human designer could hope to see during a lifetime. Provided that effective learning methods can be developed, this large experience base might offer the potential for developing significantly more extensive knowledge bases than can be achieved using current manual knowledge-acquisition methods.

## ACKNOWLEDGMENTS

We thank the members of the Rutgers AI/VLSI project for many useful discussions regarding the design of LEAP, and for creating the VEXED system in which LEAP is being constructed. Schlumberger-Doll Research has made the STROBE system available as a representation framework in which VEXED and LEAP are being implemented. This research is supported by the Defense Advanced Research Projects Agency under Research Contract N00014-81-K-0394, and by the National Science Foundation under grant DCS83-51523.

# **GENERALIZING EXPLANATIONS OF NARRATIVES INTO SCHEMATA**

**Raymond J. Mooney**

Coordinated Science Laboratory

University of Illinois at Urbana-Champaign Urbana, IL 61801

## **ABSTRACT**

This paper describes a natural language system which improves its performance through learning. The system processes short English narratives and from a single narrative acquires a new schema for a stereotypical set of actions. During the understanding process, the system constructs explanations for characters' actions in terms of the goals they were meant to achieve. If a character achieves a common goal in a novel way, it generalizes the set of actions used to achieve this goal into a new schema. The generalization process is a knowledge-based analysis of the narrative's causal structure which removes unnecessary details while maintaining the validity of the explanation. The resulting generalized set of actions is then stored as a new schema and used by the system to process narratives which were previously beyond its capabilities.

## **INTRODUCTION**

AI systems for understanding natural language narratives have relied heavily on schemata for stereotypical sequences of actions [81, 392, 88, 107]. Hand coding all the schemata a narrative processing system would need to process a wide range of text would be an arduous if not impossible task. Consequently, we have been working on a natural language processing system called GENESIS (for GENeralizing Explanations of Stories Into Schemata) which acquires new schemata in the normal course of processing narratives. After acquiring new schemata, the system is able to correctly process narratives that were previously beyond its capabilities.

GENESIS uses a non-inductive form of learning called *explanation-based learning* [90], which can be briefly defined as acquiring or modifying a problem solving method by analyzing the causal structure of a single problem solution. More specifically, the system uses what we call *explanatory schema acquisition*, which involves generalizing the successful problem solving behavior of an external agent into a new schema for

achieving an important goal. A learned schema can be used by the system to aid in understanding the behavior of external agents or in planning to achieve a specified goal. The system is fully implemented and an example sequence demonstrating the system's learning is given later in the paper. A longer description of GENESIS appears as [257].

## GENERAL SYSTEM ORGANIZATION

First, English input is processed by a parser into a *conceptual representation*, a case-frame representation which uses some *conceptual dependency primitives* as well as predicates for complex schemata. Currently, we are using an adaptation of Dyer's McDYPAR [107] for this purpose.

The basic task of the *understander* is to construct a causally complete representation called the *model*. A model for a narrative has explicit representations for all the inputs as well as the many inferences that must be made to causally connect them together. There are four types of causal links for connecting assertions in the model of a narrative. These are:

- precondition:* A link between a state and an action it enables.
- effect:* A link between an action and a resulting state.
- motivation:* A link between a volitional action and the beliefs and goals of the actor which motivated him to perform the action.
- inference:* A link between a state and another state which it implies.

Inferring causal relations necessarily employs a large amount of background knowledge which is stored in the *schema library*. The techniques and representations used in this process are similar to those used in past work in narrative understanding [81, 392, 88, 107].

In order to demonstrate the abilities of the understander, a simple question answering system is used to inspect the model. The reason why an actor performed a certain action is easily retrieved by inspecting the causal links between assertions in the model. A simple natural language generator for translating replies into English is also included as part of the system.

If an actor in a narrative achieves an important goal through a novel combination of actions, the *explanation* for how the goal was achieved is generalized into a new schema. The combination of actions which supports the achieved goal is generalized as far as possible without breaking any of the connecting causal links or violating the well-formedness of individual

actions and states. This generalized structure is then stored as a new schema and used to facilitate the processing of future narratives. The details of the generalization process are given in [257].

## AN EXAMPLE

Currently, GENESIS has acquired two new schemata. In one example, the system learns a schema for kidnapping for ransom. In the other, it learns a schema for someone burning his own building to collect the insurance. Here we will show the performance of the system on the kidnapping example. Before processing the following narratives, GENESIS has information about bargaining, capturing, and many other concepts; however, it does not have a schema for kidnapping for ransom. First it receives the following "test" narrative:

INPUT: Ted is the husband of Alice. He won \$ 100000 in the lottery. Bob imprisoned Alice in his basement. Bob got \$75000 and released Alice.

Processing Story... Finished processing. Ready for questions:

? (Who gave Bob the money)

Answer unknown.

? (Why did Bob lock Alice in his basement)

Cannot find sufficient reason for his action.

? (Why did Bob release Alice)

Cannot find sufficient reason for his action.

Notice that in this narrative it is not mentioned how Bob got the money or why Bob imprisoned Alice and then released her. Since the system does not have a schema for kidnapping, it cannot infer the missing information and construct a causally complete explanation of the narrative. Next, it is given the following narrative:

INPUT: Fred is the father of Mary and is a millionaire. John approached Mary. She was wearing blue jeans. John pointed a gun at her and told her he wanted her to get into his car. He drove her to his hotel and locked her in his room. John called Fred and told him John was

holding Mary captive. John told Fred if Fred gave him \$ 250000 at Trenos then John would release Mary. Fred gave him the money and John released Mary.

Processing Story... Finished processing.

John achieved the thematic goal: John has \$250000.

Generalizing...

Building new schema: CAPTURE-BARGAIN.

Ready for questions:

? (Why did John imprison Mary in his room)

So John and Fred could make a bargain in which John released Mary and Fred gave John 250000 dollars at Trenos restaurant.

? (Why did Fred give John the money)

Because John and Fred made a bargain in which John released Mary and Fred gave

? (Why did Fred make the bargain with John)

Because Fred wanted Mary to be free more than he wanted to have 250000 dollars.

? (Why did Fred want Mary free)

Because Fred was Mary's father.

Unlike the first narrative, this one is detailed enough to allow GENESIS to causally connect the individual actions. The resulting causal structure is then generalized into a new schema for kidnapping for ransom. Next, the system is given the first narrative again, and using the schema it has just acquired, it is able to infer the missing information and causally connect the actions. Consequently, it is able to answer the questions which previously it could not answer.

**INPUT:** Ted is the husband of Alice. He won \$ 100000 in the lottery. Bob imprisoned Alice in his basement. Bob got \$75000 and released Alice.

Processing Story... Finished processing.

Ready for questions:

? (Who gave Bob the money)

Ted gave Bob 75000 dollars.

? (Why did Bob lock Alice in his basement)

So Bob and Ted could make a bargain in which Bob released Alice and Ted gave Bob 75000 dollars.

? (Why did Bob release Alice)

Because Bob and Ted made a bargain in which Bob released Alice and Ted gave Bob 75000 dollars.

## RELATION TO OTHER WORK

GENESIS' generalization process is most similar to the method used by STRIPS to generalize planning sequences into new MACROPs [119]. However, unlike STRIPS, GENESIS generalizes actions and states as well as objects and locations, and generalizes the order of independent actions. For example, in the detailed kidnapping narrative, the ransom payer is the victim's father; however, the system realizes that the important thing is that the ransom payer value the victim's freedom more than material possessions. Consequently, the father state is generalized to any positive interpersonal relationship since such relationships normally result in this goal priority.

The learning process used by GENESIS is also closely related to a growing body of recent work in *explanation-based learning* [249, 243, 401]. One of the major differences between GENESIS and other explanation-based learning systems is that GENESIS increases its ability to understand external problem solving behavior rather than improving its own problem solving abilities.

## FUTURE WORK

Since the generalization processes used in most explanation-based learning systems are very similar, we are currently constructing a domain independent explanation-based generalizer. A prototype of this system has been tested on simple examples in various domains including a STRIPS robot planning example [119], a LEAP logic design example [254], a MA natural deduction example [271], and the Winston "cup" example [401]. We are currently in the process of changing the GENESIS system to use this domain independent generalizer. Another area for research is refining learned schemata when the system encounters an example which violates its expectations.

## **ACKNOWLEDGMENTS**

This work was supported in part by the Air Force Office of Scientific Research under grant F49620-82-K-0009 and in part by the National Science Foundation under grant NSF-IST-83-17889.

# WHY ARE DESIGN DERIVATIONS HARD TO REPLAY?

Jack Mostow

Rutgers University Computer Science Department  
New Brunswick, NJ 08903

## ABSTRACT

A derivational approach to learning and design, based on the notion of reusing a design by “replaying” an idealized history of design goals and decisions, has been touted for the past few years as a paradigm for both software and hardware redesign. However, the limited success of this approach to date suggests that it is less straightforward than it appears. Closer study reveals some difficulties that intelligent replay mechanisms will need to address.

## INTRODUCTION

Design is an important domain in which to study learning. The complexity of the problem space appears to demand that the search process incorporate learning. Moreover, the space is sufficiently regular to make learning feasible: a design can be analyzed to guide subsequent iterations on the same problem, or modified to solve a related problem. Recent research has emphasized the notion of solving a design problem by replaying the stored derivation of a solution to a similar problem, modifying the derivation wherever necessary to fit the new problem [58, 250, 390, 12, 260, 261]. Clearly an effective mechanism for designing new circuits, programs, or widgets by analogy with old ones could greatly enhance design productivity. However, this approach has not yet proved successful, and it is worth analyzing why.

We have focused on an important special case of derivational analogy, where the new problem is to reimplement a program given a change in the program specification, target language, resource constraints, or other aspect of the original problem.

The derivation describes the construction of the original program as a series of transformations leading from a high-level specification to the implementation, where each step may be anything from a low-level edit command to an application of a general correctness-preserving rewrite rule. In addition, the derivation describes the goal structure motivating the

sequence of transformations [260, 118, 367]. However, it is still an idealization of the actual design history of the program; for example, starting from a precise specification omits information about how the specification evolved.

As Scherlis has observed [259, 322], a derivation is itself a designed artifact. Our experience confirms that developing a derivation is a highly iterative process:

1. Start replaying.
2. Notice a bug (a step that can't or shouldn't be replayed) or possible improvement.
3. Localize the desired change to one or more parts of the derivation.
4. Patch the derivation appropriately.
5. Go back to 1 and repeat until satisfied.

To investigate the derivational approach in more detail, we performed two rather different studies. In one, CMU doctoral student Steve Minton analyzed a fragment of a program derivation from [118] to see what knowledge would be needed to replay it given various kinds of changes in the problem. In the other, I sketched out a transformational derivation for a simple two-robot algorithm from [356], and USC graduate student Randy Kerber implemented the derivation in the POPART transformation system [390]. The process of debugging the derivation and modifying it to fit POPART's limitations gave us considerable experience with POPART's automatic (though primitive) replay facility. These two studies exposed several difficulties in applying the derivational approach to software design, many of which do not appear specific to software, or even to design.

## DERIVATIONS AS IMPERFECT HYPOTHESES

An ideal derivation of a program would precisely delineate the class of problems that could be solved by repeating the same design decisions, and tell exactly how to do so. However, real derivations only approximate this ideal -- i.e., they are imperfect hypotheses about how some class of problems can be solved.

We can view a derivation as both a *predictive* hypothesis about which problems can be solved, and a *prescriptive* hypothesis about how to solve

them. However, this view depends not only on how the derivation is represented but also on the methods used to interpret it. For example, a simple method for predicting whether a given problem can be solved by replay is to try replaying the derivation to see if it breaks. A more sophisticated method would test explicit preconditions associated with the derivation. Similarly, a derivation represented as a procedure can be replayed simply by executing it. A more sophisticated method might re-interpret each step based on an analogical mapping between the old and new problems.

A derivation is liable to be too specific in some ways and too general in others, especially if it is constructed by recording the solution of a single problem.

The next two sections discuss replay errors caused by missing preconditions and context-sensitive references in derivations. The last section discusses the problem of patching derivations.

## MISSING PRECONDITIONS

A derivation describes preconditions that should be satisfied before replaying each step. These preconditions constitute a predictive hypothesis about the class of situations in which replaying the step will lead to an acceptable design. A missing precondition makes the hypothesis over-general; it may cause the step to be replayed when:

1. The step cannot be performed;
2. Replaying the step yields an incorrect result;
3. Replaying the step fails to achieve its original purpose; or
4. A different step should be performed instead; either it was unavailable when the original problem was solved, or the criteria for selecting among alternative steps have changed.

Whether a derivation is constructed by hand or inferred by “looking over an expert’s shoulder” [254], it is liable to omit some preconditions, thereby allowing steps to be replayed when they shouldn’t. This situation may be recognized immediately (as in case 1), several steps later, or not at all. A robust replay mechanism should help detect and diagnose such omissions.

For example, suppose the derivation explicitly describes the goal  $G$  achieved by some step (or sequence of steps)  $s$ , and a rationale  $R$  for why

applying  $s$  to the original expression  $e$  achieved  $G$ . By checking that the new expression  $e'$  satisfies  $R$  before applying  $s$ , a replay mechanism can prevent many errors of type 3. By testing  $G$  afterwards, it can detect and help diagnose errors caused by weak rationales: if  $e'$  satisfies  $R$ , but  $G$  fails anyway,  $R$  must lack a condition on some part or property of  $e$  that differs from  $e'$ .

## THE REFERENCE PROBLEM

A transformational derivation serves as a prescription for how to implement similar specifications, i.e., by replaying the derivation. Since a parameterized operator may be applied to the same expression in more than one way by instantiating its parameters differently, the derivation must describe not only which operators to apply but also how to instantiate their parameters. One way is to put the operator parameters in a *parameterized pattern* like "exp1+exp2." Matching this pattern against the expression to be transformed binds each parameter to the appropriate subexpression. Another kind of description for how to instantiate an operator parameter is a *descriptive reference* like "the call to FOO." Resolving this reference in the context of the current expression yields the desired subexpression. In either case, replay is vulnerable to the problems that can occur whenever a description developed in one context is used in another:

- Ambiguous references (over-general prescriptions): A description with a single match in the original context may have more than one in the new context.
- Non-existent referents (over-specific prescriptions): A description at too low a level may fail to match the expression to which the operator should be applied.
- Mistaken identity (context-dependent prescriptions): Even if a description has a single match in the new context, it may be the wrong one.

An intelligent replay mechanism must effectively interpret references that made sense in their original context but are anomalous in the new one. In particular, if the derivation was constructed by transcribing the behavior of an expert, the mechanism must infer salient higher-level descriptions of objects selected by the expert.

## DERIVATION PATCHING

So far we have focused on the problems of deciding whether and how to replay a step in a derivation. While very few problems can be solved by repeating the exact same sequence of steps, many more can be solved by adding, deleting, or changing some steps and replaying the modified derivation. However, this sort of extended use raises two issues.

- Localization: Given a change in the problem, which steps in the derivation must be modified?
- Context-sensitivity: A step in a derivation makes implicit assumptions about previous or subsequent steps. Each step relies on its predecessors to construct a context where it can be applied, and on its successors to complete the design process without hitting a dead end. Replay becomes problematic when implicit assumptions satisfied in the original context are no longer satisfied in the new context -- in particular, when the derivation is patched.

Since changing one step in a derivation may violate assumptions made in others, the derivation-patching process may have to change those steps as well, and so on, in order to make the modified derivation consistent. An intelligent replay mechanism should provide facilities for monitoring explicit assumptions and diagnosing violations of implicit assumptions.

## NOTE

To conform to length limitations, this paper omits over half of the version in the conference proceedings. For illustrative examples and more detail, see [262] (available from the author).

## ACKNOWLEDGMENTS

I thank David Wile, Steve Minton, and Randy Kerber for their ideas, Lou Steinberg, Rich Keller, Smadar Kedar-Cabelli, Michael Sims, John Roach, Pat Langley, and Tom Mitchell for their comments on various drafts, and USC Information Sciences Institute, where this research was supported by the Defense Advanced Research Projects Agency under contract No. MDA903 81 C 0335. Views and conclusions contained in this

report are the author's and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

# **AN ARCHITECTURE FOR EXPERIENTIAL LEARNING**

**Michael C. Mozer, Klaus P. Gross**

Carnegie-Mellon University, Department of Computer Science  
Pittsburgh, PA 15213

## **INTRODUCTION**

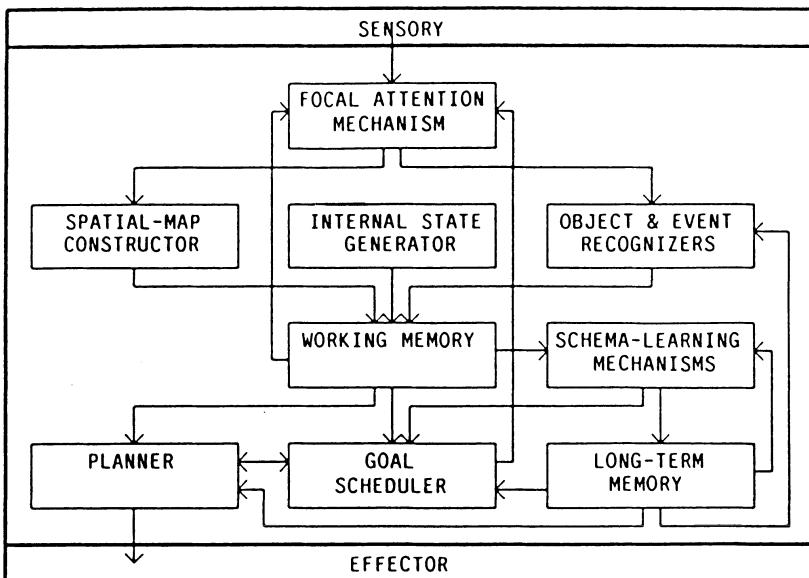
Our aim in this paper is to propose a cognitive architecture for learning in a complex, reactive environment. We hope that the architecture is sufficiently general that many forms of learning can be studied, yet not so general that it fails to constrain the nature of the learning task.

The architecture is that of an intelligent organism residing in the World Modelers environment [Carbonell (this volume)]. The World Modelers environment is a discrete-time simulation of a world that obeys a set of simplified physical laws. At each time step, an organism can interact with the world via a *Sensory-Effector Interface*. It may receive visual, auditory, tactile, olfactory, and gustatory information from the sensory commands and may act upon the environment through the effector commands (e.g., apply-force and turn-head). These commands are innate, non-adaptive, and serve as the basis for all interactions with the environment.

The World Modelers environment imposes several constraints on the organism architecture. First, the environment is information rich; hence, the organism must filter and condense the information available to its senses. Second, the environment is sufficiently complex that it cannot be completely predicted; hence, the organism requires mechanisms for handling unexpected situations. Third, the environment is reactive; hence, support should be available for active information gathering. Fourth, the environment provides an organism with information gradually over time; hence, the representation of temporal structure is necessary.

## **MAJOR COMPONENTS OF THE ARCHITECTURE**

The architecture is partitioned into ten basic components, shown in Figure 1 along with arrows indicating the primary flow of information. These components function as modules in the sense that the design and implementation of one is relatively independent of the others, and each operates in a relatively autonomous manner.



**Figure 1:** The Proposed Architecture

## INTERNAL STATE GENERATOR

The Internal State Generator generates internal states of the organism (e.g., hunger, thirst, sickness, reproductive urges) based on the organism's physical characteristics (e.g., the quantity of food and drink it requires, the types of food and drink that are nourishing, the period of the reproductive cycle). These states simulate the physiology and innate drives of the organism. While external stimuli may have a bearing on these states, such stimuli do not trigger the states directly; for example, the quantity of food ingested affects whether the organism becomes satiated, but the satiated state is triggered by the organism's "stomach". All states that are derived from the external senses (e.g., pain as a result of tactile pressure and fear at seeing a tiger) are generated by the Object and Event Recognizers.

## OBJECT AND EVENT RECOGNIZERS

The Object and Event Recognizers are responsible for producing descriptions of objects and events as perceived by the senses. These descriptions should be at a sufficiently high level that they may be used directly to suggest and formulate plans of action. In addition to directly-observable physical properties of an object or an event, these descriptions may include abstract properties such as the function of an object or the significance of an event.

All objects and events are interpreted in terms of existing knowledge structures. This knowledge is represented in the form of object and event *schemas*, which provide a framework into which object properties and event sequences are integrated. Schemas produce expectations (e.g., what color a banana should be or what an organism will do once it locates food). If these expectations are violated, a signal indicating the type and magnitude of violation will be sent to the Goal Scheduler.

We expect a high degree of interaction between Event and Object Recognizers. Each can help guide the search performed by the other. For example, a relevant event schema can focus the Object Recognizer on particular properties of an object. Similarly, properties of an identified object can focus the Event Recognizer on events involving objects with those properties.

## **Spatial-Map Constructor**

The Spatial-Map Constructor is responsible for creating a world-based description of the local environment (a *spatial map*). The spatial map contains information necessary to determine spatial relations among objects; for example, that food is a couple of feet behind the organism or that object x is located between objects y and z.

## **Focal Attention Mechanism**

The Focal Attention Mechanism determines what is interesting in the world given: (1) expectations produced by schemas, (2) states produced by the Internal State Generator, (3) the organism's goals, and (4) learned or innate biases associated with particular objects, events, or regions of the spatial map. The Focal Attention Mechanism is necessary because of resource limitations on the organism which prevent the processing of all perceptual data. The focus of attention can be on a specific object, a particular spatial region in the visual field, or perhaps even on a particular set of schemas, which would allow the schemas to perform more effectively. The focus of attention must also specify a granularity at which objects and events should be perceived.

## **Working Memory**

The Working Memory (WM) maintains a capacity-limited short-term store of (1) internal states of the organism, (2) recently recognized salient properties of objects, (3) recently recognized events, and (4) relations among objects. This information is provided by the Internal State Generator, Object Recognizer, Event Recognizer, and Spatial-Map Constructor, respectively. Because of capacity limitations on WM, the WM

requires a memory manager to determine what information to displace as the memory fills. If a request is made to the WM for information that is not present in the memory, then the memory manager will set up a goal to discover the information. This information can be discovered either by refocusing attention, or by physically repositioning the organism so that the information will be directly perceivable.

## LONG-TERM MEMORY

The Long-Term Memory (LTM) contains the organism's permanent, unlimited-capacity knowledge base. This knowledge includes object and event schemas, which can range from the specific, referring to a particular object or event, to the general, referring to a class of objects or events. Requests to LTM are in the form of partial descriptions of the desired information. For example, the Planner may request event schemas that satisfy a particular goal, or the Object Recognizer may request object schemas satisfying certain properties. Because the LTM knowledge base is so vast, a major design issue is how the knowledge should be organized.

## GOAL SCHEDULER

The Goal Scheduler is responsible for determining the relative priorities of the organism's current goals, and for resolving conflicts among competing goals. In doing so, the Goal Scheduler provides centralized control over the various components of the architecture. The Goal Scheduler must also determine when a goal has been satisfied, at which point the goal is removed from consideration. The Goal Scheduler might include a learning mechanism that allowed for the acquisition of context-dependent goal priorities. For example, if it is easy to satisfy the *obtain-food* goal in warm weather and difficult in cold weather, the Goal Scheduler may learn to adjust the priority of the goal depending on weather conditions.

Goals of the organism come in one of several forms: (1) goals for satisfying an internal drive, (2) goals to handle expectation violations produced by the Object or Event Recognizer, (3) goals requesting information about the environment directed by the Working Memory, (4) goals created by the Planner in the course of attempting to achieve a higher-level goal, and (5) goals for exploratory behavior produced by particular learning mechanisms.

## PLANNER

The Planner determines how to achieve the highest priority goals specified by the Goal Scheduler. Event schemas used by the Event

Recognizer can also be invoked for planning. An event schema, which specifies a sequence of sub-events, can be viewed as a plan for achieving the consequences of the event. The Planner must find an event schema or collection of event schemas whose consequences are the desired goals. The sub-events of a selected event schema in turn become subgoals of the plan, and the Planning process iterates. Ultimately, the Planner will send effector commands to the Sensory-Effector Interface.

## SCHEMA-LEARNING MECHANISMS

While we do not want to characterize all learning as such, many forms of learning in our architecture will involve the acquisition and modification of schemas in LTM. Collectively, we call these the Schema-Learning Mechanisms. Consider one such mechanism, learning by imitation, the goal of which is to acquire a new event schema based on observations of another organism carrying out some action. (This event schema can later be used for performing the action.) The mechanism must determine *when* learning should take place and *which* subset of the objects and events in the environment are relevant to the learning task.

With regard to the "when" problem, our architecture offers several possibilities: learning might occur (1) when a violation of expectations occurs, signaling that properties of the environment cannot be accounted for by existing knowledge structures, hence new structures should be added; (2) when a state is observed in the environment that is related to a high-priority goal of the organism (e.g., another organism obtaining food), in which case the organism may wish to record how that state was reached; or (3) when a series of events involving attended objects is observed.

With regard to the "which" problem, the Focal Attention Mechanism and capacity limitations of WM offer a partial solution because they filter out much irrelevant information from the environment. The learning mechanism must still determine which states and events in WM are relevant, but this is a much reduced problem. It may appear that we have simply moved part of the filtering process to the Focal Attention Mechanism and Working Memory, but such a process is a general part of the architecture, necessary in many information processing tasks.

## A UNIFORM PROCEDURAL REPRESENTATION

In the remainder of the paper, we focus on the role of event schemas in our architecture. Event schemas lie at the heart of the architecture because the procedural knowledge embodied in these schemas is useful to nearly all components of the architecture. Furthermore, event schemas

provide a uniform representation of knowledge across components, which is essential for enabling efficient communication among the components and for facilitating interactions among various learning mechanisms.

## EVENT REPRESENTATION

We represent an event by a triple composed of an *action*, a set of states that are true before the event takes place (the *before-states*) and a set of states that are true after the event takes place (the *after-states*). The action is a description of the event, consisting of an event name, the times delineating the event, object(s) involved in the event, and any other parameters not captured by the state changes. The before- and after-states capture only states that change during the course of the event and whose change can be attributed to the event.

## EVENT-SCHEMA REPRESENTATION

Our representation of an event schema is described in [263]. To summarize briefly, the representation can be divided into six sections. The *schema-vars* section specifies internal state variables of the schema. The *update* section contains code for updating the schema-vars. The *init-states* section specifies a set of states that must be true for the schema to be applicable, in essence, a context in which the event occurs. The *goal-states* section specifies a set of states that must be true for the successful completion of the event. The *script* section specifies a sequence of lower-level, more basic events (hereafter, *sub-events*) that compose the event. Finally, the *restrictions* section specifies either numeric or propositional restrictions on parameters of the sub-events, often in terms of the schema-vars. The script provides only a partial ordering on sub-events; precise temporal relations are specified via the restrictions section.

The script and restrictions sections suggest a conceptual partition of knowledge into structural and featural components. To see this, consider the *Rectilinear-motion* schema, which encodes knowledge of motion in a straight line. The script indicates that rectilinear motion of an object is composed of a sequence of location changes (each a primitive event), terminated when the object stops or collides with another object. The restrictions on the sub-events indicate that with each location change, the new location of the object must lie along the straight-line path formed by the previous locations.

## EVENT SCHEMAS AND LEARNING

The partition of an event schema into script and restrictions sections suggests two distinct Schema-Learning Mechanisms. The mechanism

involved with script learning must pay attention to the order in which events occur and the boundaries of an event; the mechanism involved with the learning of parameter restrictions, however, overlaps highly with the learning of object schemas. In fact, the parameters of an event provide a basis for grouping objects according to their functional role in an event. For example, suppose that org1 repeatedly observes org2 moving towards objects and then consuming them. After an initial observation, org1 may construct an *eat* schema, consisting primarily of a script of basic actions. To comprehend subsequent instances of org2's behavior in terms of the *eat* schema, org1 must note that every object eaten fills the same parameter slot of an event. A generalization mechanism that attempts to find the restriction on this parameter would also discover the concept of food.

### PREDICTIVE NATURE OF EVENT SCHEMAS

In our implementation, event schemas are predictive. That is, an event schema can generate an exact prediction as to what sub-event should occur next, or the relative likelihood of a small number of sub-events. The prediction is specific to the point of indicating not only the general type of sub-event, but also the expected time of occurrence, objects involved, and resulting state changes. This expectation-based encoding allows event schemas to be used by many components of the architecture. We have found application for event schemas in (1) perceiving higher-order events, where the schema predictions are matched to observed events; (2) signaling novelty or unexpected events, as when a predicted event fails to occur, or when an observed event cannot be assimilated into any active schema; (3) planning actions, where the schema predictions are considered as actions to be executed; (4) monitoring the execution of a plan, which can be achieved by invoking a schema simultaneously in perception and in planning, and using the perceptual schema to check that the planner schema produces the desired effect; and (5) mental simulation, where the schema predictions are used to determine the effectiveness of a proposed plan.

### FUTURE DIRECTIONS

We have attempted to convey a general flavor of our architecture, along with several examples of how the architecture, though fairly general, does help to define the nature of the learning task. The most valuable function of the architecture, however, is that it specifies a set of components that can be assumed by researchers interested in learning issues. These components have proven sufficiently useful that we are currently implementing and elaborating the architecture. Once individual

learning mechanisms are understood in terms of this framework, we can attempt to integrate the mechanisms into a complete, intelligent organism.

## **ACKNOWLEDGMENTS**

The ideas presented in this paper developed from discussions by the World Modelers Group at C-MU, in particular Jaime Carbonell, Keith Barnett, Greg Hood, Hans Tallis, and Alain Rappaport. This work was supported in part by ONR grants N00014-79-C-0661 and N0014-82-C-50767, DARPA contract number F33615-84-K-1520, and a grant from the System Development Foundation.

# **KNOWLEDGE EXTRACTION THROUGH LEARNING FROM EXAMPLES**

**Igor Mozetic**

Department of Computer Science<sup>1</sup>  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801

## **ABSTRACT**

We present a method for a knowledge-base compression, restricted to rules of specific form, which uses a learning from examples facility. A large medical knowledge-base, which was a source of more than 5000 training examples, was compressed to 3% of its original size. Most of the extracted rules turned out to be quite meaningful from the medical point of view.

## **INTRODUCTION**

The *NEWGEM* inductive learning program is a descendant of the *AQ11* [230] and *GEM* [288] programs. The program automatically generates decision rules by performing inductive inference on training examples. Training examples are expressed as conjunctions of attribute values, and induced decision rules are logical expressions in disjunctive normal form. The program performs a kind of heuristic search through the space of all possible logical expressions, until it finds a decision rule that satisfies both the *completeness* and *consistency* conditions [229]. Apart from the features that were available in the previous versions of the program, *NEWGEM* provides the following additional features:

- the user may specify the *type* of decision rules, which may be most *general*, most *specific*, or *minimal* (i.e., involving the minimum number of terms),
- the program can handle *ambiguous examples*, i.e., examples that are members of more than one decision class,

---

<sup>1</sup>on leave from: Jozef Stefan Institute, Ljubljana, Yugoslavia

- the *incremental learning* facility was revised and improved,
- a *Prolog interface* was built which enables the use of the inductive learning facility in a Prolog environment.

The primary motivation for this research was a problem of an *intractable* medical knowledge-base. The domain of application is the ECG (electrocardiographic) diagnosis of heart disorders. Because of the combinatorial nature of the problem, the causal model of the heart was developed that simulates its electrical activity. The model was used for automatic generation of the knowledge-base that relates all physiologically possible multiple heart disorders to their corresponding ECG descriptions [38]. However, the ECG knowledge-base requires too much computer memory to be useful for practical applications (5 megabytes of storage). Therefore several attempts were made to find its sufficient subsets and to apply inductive learning algorithms for their compression [38, 264].

The problems that were encountered during these efforts motivated the development of the improved *NEWGEM* program. We tried to develop a useful knowledge acquisition tool and test it on a large problem, without much guidance from the user. The goal was to compress a subset of the ECG knowledge-base consisting of 943 rules with 5240 ECG descriptions. The resulting compressed knowledge-base consists of 88 rules and occupies only 3% of the original storage. The extracted rules are of two types:

- *descriptions of simple heart disorders*, very similar to definitions one can find in the medical literature, and
- *diagnostic rules*, indicating possible causes of particular ECG features.

In the following section we give an outline of the general method for a knowledge-base transformation, show an application of an algorithm for learning from examples for its compression, and outline a relation between the original and extracted rules.

## **KNOWLEDGE-BASE COMPRESSION**

We consider a knowledge-base with rules of the following form:

$$\text{Ante} \Rightarrow \text{Cons}_1 \vee \text{Cons}_2 \vee \dots \vee \text{Cons}_m$$

There are two languages, i.e., sets of attributes with corresponding values, one for an object on the left hand-side, and another for objects on the

right hand-side of any rule. Each object is defined by a conjunction of attribute values. In the following example from the ECG knowledge-base, the left hand-side represents a multiple heart disorder (sinus tachycardia with atrial ectopic beats and wenckebach block), and the right hand-side two symbolic descriptions of ECG curves that may result from the disorder:

[SA_node=st] &	[Rhythm=irregular] &	[Rhythm=irregular] &
[Atr_focus=aeb] &	[P_wave=normal] &	[P_wave=normal] &
[AV_cond=wen] &	[P_rate=100_250] &	[P_rate=100_250] &
[Junction=none] &	[P_QRS=after_P_no_QRS] & $\vee$	[P_QRS=after_P_no_QRS] &
[BB_cond=none] &	[PR_interval=prolonged] &	[PR_interval=prolonged] &
[Reg_vent=none] &	[QRS_complex=normal] &	[QRS_complex=normal] &
[Ect_vent=none]	[QRS_rate=100_250] &	[QRS_rate=60_100] &
	[Ectopic_P=abnormal] &	[Ectopic_P=abnormal] &
	[Ectopic_PR=prolonged] &	[Ectopic_PR=prolonged] &
	[Ectopic_QRS=normal]	[Ectopic_QRS=normal]

We refer to the attributes defining objects on the left hand-side of rules as *decision variables*, and to their values as *decision classes*. The transformation is done for each decision variable separately. First, the remaining decision variables are deleted from all rules. Then, for each decision class of the variable, all corresponding right hand-sides are disjunctively associated. This results in only a few rules, one for each decision class:

$$\begin{aligned} [\text{Var}_i=\text{Val}_1] \Rightarrow & \text{Cons}_1 \vee \text{Cons}_2 \vee \dots \\ [\text{Var}_i=\text{Val}_2] \Rightarrow & \text{Cons}_1 \vee \text{Cons}_3 \vee \dots \\ & \dots \\ [\text{Var}_i=\text{Val}_n] \Rightarrow & \text{Cons}_3 \vee \text{Cons}_4 \vee \dots \end{aligned}$$

Rules of this form are now appropriate for compression through learning from examples. For each decision class, objects on its right hand-side are considered positive examples and all other objects are negative examples. The learning algorithm will eventually find a simple logical expression for each class, involving relations between objects' attributes, that is satisfied by all positive examples and by no negative ones. The whole process must be repeated for each decision variable. The number of rules in the compressed knowledge-base is so reduced to the total number of values of all decision variables.

The compressed knowledge-base is not logically equivalent to the original one. However, the conclusions that can be derived from the extracted rules are equivalent to those derivable from the original rules (see [264] for details), if one can:

- recognize objects not occurring in the original knowledge-base,
- for any consequence "Cons" that occurs in the right hand-side of several original rules, find all antecedents which imply it, and construct all possible objects combining their constituent attribute values. If there exists an antecedent "Ante" between the remaining rules, equivalent to any constructed object, the following exception must be added to the extracted rules:  
Ante  $\Rightarrow$   $\neg$ Cons.

## RESULTS

The method described above was applied to the ECG knowledge-base. An example of two extracted rules (of the minimal type) follows:

```
[SA_node==st]  $\Rightarrow$  [P_wave==normal] & [P_rate=100_250]
[SA_node==sb]  $\Rightarrow$  [Rhythm=regular] & [P_wave==normal] & [P_rate=under_80]  $\vee$ 
[P_rate=under_80] & [P_QRS=after_P_no_QRS] & [PR_interval=prolonged]
```

It turned out that these rules are very meaningful from the medical point of view. Each rule represents a description of a simple *heart disorder*, whether it is combined with other disorders or not. Typically, the extracted rules are more specific than descriptions in the medical literature, which omits many details essential for computer diagnosis.

These rules would be sufficient for diagnosis if we added exceptional rules mentioned in the previous section, to compensate the loss of information. Instead, we took advantage of the completeness of the ECG knowledge-base. The knowledge-base contains all possible ECG descriptions for all possible heart disorders, which allows for its inversion [264]. In inverted rules, each ECG description implies a disjunction of all heart disorders that can cause it. Compression of the inverted knowledge-base resulted in *diagnostic rules*, indicating possible causes of each single ECG feature:

```
[P_QRS=after_P_no_QRS]  $\Rightarrow$  /Atr_focus=wp,at,mat,acb,none/ & [AV_cond=wcn,mob2]
& /Junction=jeb,none/ & /Reg_vent=none/  $\vee$ 
/SA_node=none/ & [Atr_focus=afl,afl] & [AV_cond=none]
& /Junction=jeb,none/ & /Reg_vent=none/
```

In this example, terms in **bold** represent the minimal description. In this

case this is also the most general description. With the terms in *italics* the most specific description results, which improves computational efficiency of the diagnostic program.

The following table summarizes the complexity of the extracted rules, resulting from compression of the original and inverted ECG knowledge-base:

number of:	rules for heart disorders	diagnostic rules
decision variables (runs)	7	10
total examples per run	5240	5240
attributes/values	10/5.2	7/5.1
extracted rules	36	52
conjunctions per rule	2.8	1.7
attributes per conjunction	3.5	2.2
values per attribute	1.9	2.4

Pascal implementation of the *NEWGEM* program running on the SUN workstation spent one week (sic!) of CPU time for all 17 runs together. In further research, we plan to investigate the predictive power of the extracted rules and compare them with descriptions found in the medical literature.

## ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under Grant No. DCR 84-06801, the Office of Naval Research under Grant No. N00014-82-K-0186, and by the Slovene Research Council.

# **LEARNING CONCEPTS WITH A PROTOTYPE-BASED MODEL FOR CONCEPT REPRESENTATION**

**Donna J. Nagel**

Rutgers University, Department of Computer Science  
New Brunswick NJ 08903

## **ABSTRACT**

In this research we are developing a prototype-based model for concept representation that requires the building of transformations for matching instances of a concept. This model allows us to learn concepts that could not be readily defined in previous learning systems that use logical definitions requiring necessary and sufficient conditions for specifying concept membership.

## **INTRODUCTION**

The use of prototype-based models for concept representation has been well motivated in the literature on Psychology and Philosophy [130, 75]. The goal of this research is to demonstrate the importance and the capabilities of a prototype-based model for concept representation in a machine learning scenario. There may be more than one prototype in our representation, and we have adopted an "exemplar view" where a concept is represented by a collection of separate descriptions of some of its exemplars (either instances or subsets) [351]. Variations of this model were designed in the TAXMAN project for defining legal concepts which tend to be amorphous and open-textured [224, 358, 265]. We have also experimented with furniture concepts trying to capture the elements of "chairness" in a supervised training mode.

## **RESEARCH QUESTIONS**

Some general mechanisms are summarized here for addressing the following questions that are central to the development of a prototype-based model.

- 1. What bases do we have for describing and uncovering similarity between*

*instances?*

Typically in AI, concept membership is evaluated by matching an instance to a single logical description called the concept description. However, we argue that instances are members of the same concept if they match any prototypical instance of the concept by undergoing a series of transformations. One important way to match instances is by using transformations as operators for translating the language of their descriptions (e.g., from perceptual features to functional features). Another important way is by decomposing the instance descriptions and matching the parts both within each instance and between the two instances. For example, it might be possible to readily match lower level objects like legs but not higher level objects like chairs that are composed of legs. Each language term can have connected with it a prototype-based concept description. Thus, the parts of the instances can also be evaluated for membership in the same subconcepts.

## *2. How can the instances of a concept be represented as a coherent whole?*

The simplest form of a concept representation is an enumeration of its instances. Another form is an intersection of the instances using a generalization language. In the first form the concept is too specific and does not demonstrate any coherence or allow for inductive inference. The latter form relies on having a good generalization language for coherence, and the concept could become overly general. We can improve on either of these forms by combining them into a disjunctive concept representation. Here the disjuncts (i.e., clusters of instances) are formed using a transformation language that has several different bases for describing the similarity of the instances. A disjunct includes each of the matching instances plus each transformation that links the instances. Each disjunct has a representative instance called the prototype. So the concept is organized, but not overly constrained. The concept can also be easily changed and incrementally updated. Since transformations are based on what instances have already been seen, the history of the concept is reflected in its structure.

## *3. How can typical and borderline instances of a concept be represented?*

A main advantage of a prototype-based model is that certain instances can be distinguished as more or as less representative of the concept than other instances. Instances with mutually exclusive features or

exceptional features can be assimilated by matching or building different prototypes of the concept. The use of transformational units and the different ways of classifying these units, allow us to establish a rough distance metric between instances. Furthermore, since a prototype-based model can represent the notion of typicality, the trainer can take advantage of this by presenting more typical instances to a student early in the training sequence.

*4. How can the student's learning process and available resources be organized and controlled?*

Many parts of the learning process noted here can be characterized as search problems: the search for a set of features of the instance to focus on for matching, the search for an appropriate prototype to match, and the search for an appropriate transformation to make a match. Briefly, some sources for controlling the searches are: focusing help from the trainer, classification of and multiple indexing of transformations which reflect a strong domain theory, standard search techniques (e.g., means-ends analysis and discrimination nets), heuristic preferences for general styles of matching, and guidelines for relatively evaluating the appropriateness of transformations.

## EXPERIMENTS

Over the past few years we have developed a formalism for specifying transformations between instances described in a relational network. The transformations have a uniform representation that allows them to be recursively called in the body of other transformations. This work is described in [265] and more fully described in the forthcoming dissertation. We have implemented this formalism in LISP to experiment with transformational matching for building legal concepts from the TAXMAN research and for building furniture concepts. The learning task is to build a prototype-based concept description given: primitive transformations; mapping procedures for applying transformations and building macro-transformations; heuristics for using the mapping procedures; and positive and negative training instances organized by a benevolent trainer. The student's domain theory is portrayed in the transformations, and the mapping procedures and heuristics are domain-independent.

A concept is constructed in four stages where the training instances progressively become more borderline and the student becomes more independent from the trainer. For example, in the chair domain the

student sees a basic wooden chair, a child-size chair, a one-legged plastic chair, a beanbag chair, and others. Negative instances such as tables and steps are also presented to highlight the differences between concepts so the transformations can be refined.

One way that chairs are assimilated into a concept is by appealing to transformations which map their perceptual features (e.g., type of material) to functional features (e.g., support function) that can be matched. Other transformations map specific sizes of objects to more abstract descriptions such as human size. This is done in an analogical framework where the trainer can direct the student to apply certain types of transformations toward a certain goal [58, 400, 164]. In the legal domain the facts of a case are mapped to the expected risk and return of a stock ownership, and the cases are compared along this dimension. The transformations provide an "explanation" of the important features for assimilating the instances. New transformations that are composed are recorded as part of the concept description and can be reused for assimilating new instances. This analytic approach to generalization is a major focus in current machine learning research [253], [DeJong (this volume)].

In this research we propose an important new way of thinking about concept representation and the beginning of how to use this representation in machine learning. The experiments have been example-driven but have led to the development of a general set of mapping procedures (e.g., procedures for applying sets of transformations based on problem reduction or problem translation) and annotations for cataloguing transformations.

## ACKNOWLEDGMENTS

This work has been influenced by many helpful discussions with N. S. Sridharan, Thorne McCarty, Tom Mitchell, Saul Amarel, Rich Keller, Paul Utgoff, and Smadar Kedar-Cabelli. The research has been supported by a Rutgers University Fellowship and by Grant No. MCS-82-03591 from the National Science Foundation, Washington, D. C.

# RECENT PROGRESS ON THE MATHEMATICIAN'S APPRENTICE PROJECT

Paul O'Rorke

Coordinated Science Laboratory  
University of Illinois at Urbana-Champaign  
1101 W. Springfield Ave., Urbana, IL 61801-2987

## ABSTRACT

This paper briefly describes the "Mathematician's Apprentice" project. The project is a case study of explanation-based schema acquisition in the context of an experimental automated apprentice (called MA). The focus of the project is on the development, demonstration and formalization of explanation-based generalization methods.

## INTRODUCTION

Many knowledge-based AI systems have used schemata (knowledge packets such as frames or scripts) as the basis of computational models of understanding , planning or other problem solving, but very few of these systems have been capable of generating their own schemata. As a result, most schema-based systems have been unable to automatically profit from their experiences, so that the main way of improving their performance has been by laboriously hand-coding new schemata.

At the University of Illinois, a group led by Prof. Gerald DeJong has been exploring and automating a solution to this particular knowledge-acquisition bottleneck. The solution is a brand of explanation-based learning called **explanation-based schema acquisition (ESA)** [DeJong (this volume)]. **ESA** is a kind of learning by observation of problem solving behavior. The key idea is that examples of novel successful and unsuccessful behaviors can best be generalized by explaining the reasons underlying their success or failure.

## RECENT PROGRESS

One of the research projects underway at Illinois is the "Mathematician's Apprentice" project [271]. The major goal of this project is to develop and demonstrate powerful and robust **ESA** generalization

methods- methods which can enable automated apprentices to learn as much as possible from many different examples of masterful problem-solving.

An advantage of doing **ESA** research in formal mathematical domains is that experiments involving many different examples are relatively easy to execute. This is because **ESA** is a knowledge-based learning method, in other words an **ESA** system already has to know a lot before it can learn anything new. In informal domains the task of extending a system to enable it to learn from new examples is often difficult because so much effort must be expended representing additional initial knowledge. It is easier to provide the necessary basic knowledge in mathematical domains. Large scale experiments are more valuable than experiments involving a few isolated examples because they provide a better feeling for the robustness of the learning method being tested.

An experimental automated apprentice (a computer program called MA) has been implemented. The mathematical logic of natural deduction was chosen as MA's initial domain partly because simple logical reasoning seems like a prerequisite to most other mathematical reasoning. Another advantage of natural deduction as an initial domain is that one can easily run an experiment involving a large number of examples after providing only a small collection of simple prototype schemata for the axioms and inference rules of the logic.

Like any other apprentice, MA starts life with very limited problem solving ability. Initially, MA can only make tiny contributions to most problem solving efforts; its master must supply the insights which lead to successful proofs. At first, MA "merely" observes the master's behavior, but MA recognizes when this behavior leads to success. Then by generalization based on analysis of the reasons for success MA improves its own future performance: learning new schemata and rules for their use.

MA's initial generalizer worked on some natural deduction examples but failed to work on others. The generalizer worked by stripping away all but the essential facts of the specific examples. The problem was that stripping away the irrelevant details often left ambiguities which depended on the details for their resolution!

Two ways to get sensible results out of generalization in the face of such ambiguities have been developed and implemented. The first *avoids* the ambiguities: interpreting them just as they were interpreted in the observed example. The second method *analyzes* the ambiguities: computing generalizations for all consistent interpretations of the ambiguities.

The problem of ambiguities (or disjunctions or multiple compositions) has been encountered by several researchers working independently on explanation-based learning (including Smadar Kedar-Cabelli and Sridhar

Mahadevan of Rutgers and Steven Minton of CMU). The current consensus seems to be that one should compute unambiguous generalizations by interpreting the ambiguities just as they were interpreted in the observed example. Analyzing the ambiguities is a more powerful generalization method because it may learn more from an example: enabling a problem solver to solve more problems. Unfortunately, most domains have so many ambiguities that computing all possible interpretations is not feasible.

Experiments with the more powerful generalizer have shown that little may be lost by ignoring the alternative interpretations. In the natural deduction domain at least, they tend to be much less useful than the observed interpretation.

Experimental progress in the domain of natural deduction has been complemented recently by work on formalizing generalization methods. Several advantages result when the generalization method is formalized as well as the domain. One advantage is that clear descriptions of the design and desired behavior of the generalizer make it possible to prove that the design behaves as it should. To construct such a proof, one must make explicit the assumptions underlying correct behavior of the generalizer. Once these assumptions are known, it becomes possible to argue convincingly that a given generalization method will work on arbitrary examples within a domain. Such an argument should be a much more convincing testament to the power of a generalization method than any demonstration involving a limited number of examples from a domain.

## FUTURE WORK

We plan to continue the natural deduction experiment by leading the learning program through a series of theorem proving exercises from a standard textbook. The classes of problems the system learns to solve will be noted and the incremental improvements in performance will be tested on increasingly difficult problems from the same text. An experiment of this kind, involving more than a few isolated examples, gives some feeling for the power or weakness of a learning method. This experiment may demonstrate that **ESA** alone is capable of taking an automated apprentice from novice to expert performance levels.

On the theoretical side, we plan to continue formalization of MA's generalization methods. The main value of formalization is that it is a way of forcing all the assumptions underlying the success of a learning method to be made explicit. This provides an additional starting point for efforts to extend the method and also enables more convincing

demonstrations of the power of the learning method. In addition to demonstrating that MA works well on *many* natural deduction examples, we will try to prove that the method of generalization does what it should on *any* natural deduction example.

## RELATION TO OTHER WORK

Closely related research on explanation-based learning for apprentices is in progress at many other research centers including Carnegie-Mellon [Minton (this volume)], Columbia [Ellman (this volume)], Rutgers [Mitchell (this volume)], Schlumberger [Winston (this volume)] and Stanford [Wilkins (this volume)].

## ACKNOWLEDGMENTS

Special thanks to my advisor Gerald DeJong and fellow students Ray Mooney and Shankar Rajamoney for many interesting discussions. Group meetings including the other members of the CSL machine learning group have also contributed to this research. Discussions over the ARPA/CSNETs and at the last two Machine Learning Workshops have also been useful. This research is supported by the National Science Foundation under grant IST-83-17889.

# **ACQUIRING DOMAIN KNOWLEDGE FROM FRAGMENTS OF ADVICE**

**Bruce W. Porter, Ray Bareiss, and Adam Farquhar**

Computer Sciences Department, University of Texas at Austin, 78712

## **INTRODUCTION**

This research examines the acquisition of domain knowledge from an expert. Currently, the transfer of expert knowledge to a computer program is a painstaking process. The human expert is asked for a set of problem solving heuristics. Each heuristic is a fragment of domain knowledge which the expert has packaged with other heuristics to produce well orchestrated problem solving behavior. Twenty years of experience by AI researchers handcrafting expert systems demonstrates the difficulty of acquiring complex knowledge.

Part of the reason that knowledge transfer is difficult is that the explication process is unfocused. Experts are asked to "*tell* all they know" which is considerably less natural than *demonstrating* their problem solving skill. Presenting experts with specific problems to solve focuses their attention and structures their explanations. However, the resulting advice tends to be fragmentary. Focused, task-oriented teaching is quite effective for the teacher, but the student gains the arduous responsibility of integrating the fragments of advice into a coherent knowledge base. The first part of this research examines the learning requirements for supporting knowledge integration.

The second part of this research examines the use of acquired knowledge to guide both further learning and expert reasoning. Since the initial knowledge base is constructed from specific examples, further learning is needed to convert it into a more usable form. Characterizations of the knowledge are formed by generalizing the examples and asking the expert questions. These characterizations support expert reasoning. The reasoning process is driven by knowledge-based pattern matching of new data to learned characterizations. The epistemological motivation for this research is described in [280].

We have been working in the domain of clinical audiology with the assistance of Dr. Craig Wier of the Speech and Hearing Department of the University of Texas at Austin. We are designing a learning program, called PROTOS, which acquires and uses diagnostic expertise in audiology given instruction similar to that given to human students studying audiology

under Dr. Wier.

## LEARNING DOMAIN KNOWLEDGE

PROTOS learns from training similar to that given to human students. The domain expert presents annotated case histories. Each case history contains descriptions of states in the diagnostic process (e.g. a particular disease or class of diseases) and the procedures for moving between states (e.g. conducting a particular sequence of tests).

Each state is a 3-tuple <feature list, classification, justification>. The presence of the *justification* field distinguishes this training from that previously used in concept formation research [93]. The justification explains the relationship between each feature, *f*, in the feature list and the assigned classification, *C*. For example, *f* may be a *criterial*, *function enabling*, or *empirically associated* feature of *C*. Indirect connections are more complex relationships. For example, given a feature *g* and its relationship with *C* from previous training, a valid justification for *f* is that *f causes g*. Similarly *f* might be a *specialization* of *g*. From our initial analyses of diagnostic protocols, we have identified twelve justification types. We will add more types as they prove necessary.

PROTOS checks that the justification of each feature given by the teacher makes explicit a relationship between the feature and the classification. If the relationship is not adequately established by the justification then PROTOS asks for further substantiation of the relevance of the feature. Each justification is a schema which the teacher instantiates. One slot common to all justification types is a measure of relative confidence in the relationship (which might itself be subject to justification).

Procedures for moving between states are also made explicit by the teacher. Each procedure is a 4-tuple <preconditions, postconditions, expectations, alternatives>. The *preconditions* and *postconditions* are necessary constraints on the procedure. The *preconditions* of the procedure must be satisfied before the expert would recommend applying the procedure. Similarly, the expert expects the *postconditions* to be satisfied after applying the procedure. The *expectations* are the reasons for performing the procedure (e.g. the data to be collected by a particular test). The *alternatives* list other procedures with similar preconditions, postconditions and expectations.

Three forms of knowledge are constructed from the training provided to PROTOS. First, the features and justifications of each training instance are merged into a network. Each justification is a fragment of domain

knowledge expressing relationships among the features and classes. The primitive justification types are the permitted arc labels in the network. Second, a *state cluster* is formed for each of the classifications in the training set. Each cluster consists of *points* described with the feature list descriptions from the training set. Third, a lattice of interconnections among the state clusters is formed from the procedures used in diagnosis. All three forms of knowledge grow incrementally as more case histories are presented by the teacher.

We anticipate that the domain knowledge learned by PROTOs will be relatively "deep." Michalski and Chilausky [234] demonstrated the feasibility of learning expert level classification rules from examples. However, the resulting knowledge was too superficial to either generate explanations of its conclusions or overcome the brittleness of rule-based systems. One strength of the PROTOs design is the disambiguation of the inference rule used in expert systems. The IF-THEN form of expert system rules conceals the variety of relationships which are made explicit by the justification field of each training instance.

Furthermore, we anticipate that PROTOs will be relatively easy for the teacher to use. PROTOs focuses the teacher's attention on the justification of individual features in a training instance. Yet the justification language permits the expression of complex relationships and interdependencies among the features and classifications.

## USING AND REORGANIZING LEARNED KNOWLEDGE

The domain knowledge learned by PROTOs is used for expert reasoning. When a new case is presented to PROTOs for diagnosis, it is matched against known cases and diagnoses. The learned network of domain knowledge drives a complex pattern matcher to find relevant past cases to consider. This may require a reorganization of the learned state clusters. The points in each cluster are defined by the specific feature lists of the training instances. PROTOs generalizes the points in each state cluster to form a prototypical description which is used for diagnostic reasoning.

Prototypes are formed which characterize the points in each state cluster. This is done by successive application of generalization operations to the points. Each generalization operation merges a pair of points in the cluster yielding a new point which characterizes the original pair. A generalization operation is enabled if there is sufficient knowledge in the learned network to establish a connection between the points being merged. In effect, the domain-specific network guides the pattern matching of

examples.

The degree of reorganization of the learned clusters is determined by the amount of knowledge in the network. Two points in a cluster will not merge if their features cannot be matched by traversing legal paths in the network. A "tentative merge" is performed if the match is exceedingly indirect or the links traversed have low confidence. In this case the original points are retained and the resulting generalized point is flagged as suspicious. If a match is almost established, PROTOs attempts to obtain substantiating information from the expert. PROTOs might ask for the relationship, if any, between two unmatched features or ask for the results of a data gathering procedure. As always, interaction with the expert is task-oriented and focused.

The search for prototypical characterizations is complex due to the knowledge-based pattern matching and the number of points in each cluster. PROTOs addresses this problem with a distributed agenda. Each point in a cluster starts with a nominal "strength." A characterization of two points inherits strength from each point resulting in a new point with greater strength. The strength of a point determines the degree of "attraction" with other points. PROTOs attempts to pattern match (and merge) points which have high mutual attraction due to strength and estimated proximity. The degree of attraction determines the level of effort expended on pattern matching. In effect, each point in a cluster influences neighboring points and control decisions are distributed and parallel.

The same knowledge-based pattern matching mechanism is also used for expert reasoning. After some training, PROTOs is presented a new case for diagnosis. Each cluster of points attempts to attract the new case by coercing its interpretation to match the classification of the cluster. As in prototype formation, the degree of effort expended is a function of the strength of the cluster (which is a function of the strengths of the individual points in the cluster). Therefore, prototypes serve as models which guide the interpretation of new cases subject to the domain knowledge in the learned network.

## CONCLUSIONS

This research examines the acquisition of domain knowledge for diagnosis from training similar to that given to human students. The two main issues in this research are the integration of knowledge "fragments" into a cohesive network and the use of knowledge-based pattern matching to reorganize and apply the learned knowledge. We are in the process of building PROTOs which demonstrates these ideas in the domain of

diagnostic audiology.

## **ACKNOWLEDGMENTS**

Support for this research was provided by the Army Research Office, under grant number ARO DAAG29-84-K-0060.

# CALM: CONTESTATION FOR ARGUMENTATIVE LEARNING MACHINE

J. Quinqueton<sup>1</sup> and J. Sallantin<sup>2</sup>

## ABSTRACT

We present in this paper the main features of a Learning Engine, CALM, based upon a multi-agent learning technique and an artificial argumentation system.

## INTRODUCTION

In Machine Learning, there exist two approaches, which are generally mixed in the various learning methods:

- Learning Engines
- Learning Strategies

We define a **Learning Engine** as a problem-independent organization of macro-operators of Learning.

A **Learning Strategy** is a goal-directed organization of Learning mechanisms. Such a goal can be to explain [Mitchell (this volume)], to solve [24], to adapt [Langley, Rappaport, Michalski (this volume)], to control [Carbonell (this volume)] or to discover [24].

CALM is a Learning Engine which is

- Multi-agents
- accepting contradiction in the examples
- working in a discrete space of modalities (modes)

The Logical foundation of CALM is in non-distributive logic (like quantum logic or majority logic)

---

<sup>1</sup>INRIA - BP 105 - 78153 LECHESNAY CEDEX (France)

<sup>2</sup>CRIM - 860 route de St Priest - 34100 MONTPELLIER (France)

Strategies using CALM have been developed in various domains

- Genetic sequences Analysis [24]
- Speech Recognition [277]
- Earthquake Prone Areas [284]
- Ethological process [285]

## PRESENTATION OF CALM

In our system, a learning problem is made of 4 elements: a description language, a learning set, a system of links between objects and concepts, and a system of artificial argumentation. The first 3 makes what can be called the "Learning Engine".

### The Description Language

The objects of the problem are supposed to be described in a given structural language, say **L**.

**L** can be simple binary description of the objects (to fit or not a given property) as well as a higher level description, such as first order logic predicates (statement of relations between parts of the object).

### The Learning Set

Classically, a learning set **X** is a set of objects which is supposed to be a representative sample of the concept to be learnt. So it is in CALM, and we can also have a family of sets of examples, to learn a family of concepts. This feature is useful in some cases, mainly in prediction problems [284].

### The Objects/Concepts Links

To state and describe the relationship between objects and concepts, we define in CALM a set **M** of logical modes which are used as argumentative links. These modes can be classified in three groups, according to their semantic interpretation:

- two modes of Certainty (True, False)

- two modes of **Plausibility** (Justification, Refutation)
- three modes of **Propensity** (Contestation, Rectification, Silence)

Of course, these modes are related to an underlying logical theory  $T$ , stated as a constraint of the problem. Such a theory can be implemented as a parameter of the Learning Engine, and viewed as a **Rational Agent**, i.e. a part, or projection, of the concept to be learnt.

Then, the concept  $X^*$  is an **intensive** expression, according to  $T$ , of the **extensive** expression  $X$ , and the link between them can be stated as:

$$\text{Just} \langle X \ X^* \rangle_T$$

This "Hilbert-space-like" notation points out the duality between objects and concepts. Such links are used in CALM for Induction, Deduction and Control, and their general form is, for a mode  $m$  and an object  $y$  (or a set of objects) of language  $L$ :

$$m \langle y \ X^* \rangle_T, \text{ for } m \in M \text{ and } y \in L.$$

### **Artificial Argumentation**

As CALM appears, through the previous definitions, as "multi-agents" learning system, an Artificial Argumentation (AA) has to be defined, and will depend on:

- $X$  the learning set ( $s$ )
- $T$  the theory (ies) of each rational agent
- $M$  the set of modalities.

The argumentation will have to solve conflicts, i.e., for the interpretation of a given example, to find an application of  $M^n$  on to  $M$ , where  $n$  is the number of rational agents, in order to combine the  $n$  links of the example to the various rational agents.

For instance, a vote is simple argumentative process. As we control the quality of each agent and the object/concept relationship, it is possible to organize the artificial argumentation as the deliberation of a staff and the voting process as a way to obtain a decision, like a "conclave".

## RELATION BETWEEN CALM AND OTHER LEARNING STRATEGIES

The system we described above can be situated regarding other learning strategies through 3 features: explanatory capabilities, static/dynamic learning and data-driven/model-driven aspect. These features point out the essential aspects of the **control** of CALM.

### **Explanatory Capabilities**

As CALM is rather a data-driven process, its result will be an explanation if the agents and concepts are selected for this goal.

For instance, if we attempt to create an agent to justify or refute each element (observation, variable, or predicate) of the description language **L**, then the agents will be rather self-explanatory. They will be organized sets of rules to forecast an element of description from the others.

But the main element is not the content of the agents built by the learning engine, but the Argumentation process, i.e. the **active** way the agents will communicate with each other. From this point of view, the user of the system can be regarded as a particular concept or agent among what was learnt by the system.

In this case, the Artificial Argumentation defined earlier will provide an active **dialogue** with the user. In practice, the argumentation process is anthropomorphic enough to be rewritten as an explanation.

### **Static or Dynamic Learning**

Regarded as a part of an experimental learning machine, CALM is essentially **static**, i.e. it does not have to adapt to the real world [Langley, Rappaport (this volume)].

It is so, because it is organized to learn concepts from examples, and to provide new examples of what it has learnt. But, if there is a conflict between the knowledge of the system and the real world, it will be solved by argumentation and, possibly, by a new learning. But such a new learning is not an intrinsic mechanism for CALM. We are mainly interested in the collective behavior of the agents when they perform an argumentation about the objects.

In molecular biology, for instance [24], our world is a set of sequences in a given code, and the goal is to create new sequences which have a given biological activity.

A dynamic CALM in this case would be to check, with real experiments, the sequence and then to modify the learning process according to the experimental results. But, in Biology, such experimental results need generally 6 months or 1 year to be obtained! Then the dynamic aspect is not interesting, and the learning system is rather used to select good experiments to perform.

To conclude on this aspect, let us notice that the fact that CALM is basically static is an important element to control the generalization, as it will appear in the next paragraph.

### Data or Model Driven Learning

We define a data-driven learning system as a system which builds a concept from examples. Of course, a model-driven system also checks this definition. The difference is in the way the generalization is controlled: in the learning technique itself (data-driven) or by external constraints (model-driven), like exclusion of the counter examples from the generalization.

Of course, a practical learning system is both data and model driven. But there is generally a dominant tendency among these two typical ways of controlling the generalization.

CALM is mainly data-driven. The concept  $X^*$  explains, in **intension**, the set  $X$  of examples which is an expression, in **extension**, of the concept  $X^*$ .

In another way, given  $M$ ,  $X$  and  $T$ , how to determine  $X^*$  such that:

$$\text{JUST} < X \ X^* >_T.$$

The technique used in CALM to build  $X^*$  is a couple of sets of conjunctive normal formula on the description language  $L$ .

Then, an internal control is allowed. A concept is accepted if it checks a **closure test**, i.e. the set of all the possible objects proposed by the concept must be closed regarding the learning.

In other words:  $[x : \text{Just} < x \ X^* >_T]^* = X^*$

A rational agent is accepted if it checks a **low contradiction test** and if it is able of generality.

The learning process is not looking for explanatory rules [Mitchell, Michalski (this volume)] but for building an agent which is **rational**, i.e. which has a **logical control**. This is the reason for using conjunctive formulas rather than disjunctive ones [Michalski (this volume)].

CALM is tolerant to low level noisy data. There are two kinds of noisy data:

- “real” noise, i.e. errors in measurements, for instance
- contradiction between examples and counter examples.

The CALM learning process depends on the number of examples, but not on their order.

The modification of the knowledge base of CALM can be made in two ways:

- changing an agent by addition of new examples
- changing argumentation by addition of an agent.

## RESULT OF CALM

Let us now look at the various possible results of CALM.

Case 1: no agent is created by the examples.  
then: change X, or L, or go to the beach.

Case 2:  
one rational agent is built and is good enough in artificial argumentation.  
then:

1. Goodish! It is an interesting discovery (never seen in practice)
2. Hells! We learnt a trivial thing (no interest)

Case 3: an artificial argumentation is built on a set of several rational agents, which helps the user to understand his problem. It is the most general case, that we met in practice in all the problems submitted to CALM:

- molecular biology [24]
- speech recognition [277]
- Ethology (analysis of animal behavior) [284]

## CONCLUSION

The learning system we presented in this paper has been used with a reasonable success in various domains, for problem-understanding and, sometimes has lead to a discovery (calcium binding proteins in biology).

Then, we work now to enhance this system regarding its logical and control aspects:

- to generalize the use of first order predicates in the descriptions
- to point out feasible tests, or to find theorems, regarding the closure problem that we stated earlier
- to make the argumentation a natural language dialogue between user and system, by implementing a natural language interpreter and generator based upon an ATN (Augmented Transition Network).

Confucius said that "an intelligent man could not be a slave". We can add like S. Watanabe, that "An intelligent machine cannot be a slave", or, at least, not completely . . .

# **DIRECTED EXPERIMENTATION FOR THEORY REVISION AND CONCEPTUAL KNOWLEDGE ACQUISITION**

**Shankar A. Rajamoney**

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign  
Urbana, IL 61801

## **ABSTRACT**

Most current Artificial Intelligence systems require a complete and correct model of their domain of application. However, for any domain of reasonable size, it is not feasible to construct such a model. The main thrust of this project is to build a system that initially starts with an incomplete and incorrect model. The system constantly monitors the real world verifying predictions made by its planner. When the system is confronted by observations from the real world which are inconsistent with the computations performed on the world model, it postulates hypotheses to explain the inconsistency and devises experiments to test the hypotheses. Based on the results of the experiments the system updates its world model to accommodate the previously inconsistent observations.

## **INTRODUCTION**

Acquisition of conceptual knowledge is difficult. When people discovered radioactivity, it took them a long time to formulate the proper concept. This is a case where previous knowledge is not sufficient to explain the observations and thus *reasoning has to be replaced by experimentation*. Another point to observe is that new concepts are not discovered by a *search* for them, but by noticing *discrepancies* between the world model predictions and the way the world behaves. New concepts arise out of the necessity to explain these discrepancies while maintaining a knowledge base consistent with the earlier observations.

## **SYSTEM OVERVIEW**

Our system (also described in [286]) works in the chemistry domain. It has schemata for actions like POURing liquids, MIXing liquids to form solutions, and STORing liquids in containers. It also has schemata for

processes like the FLOW of liquids, EVAPORATION of liquids, CONDENSATION of vapors, ABSORPTION of liquids by solids and RELEASE of the absorbed liquid by the solid.

Initially the system does not know of any process that involves the flow of liquids through solids. In the real world, however, there is a natural phenomenon, osmosis, which takes place when two solutions of different concentrations are separated by a permeable membrane. There is a flow of solvent through the membrane which results in a decrease in the difference in concentrations of the two solutions.

The system is given a goal of forming a mixture of specific amounts of solutions,  $\#\$Solution1$  and  $\#\$Solution2$ , of known concentrations, conc1 and conc2 ( $conc1 > conc2$ ). In order to achieve this goal it generates a subgoal of temporarily storing the specified amounts of the two solutions in containers. As it happens the only suitable container has two compartments separated by a permeable membrane. The planner, not realizing the significance of the partition, plans to pour the two solutions of differing concentrations into the two compartments. As a part of its monitoring of the real world, it expects a number of observations including a decrease in the amount of solutions in the two original containers, and the appearance of specified amounts of solution in the two compartments of the selected container.

The system verifies the predictions made by the planner by comparing the predictions with its input observation stream. Immediately after execution of the plan it finds that all the predictions made by the planner are confirmed. However the next time it examines the container it finds that the amounts of the two solutions have changed. This change was not predicted by the planner.

The explanation module tries to relate the change to an effect of one of the known processes. But the system finds that none of the relevant processes - flow, evaporation, condensation, absorption and release - can be activated as they have preconditions that are not satisfied in the present situation. So the system is left with a contradiction: it has observations for which its current world model has proven inadequate (Figure 1).

The beliefs on which the contradiction rests are obtained from the reasons for the failure of each process to run. They are :

1. The procedure for classifying solids into absorbent and non-absorbent classes is right.
2. Liquids require a clear, solid-free path to flow, and
3. Evaporation and condensation require exposure to the

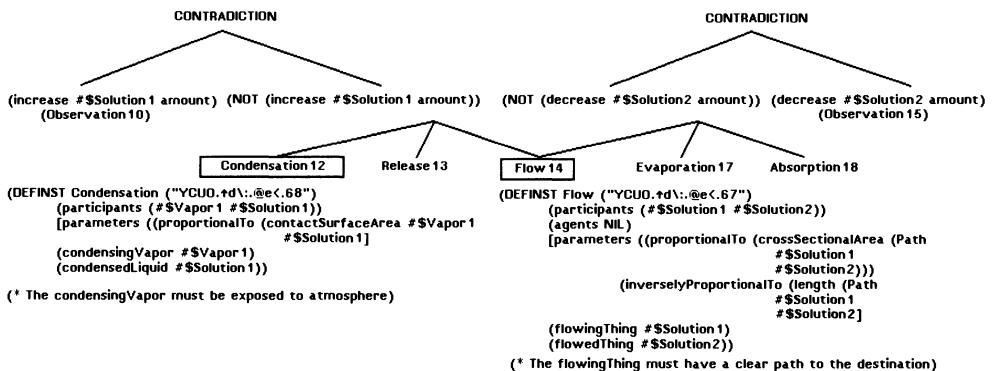


Figure 1. The contradiction diagram, a brief description of two processes and the preconditions that failed.

atmosphere.

The system first checks the validity of the preconditions. It may be that one of the preconditions is wrong and some process is actually taking place. To determine if this is indeed the case the system performs a series of discrimination experiments on all the instances of the relevant processes. It selects a discriminant which may be any characteristic of a process that can distinguish one set of processes from another. It then constructs experiments that bring out the difference in behavior between one set of processes and the rest with respect to the selected discriminant. On examining the results of the experiments it can decide which set of processes could have taken place. It repeats the above sequence choosing a new discriminant.

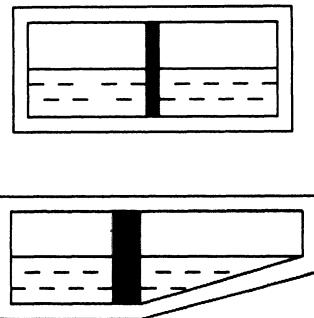
For example, one characteristic of a process is the time rate at which the process progresses. This depends on geometrical parameters (length, area etc.), state variables (temperature, pressure etc.), and the properties of the participants. Experiments are designed such that one of the processes is allowed to dominate the rest by having an environment in which its rate is enhanced and the rates of the other competing processes are inhibited. If the original observations are reproduced in a much shorter time period then the evidence points to the dominating process as the cause of the observation. The flow process rate depends on the cross-sectional area and the length of the path from the source to the destination, the evaporation and condensation rates depend on the surface area of contact between the liquid and its vapor, and the absorption and release rates depend on the surface area of contact between the absorbing solid and the absorbed liquid. By manipulating the geometry of the container it is possible to build a container in which parameters like contact surface area, cross-sectional

area and length are maximized or minimized to allow one process to dominate the other competing ones. The computer generated experiment specifications to distinguish the process evaporation from the competing processes flow and absorption and an experiment that meets these specifications relative to the original setup is shown in Figure 2. The design of the container based on the generated specifications is presently assumed to be input by an oracle.

```

# $EXPERIMENT13
Process being tested
    Evaporation17
Specifications
    ((maximize (contactSurfaceArea # $Vapor2
                # $Solution2))
     (minimize (contactSurfaceArea # $Container3
                # $Solution2)))
    ((maximize (length (path # $Solution2
                # $Solution1)))
     (minimize (crossSectionalArea (path # $Solution2
                # $Solution1)))))
Competing Processes
    (Flow14 Absorption18)

```



**FIGURE 2.** Experiment specifications to distinguish evaporation and one possible container design meeting the specifications.

The system requests that the experiments be carried out and the results returned. Based on the combined results the system concludes that the process flow of the solvent through the membrane caused the original observation.

The system now creates a new process schema that resembles the above flow process. However the new process has preconditions that permit flow when the two specific solutions are separated by the specific partition as in the example above.

## PLANS FOR THE FUTURE

Our current efforts are concentrated on generalizing this specific case using explanation-based learning [90] in conjunction with a further series of experiments aimed at discovering the properties of the participants that played a crucial role in the example. Also we hope to obtain a functional definition for the concept of "permeability".

## **ACKNOWLEDGMENTS**

This work was supported in part by the Air Force Office of Scientific Research under grant F49620-82-K-0009 and in part by the National Science Foundation under grant NSF-IST-83-17889.

# **GOAL-FREE LEARNING BY ANALOGY**

**Alain Rappaport**

Carnegie-Mellon University, Robotics Institute  
Pittsburgh, PA 15213

## **INTRODUCTION**

The design of a learning architecture in an information-rich world (IRW) is essentially that of a system able to filter information. Traditional AI systems from the production system paradigm are knowledge intensive and thus have a built-in bias in restricting information from the outside world. The issue there is to define the most adequate way of processing knowledge, which may be obtained by the original design of heuristic methods or by various learning capacities. Furthermore, those systems are not interactive, nor do they usually evolve in a changing environment.

We focus on the development of a comprehensive theory of learning in an IRW that should closely parallel human cognition and particularly its development. In the following paragraphs we describe some elements concerning the development of learning capacities of a primitive organism in an IRW such as the World Modelers reactive environment [196], [Carbonell (this volume)]. We assume that the organism interacts with the world via a sensory-motor interface, in a reflex-oriented manner prior to understanding higher level concepts. Each organism has physiological functions which impact its behavior (e.g. hunger). The organism's sensory perception is driven by changes in the environment, including those caused by similar organisms. We assume it can identify itself as a member of the class of similar organisms through the realization that they share common properties. As we will also consider implementation issues, the perception of such changes are to be the result of an interpretative, hierarchical event-memory which constructs high-level complex events from the very primitive observations and allow their perception as such [Mozer (this volume)]. The very limited knowledge of these organisms does not allow them to perform tasks such as trying to infer another's goals or subgoals so as to learn useful procedures by imitation. Therefore, analogical learning should occur in the absence of knowledge of the actual goal, in a concept-driven, goal-free exploratory fashion.

## BUILDING A CONCEPTUAL DRIVE

An elementary issue in such a model lies in the definition of an event, and thus in the discontinuous perception of reality. This property emerges from the limited resources of the organism which force the segmentation and the discrete perception of the otherwise continuous world of events. For instance, an infant's attention may be disturbed by a noise which introduces a possible break in the apparent flow of events, or two events may be separated by a period of silence, long enough for the infant to lose its interest in the initial event. This issue, although critical at this point, will be reassessed later in light of our model. Events are to be represented as object descriptions in the form of coherent lists of interpreted symbols, such of as: (*org2 move-to-self smile talk touch\* cold\* wet\**) where the learning organism is watching organism2, and starred elements are direct perceptions. Each atom can be seen as representing a schema of a schema-based event-memory that was activated during the period of observation. Such schemas can be used for recognition as well as for action [Mozer (this volume)]. We have thereby defined a general format for perceived events. The occurrence of a different atom in the list corresponds to a change in the observed reality (atoms are given semantically interpretable names so as to be able to evaluate the symbolic processing).

Primitive organisms with these properties lead us to address learning issues from a conceptual and domain-general approach rather than from a problem-solving one.

A very important type of learning in an IRW is **learning by observation**, so as to extract from the numerous inputs a limited number of abstractions or concepts. Learning by observation may actually be seen as the need to obtain a macroscopic description of the space of observations. We envision that different methods can be applied concurrently, to deal efficiently with the search versus knowledge issue. In the present case, because of the organism's lack of knowledge and goals, we cannot be concerned with an improvement of problem-solving activity except for events related somehow to physiological functions which could benefit from this criterion for a domain-specific analysis. The lack of knowledge thus suggests the use of weak methods, combining formal or semi-formal methods of clustering [76] and heuristic search. Furthermore, a simple description of the world is a static representation which, although it can be referred to for problem-solving, does not allow by itself behavioral, active consequences. Hence, we are actually looking for a very restricted set of representative abstractions (representative schemas) which must reflect a focus of attention at a conscious level, interpretable in behavioral terms and

actions. Our approach requires:

1. The definition of domain-independent, general *operators* to describe the space of events. This part of the analysis is formal. Such operators must be symbolic and describe differences between the events. For example, when considering such events as sets, we compute their *symmetric difference* ( $\Delta$ ). Hence, if  $A$  and  $B$  are two events or subevents, then by definition:

$$\Delta(A,B) = (A \cup B) - (A \cap B)$$

This operation may be used as a feature-matching function to express similarity between defined sets [368]. Furthermore, a set can contain nested sets, thereby leading to different levels of analysis and abstractions.

2. The definition of general search methods and heuristics to build abstractions from the structured space of events.

Applying the symmetric difference to the space of events creates a symbolic metric space which can be partitioned into clusters [287]. The partitions thereby obtained have a specific meaning expressed by their respective indices. Since the number of clusters grows rapidly, certain criteria should be applied to reduce the search space. The following have been implemented and applied to a space of partitions:

- *Specificity criterion:* Select minimal partition-indices with regard to inclusion.
- *Abstraction criterion:* In each partition, select maximal clusters with regard to inclusion.
- *Short term effects:* by introducing a measure of similarity to the last input observed, the clustering analysis is made *context-sensitive*.

The final output is a restricted list of elements from selected events constituting the abstraction. Another type of analysis, context-sensitive as well, is being tested where events are considered as entities interacting with one another by means of this symbolic distance. We are studying a relaxation-like approach, where the processing of inputs induce the activation of a restricted number of events corresponding to the abstraction sought.

These methodologies may be considered as semi-formal, using mathematical properties as well as heuristic operations to perform the search. It has been studied on a knowledge-based system [287] where it acts as an endogenous concept-driven mechanism for skill refinement. Although formal search methods to extract concepts have often been inefficient [76], we suggest that their psychological relevance may not have been investigated from the most adequate perspective. They are useful in the absence of formalized knowledge, particularly early in development. Other concept-generating methods which are knowledge-based [93, 400] can only take place when sufficient task-specific knowledge has been acquired. Hence, there exists a continuum of concept-generating methodologies throughout cognitive development.

## GENERATING PLANS AND ACTIONS

The next step in the expression of a concept-driven behavior is the planning and execution of actions. The mechanism of abstraction described does not take goals into account, apart from natural explicit goals such as hunger and the resulting need to look for food. Except for such explicit goals, the ever-present background goal remains at this stage *to do something* [17]. Indeed, a goal-based approach at this stage would imply the contradictory fact of knowing or of being able to infer goals from some extensive knowledge of the world yet to be acquired. Methods such as means-ends analysis or analogical reasoning [58] are not applicable until sufficient knowledge has been acquired. Hence, behavior is partially goal-directed by internal-drives and mostly environment-driven; it is by essence very stimuli-dependent *and* concept-dependent without yet a clear correlation between the two mechanisms (i.e. knowledge). It is thus exploratory and encompasses a strong degree of risk.

The organism uses its sensory-motor capacities and physiological properties and states, it can use them to generate a behavior from those abstractions. We are designing a rule-based representation of the organism's faculties which should be used to *order* the elements of the abstractions derived from the clustering analysis into an agenda, or plan of action. This production-system module is based on rules of the following general format.

*if* (conditions on the perception of the current scene)  
*then* (set priority-level of action in agenda)

For example:

```

if (and (edible ?x)
      (or (visible ?x) (at-hand ?x))
      (hungry ?org)
      ((no high-priority) ?org))
then (set ?x first-priority))

```

In the making of the agenda, rules should monitor both the external and the internal environments. The result is a different arrangement of the various elements obtained by the clustering analysis, according to their *immediate feasibility*. The plan should then be executed. Interferences from the world can occur, which can come and perturb the course of the action by shifting the focus of attention. Otherwise, resources are mainly allocated to this action.

## GOAL-FREE LEARNING BY ANALOGY

Since the abstractions used to create the behavioral drive are issued from an analysis of observations of other agents, the resulting behavior is likely to resemble an imitation of past actions of the latter. However, there is no knowledge of the original and often complex goal. This situation leads to the possible *a posteriori* identification of a goal during or after completion of the action. For instance, a learning agent might imitate another which climbed on some structure to repair something, and, while the learner now undertakes the action without this explicit goal, it discovers, in the present context and along with its perceptions, that it is a more than adequate place to go and see the surroundings better. Thus, a subjective chunk of knowledge about the reality has been constructed. Such chunks may represent incomplete and premature associations of objects and procedures, but these relations will be later generalized by more task-specific learning methods. As another consequence of these goal-free actions, rules used to establish the agenda could be modified, thereby improving future planning.

The plan built will actually be made of various elements of different nature. When the agent is undertaking a specific action scheduled within this plan, the other items belonging to this plan will be used as a *context* to which, along with state variables (physiological functions), perception of reality during the action is first matched. A resulting type of structure could be:

**NEW-ITEM**

last-event: observed schemas  
 context: abstraction  
 initial state: list, values  
 plan: agenda  
 performed-action: sequence performed  
 --->  
 Perception: final-state

Different representations for this procedural learning will be studied. But the accumulation of such elements allows the progressive formation of actual knowledge-bases. Recent work on learning of knowledge-based systems in changing environments could provide cues as to further refinement and expansion of this knowledge [214], and other knowledge-intensive learning methods could be applied [307, 400, 58, 89].

**A LEARNING ARCHITECTURE**

We have presented a learning architecture made of different modules (learning by observation, planning and acting, constructing subjective knowledge) being studied in the World Modelers Project. It concerns only one aspect of the learning issues which should be part of an integrated general architecture [Carbonell (this volume)]. As seen above, the abstractions which account for the behavioral drive act more as *contexts* than as actual concepts. Thus, our learning architecture is geared towards a shift in representational format [61, 68] and the building of domain-specific operators from a domain-general methodology, which will allow the acquisition of a deductive type of reasoning. Furthermore, the domain-general methods, closer to inductive reasoning, should remain active at all stages of development, providing at any time the domain-specific reasoning with a more general but yet meaningful, context or *general state of expectation*. Use of the latter will enhance skill-refinement and possibly focus the agent's attention towards unusual (i.e. interesting) facts. At later stages, when a sufficient corpus of knowledge has been acquired, the use of such methodologies applied to more complex events, in the *background* of all other and domain-specific activities, may give clues as to the mechanisms of phenomena such as accidental (versus empirical) discovery.

## ACKNOWLEDGMENTS

The author was supported by grants from the Fondation FYSSEN and the Fondation pour la Recherche Médicale, Paris. The World Modelers Project is supported by the Office of Naval Research (ONR) under grant numbers N00014-79-C-0661 and N0014-82-C-50767. Many thanks to Keith Barnett, Jaime Carbonell, Klaus Gross, Greg Hood, Pat Langley, Mike Mozer and Hans Tallis for their help and valuable comments. I also thank Jean-Marie Chauvet for his comments.

# A SCIENTIFIC APPROACH TO PRACTICAL INDUCTION

## Larry Rendell

Department of Computer Science  
University of Illinois at Urbana-Champaign  
1304 West Springfield Avenue, Urbana, Illinois 61801

### ABSTRACT

The purpose of *practical induction* is to create systems for powerful (efficient and effective) generalization learning. This paper argues that a *scientific* approach to practical induction promotes discovery of essential principles. Some have emerged from development of the author's learning systems, which have contributed promising methods and unique results.

### INTRODUCTION (Induction in Science and in Machine Learning)

A scientist creates and tests intelligent hypotheses. Experiment may falsify an hypothesis  $H$ ; on the other hand repeated testing may support  $H$  — i.e. raise its *credibility* [386]. For example,  $H$  might be “Localization of credit improves machine learning.” Because many implementations seem to support this, we tend to believe it (although it might be interesting to determine details [297]). The resources of science are limited, so we strive to direct efforts well, and we develop disciplines (methodologies) for this end. Powerful methodologies are both *efficient* and *effective*: they avoid poor hypotheses and promote discovery of credible ones.

### Analogue in machine learning

Hypothesis formation is *induction*, which AI tries to mechanize.<sup>1</sup> In theory, induction presents no problem: hypotheses can simply be generated and tested [9, 386]. In practice, however, the problem is so complex that effective and efficient methods for limiting search are imperative.

---

<sup>1</sup>Mechanized induction inputs *events* or *objects* and produces *classes* or *concepts* for prediction of future events. The importance of automated induction has been emphasized in, e.g. [233].

## **Practical induction: power = effectiveness + efficiency**

The study of *practical induction* in machine learning has two broad goals: construction of *powerful* (effective and efficient) representations and algorithms, and discovery of principles underlying this power. Aspects include noise management, computational complexity, dynamic inductive bias, scope of application, convergence to optimal control structures, etc. [9, 92, 211, 298, 370].

### **Search for principles**

What are the essential ingredients of a powerful inductive system? In confronting this question, some researchers have synthesized systems and created models, although this work is just beginning [9, 45, 92, 233, 293]. Despite the elusiveness of powerful induction, unified models have been aided by well-conceived systems. As is typical of science and engineering, theory guides design and experiment, which in turn hones theory.

### **Thesis of this paper**

In addition to experimentation, scientific methodology includes appropriate abstraction, inclination toward elegant theory, and determination of important relationships (which often become quantitative). The next section of this paper presents an abstraction useful for automated generalization learning. The third section analyzes inductive power. Throughout, we shall argue for scientific investigation of mechanized induction.

## **POWERFUL CLUSTERING (Ideas, Methods, and Clarifications)**

### **Task utility for inductive guidance**

The *utility* is directly related to domain of application [289, 292, 298]. Utility may be the value of an object in task performance, and it may be probabilistic [289, 290, 291, 292, 294, 295, 296, 293, 298]. Probability of task usefulness degenerates to set membership in deterministic cases (a probabilistic utility subsumes positive and negative examples of a concept) [298]. The system *PLS* uses probabilistic methods to induce probabilistic

utility, as utility provides a bridge between domain and induction.<sup>2</sup> Utility determines dynamic bias ([299] cf. [370]), and aids inductive power [298]. Utility embodies ideas of active, goal-directed perception [173].

### Other constraints

Inductive power is related to restrictions imposed on data specification, on forms of classes or concepts, and on algorithmic processing [9, 298, 386]. For example, features (attributes of objects) selected by the user, are designed to compress data even before any mechanized induction [296, 293]. Further, utility almost always bears a smooth relationship to user-selected features. This allows meaningful *clustering* of objects in local neighborhoods of feature space. See [298] for further discussion and more references.

### Cluster analysis

In our view, Samuel designed signature tables to compress similar utilities into feature space cells [313]. Much of this was not automated, whereas PLS1 and PLS2 mechanize the clustering. Cluster analysis is an established statistical technique for inductive inference which partitions similar objects into distinctive classes. Similarities and distinctions are formalized by the use of some (*dis*)similarity criterion. Normally the criterion depends only on features, but this simplification can cause problems [9].

<sup>2</sup>The following is a sketch of *probabilistic learning systems PLS* (see [77, 289, 290, 291, 292, 294, 295, 296, 293, 298, 299, 297] for details):

**Basic system capability.** The original *PLS1* is capable of efficient and effective induction in domains for which the *utility* of an object varies with its features. *PLS1* can handle noise, selecting features which are most discriminating despite error. While it can be applied to single concept learning [92], the system has been developed and tested in the difficult domain of heuristic search, which requires not only noise management, but also incremental learning and removal of bias from data acquired during task performance. The power of *PLS1* has been demonstrated in comparisons with alternative methods [290, 297]. The system can discover optimal evaluation functions, a unique result [292, 293, 297].

**System extension.** *PLS2* is a doubly layered learning system which uses both *PLS1* and a genetic algorithm [148]. *PLS2* Operations performed on utility clusters include generalization, specialization, and reorganization. *PLS2* is more stable, accurate, and efficient than its predecessor [295, 297].

**A system for creation of new terms.** A more ambitious project involves the sophisticated system *PLS0*, designed for substantial *constructive induction* [293, 299]. *PLS0* uses knowledge layering and invariance of utility surfaces to create concepts from progressively validated components. This system appears suitable for problems which were previously intractable [299].

## New kinds of clustering (Utility, Conceptual, and Higher-dimensional)

Criteria based on something other than features are *external* criteria ([5], p.194). Several years ago the author introduced *utility similarity* as a suitable external criterion when the induction relates to performance of some task [289, 290, 292, 298]. Utility similarity involves the whole *data environment*, not just features. Utility provides a firm basis for *conceptual cohesiveness* [233].

Clusters may be constrained, e.g. PLS uses feature space rectangles — conjunctions of attribute ranges. Compressing data into preconceived forms is *conceptual clustering* [233].

PLS0, the author's system for substantial constructive induction, originates a kind of clustering which groups not just attributes, or even simple utilities, but rather utility *surfaces* in subspaces of very primitive features. These surfaces represent interrelationships among components of objects. The process of clustering utility surfaces *creates structure* [299].

### Disguised conceptual clustering

Superficially, Quinlan's ID3 [282] is different from Michalski's systems [233], or from the author's PLS1. But ID3 is a veiled form of utility clustering. ID3 selects attributes having the greatest ability to discriminate. So does PLS1. The *utility dissimilarity* of PLS1 is essentially the *information* of ID3. Once ID3 chooses an attribute, it constructs one branch of the discrimination tree for each attribute value. In contrast, the clustering algorithm of PLS1 splits sets of attribute values only when discrimination is thereby improved. This suggests an obvious modification of ID3, and argues for continued synthesis like [9, 92, 296, 293]. Utility is the sole basis for clustering in PLS1 and "clustering" in ID3.

## WHAT PRODUCES POWER? (Toward Principles)

This section suggests a few incipient principles which may underlie inductive power. All paragraphs but the last refer to *mechanized* induction.

### Mediating structures

Discussed further in [293, 299, 297], this is a proposed addition to Buchanan's model [45]. Successful systems tend to incorporate knowledge

structures which *mediate* objects and concepts during inductive processing. These structures are varied. One codes' growing assurance of provisional hypotheses (through probabilistic information in PLS1). Another mediating structure houses components of tentative concepts (in PLS0). PLS0 employs divide and conquer techniques to build knowledge in chunks of increasing complexity [293, 299]. Hypotheses, gradually and tentatively constructed on lower levels, become confirmed elements of higher level concepts. Consequently the time complexity is improved [299].

### **Representation of whole sets of hypotheses using boundaries**

Mitchell's deterministic candidate elimination for version spaces [244] is efficient because limited boundaries represent whole sets of hypotheses (the boundaries gradually converge). The author's PLS1 is efficient (yet cautious) because tentative boundaries represent the restricted set of partially confirmed hypotheses (boundaries provisionally converge, with increasing assurance).

### **Multiple use of single events in credit localization**

In traditional methods of optimization (e.g. hill climbing, response surface fitting), solving a problem contributes only a single datum. In contrast, probabilistic learning systems like Samuel's checker player and PLS1 make use of every single event (e.g. each state in heuristic search). No one event can errantly overwhelm the system, but still, each one updates knowledge about *every* feature or feature space cell. A similar situation arises in PLS0, only it is much more pronounced. Here a single object provides information about a myriad of object components. (PLS0 focuses on the important ones.) This is reminiscent of *schemata* in genetic algorithms: a single structure codes and supports many combinations and generalizations of its components [148].

### **Mutual data support**

As in the previous paragraph, this involves multiple use of scarce information for the inductive process. *Mutual data support* is a term coined by the author to express a subtle combination of phenomena. In many generalization algorithms (e.g. curve fitting, clustering), the agglomeration of similar events *simultaneously* promotes data compression, noise management, accuracy improvement, and concept formation. Mutual

data support appears in various forms in all PLS systems. See [291, 292, 294, 295, 296, 293, 298, 299, 297], particularly [293, 299].

### **Proper system assessment (How much knowledge is acquired?)**

This point refers not to mechanized induction, but to *our* inference about the power of systems. Precise assessment is important, not simply to know which methods are better, but also to help discover *why* they work well, in order to improve models, theories and designs. We need standards for answering questions such as: How difficult is the inductive task being studied? How much knowledge is acquired autonomously, versus the amount given by the user [298]? To scientifically assess substantial learning in systems like PLS0 , we need to quantify *inductive difficulty* of environments and *inductive power* of systems [296, 293, 298, 299]. This suggests analysis of computational complexity, and measurement of cost effectiveness.

### **CONCLUSIONS (Suitable scholarship)**

In addition to specific methods, results, and contentions in or about mechanized practical induction (generalization learning), we have given a number of suggestions for scientific research in the field: Discovering equivalences in knowledge representations and algorithms is important for clear progress. So is quantification of the power of systems. Our machine learning investigations can also benefit from theoretical issues and results [9]. One example is the highly developed work on credibility criteria by Watanabe ([386], pp.154 ff. ).

# **EXPLORING SHIFTS OF REPRESENTATION**

## **Patricia J. Riddle**

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

### **ABSTRACT**

My research deals with automatically shifting from one representation of a certain problem to another representation which is more efficient for the problem class to which that problem belongs. I am attempting to discover general purpose primitive representation shifts and methods for automating them.

### **INTRODUCTION AND MOTIVATION**

Research in the fields of problem solving, expert systems, and learning has been converging on the issue of problem representation. A system's ability to solve problems, answer questions, or acquire knowledge has always been bounded by the problem representation which is initially given to the system. These systems can only perform efficiently to the extent that their problem representations are relevant to the problem at hand. One solution to this dilemma is to develop systems which have the ability to alter their problem representation automatically. As a problem solving system changes and improves (via learning perhaps), its problem representations must also be altered to fit this new situation.

My research deals with automatically shifting from one representation of a certain problem to another representation which is more efficient for the problem class to which that problem belongs. The study of shifts of representation is composed of two main parts. First, the exploration of various types of shifts of representation. Second, the search through the space of representations to discover which ones are better for certain problem classes and why they are better. I concur with Korf, that "...changes of representation are not isolated 'eureka' phenomena but rather can be decomposed into sequences of relatively minor representation shifts." [184] I am attempting to discover general purpose primitive representation shifts and techniques for automating them. To achieve this goal I am attempting to define and automate all the primitive representation shifts explored by Amarel in the Missionaries & Cannibals (M&C) problem, see [3]. The techniques for shifting representations which

I have already defined are: compiling constraints, removing irrelevant information, removing redundant information, deriving macro-operators, deriving problem reduction operators, and deriving macro-objects. I have not yet implemented these techniques, but those requiring experimental verification will be implemented in my thesis.

## RESEARCH SUMMARY

For the purpose of this summary, I will concentrate on the techniques for deriving macro-operators, deriving problem reduction operators, and deriving macro-objects. The definition of a "good" macro-operator has one main requirement. It must justify the cost of its synthesis by being useful as a primitive operator in solving subsequent problems in the same problem class. I have chosen to make guaranteed general macro-operators. A general macro-operator is one which can be applied at many nodes in the problem's state space graph. A guaranteed macro-operator is one which requires almost no decision time at a node, because the macro-operator should always be used whenever it's preconditions are met.

How can I derive guaranteed general macro-operators which are useful for an entire problem class? The precondition of the macro-operator must not only determine when it is "valid" to apply, but also when it is "good" to apply. To achieve this there must be a part of the precondition which states the macro-operator's purpose. I call this the goal template of the macro-operator. When a operator must be chosen to solve a goal of the problem, the goal templates determine which macro-operator to choose. The system may contain goal templates at many levels of generality. But each macro-operator must contain a specific goal template which states the goal which it directly achieves. This is very similar to the GPS difference tables. The reason I have referred to this as a goal template as opposed to a goal is to stress it's generality. For example, a goal template for a M&C macro-operator might be "moves N people from left bank to right".

If a macro-operator's goal template exactly matched the goal of the problem then it would be the operator to apply. As you can see, a macro-operator's need to match the goal of the problem exactly means that a different macro-operator would be needed for every initial and final configuration of the problem. This kind of macro-operator would not be very useful, because it would be used seldom. It would not satisfy our requirement of generality. To avoid this, the original problem is broken into subparts with their respective subgoals. Now a macro-operator's goal template is matched against the subgoals of each subpart.

If a system could develop general guaranteed macro-operators for a

problem which was given to the system already decomposed into trivial subproblems, it would not be very impressive. Most of the solution to the problem would have already been given to the system in the problem reduction operators. I have developed a technique, called *critical reduction*, whereby the system can derive its own problem reduction operators by analyzing the solution to one problem of a problem class. The process it uses is based on finding the *narrow of the graph*. [3] The *narrow of the graph* is a syntactic term which defines a part of a graph which weakly connects a set of highly connected subgraphs. The *narrow of the graph* is equivalent to the *critical* subpart of the graph.

The basic assumption of *critical reduction* is that the subparts of the problem can be given an importance ordering. That is, there is a certain subpart of the problem which can be defined as the *critical* subpart. The *critical* subpart is defined as the subpart of the solution graph which is the most constrained (i.e., the *narrow of the graph* over a problem class). Once a *critical* subpart is found, the solution graph is divided into three basic subparts: *subproblem* subpart, *critical* subpart, *pseudo-problem* subpart. These subparts are basically the same as those defined by Banerji and Ernst in [14]. The addition of the *critical* subpart was required to supplement their notion of *highest difference*. They assume this difference can be removed by a single operator, whereas I allow a sequence of operators. Lets start by assuming the *critical* subpart of the solution graph has been given. In order to execute this subpart of the problem, the initial state must be transformed into a state where the *critical* subpart's preconditions are satisfied. This initial segment of the solution graph is called the *subproblem* subpart. The *pseudo-problem* subpart is the final portion of the solution graph, which consists of reaching the final goal. The *critical reduction* process is applied recursively to determine if further decompositions are possible.

How is the *critical* subpart of the solution graph chosen? I have chosen a bottom-up approach for finding the *critical* subpart of a problem, which relies on the discovery of *invariants over part of the problem*. The existence of *invariants over part of the problem* is inherent in the *subproblem / critical / pseudo-problem* methodology. *Invariants over part of the problem* and *critical* subpart are really two ways of viewing the same phenomena. There is some state component which is invariant throughout the *subproblem* subpart, it is changed during the *critical* subpart, and it remains invariant again (at some new value) throughout the *pseudo-problem* subpart. For instance, in the M&C problem class all the Missionaries remain on the left side during the *subproblem* subpart, then they all move to the right during the *critical* subpart, where they remain throughout the *pseudo-problem* subpart.

How can we discover these invariants automatically? My method breaks the goal into its natural components and does a backward search on the solution graph to see which component has been invariant for the longest sequence of states. This invariant is then used to decompose the problem into the *subproblem / critical / pseudo-problem* subproblems. Now that the system has derived a decomposition of the problem, it must derive subgoals for each of the subparts of the problem. Knowing the invariants of the subsequent subparts of the problem, it can use the weakest precondition technique through the solution path operators, to arrive at subgoals for each subpart of the problem.

In the previous paragraph I discussed the natural components of a goal. In Towers of Hanoi (TofH) the components used by my system are those which are given in the initial problem representation (i.e., Goal position of disk 1, Goal position of disk 2, etc). In M&C the concept of natural components is more abstract. The initial problem representation (i.e., the formulation generally given to humans) contains a large amount of irrelevant and redundant information. This changes the problem's state description from one containing cannibals Mary, John, and Sue to one containing 3 cannibals. This change from distinct cannibals to the cardinality of a set of identical cannibals allows the creation of the proper goal components (i.e., goal position of missionaries and goal position of cannibals). For further discussion of the creation of macro-objects, see [302].

These techniques work on some problems (i.e., Towers of Hanoi) for which other systems have acquired problem solving strategies (e.g., macro-operators and/or subgoals). [185, 14, 109] And they also work for some problems (i.e., M&C) on which no system that I know of has ever acquired suitable problem solving strategies.

## FUTURE RESEARCH PLAN

My plan for future research is as follows:

1. Explore further the automation of removing redundant information, removing irrelevant information, deriving macro-operators, deriving reduction operators, and deriving macro-objects.
2. Finish the analysis of the shifts of representation in Amarel's paper [3] on M&C. See if there are any other general techniques for shifting representations needed to explain the shifts presented in that paper.

3. Assuming there are new techniques found, try to develop methods to automate them.
4. Implement those portions of the research that need empirical evidence to prove their validity.

## **ACKNOWLEDGMENTS**

I would like to thank my advisor Saul Amarel and the rest of my committee Tom Mitchell, Marv Paull, and Chuck Schmidt. I would also like to thank those people who critiqued drafts of this summary Mike Barley, Michael Sims, and Peter Spool. This research was partially supported by the National Science Foundation under grant DCS83-51523.

# CURRENT RESEARCH ON LEARNING IN SOAR

Paul S. Rosenbloom<sup>1</sup>, John E. Laird<sup>2</sup>, Allen Newell<sup>3</sup>,  
Andrew Golding<sup>1</sup>, and Amy Unruh<sup>1</sup>

## ABSTRACT

In this article we describe five projects in progress on learning in the **Soar** architecture: (1) problem-space creation, (2) learning of macro-operators, (3) data chunking, (4) generalization and analogy, and (5) abstraction planning. We also briefly review **Soar** itself and its learning mechanism *chunking*.

## INTRODUCTION

The **Soar** project is attempting to build a system capable of general intelligent behavior. We seek to understand what mechanisms are necessary for intelligent behavior, whether they are adequate for a wide range of tasks — including search-intensive, knowledge-intensive, and algorithmic tasks — and how they work together to form a general cognitive architecture. A necessary component of this is the development of a general learning mechanism (or a set of mechanisms) and its integration with the problem solving mechanisms. Our research strategy in investigating the architecture in general, and learning in particular, is driven by a set of heuristics. One of the most important heuristics is to favor projects that challenge the weakest points of the architecture or suggest ways of filling in obvious holes. Striving to increase the varieties of learning exhibited by the architecture, and to understand the varieties of learning of which it is not capable, is a central concern of the **Soar** learning research. Another heuristic is to favor projects that demonstrate the generality and power of the current mechanisms by showing how they

---

<sup>1</sup>Knowledge Systems Laboratory, Department of Computer Science, Stanford University, 701 Welch Rd. (Bldg. C), Palo Alto, CA, 94304

<sup>2</sup>Intelligent Systems Laboratory, Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA, 94304

<sup>3</sup>Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 15213

can easily replicate (and sometimes go beyond) a variety of existing results in the field. A third heuristic is to favor projects that can combine synergistically with other parts of the system to produce extra effects.

The five learning projects to be described in this article — (1) problem-space creation, (2) learning of macro-operators, (3) data chunking, (4) generalization and analogy, and (5) abstraction planning — are somewhat disparate, but they all stem from some combination of these three heuristics. Before describing these projects, we briefly review the **Soar** architecture and its learning mechanism *chunking*.

## SOAR AND CHUNKING

The **Soar** architecture [189] is based on formulating all goal-oriented behavior as search in *problem spaces*. A problem space consists of a set of states and a set of operators that move between states. A goal is formulated as the task of reaching one of a set of desired states from a specified initial state. Under conditions of perfect knowledge, satisfying a goal involves starting at the initial state, and applying a sequence of operators that result in a desired state being generated. When knowledge is not perfect, the system may not know how to proceed. For example, it may not know which of a set of operators should be applied to the current state. When such an impasse occurs, **Soar** automatically generates a subgoal to resolve the impasse. These subgoals are themselves processed in additional problem spaces, possibly leading to further impasses. The overall structure is one of a hierarchy of goals, with an associated hierarchy of problem spaces.

In **Soar**, learning occurs as a byproduct of the processing of goals. Whenever a goal is satisfied a *chunk* is created that can generate the results of the goal when a similar situation recurs. These chunks, and in fact all long-term knowledge, are represented as productions. The conditions of the production are based on the working-memory elements that existed prior to the creation of the goal and that were examined during the processing of the goal. These conditions define when a new situation is similar in all relevant aspects to the situation in which the chunk was learned. The actions of the production are based on those working-memory elements added during the processing of the goal that are potentially usable by other goals in the hierarchy. Essentially chunking is performing a form of permanent goal caching.

By now, there is a fair amount of history to the concept of chunking, and growing evidence about its generality. It was first developed, and heavily investigated, as a model of memory in the field of

psychology [238, 84, 35, 159, 67, 66]. More recently it was proposed as a model of human practice [266, 305, 306, 303, 307]. As part of the **Soar** architecture, it has been turned into a machine-learning mechanism that has proved capable of improving a problem solver's performance on a variety of tasks, in a variety of ways — including forms of strategy acquisition, macro-operator acquisition, and knowledge acquisition — and of being able to transfer these improvements during the performance of a single task (learning is incremental and immediately useful), to other instances of the same task (practice effects), and to different tasks (generalization) [190, 191].

The projects described in the remainder of this article extend the investigation of chunking in **Soar** to new domains<sup>4</sup>. We begin with the most important of these projects — problem-space creation.

## PROBLEM-SPACE CREATION

Given a problem space (or a set of problem spaces) for a task, chunking can learn how to solve the task efficiently by learning rules to control the search and rules to efficiently apply the operators. This type of skill acquisition is important, but not all-encompassing. What chunking cannot do, at least at present, is perform the initial knowledge-acquisition process of operationalizing an initial description of the task. That is, it can't acquire the initial problem space for the task. All of the problem spaces that are currently used are instances of a few general problem spaces or of user-created spaces. Indeed, it came as a surprise that we were able to avoid problem-space creation as a major roadblock early in the development of **Soar**. But any substantial degree of generality for **Soar** requires a powerful capability for creating problem spaces.

To approach the issues, we are attempting to create problem spaces from a natural-language description of a task, e.g., from the paragraph describing a typical puzzle such as the Picnic Puzzle. This is one familiar situation where a problem solver moves rapidly from where it does not have a task at all to where it understands its goal and can work on it — i.e., has a problem space. This task of problem-space creation has already been accomplished in the **Understand** program for the Tower of Hanoi puzzle [141]. Thus, we have some strong guidance. The approach there is equivalent to *deliberate creation*, namely, working in a problem space for

<sup>4</sup>Chunking is the first and only learning mechanism in **Soar**, but other mechanisms may eventually be necessary.

building problem spaces. However, our initial approach, *en-passant creation*, is one where **Soar** attempts the given task in the natural-language-comprehension space in which it initially acquires the tasks. As this attempt fails (as it must, since solving such puzzles involves much more than just parsing the sentences) the opportunities should arise for developing bits of representation and specialized operators. Under continued problem solving, functionally irrelevant aspects of the initial problem space should become structurally irrelevant as well, and a specialized problem space should emerge.

**Soar**, of course, must ultimately be able to do deliberate problem-space construction, and we expect to get **Soar** to do these natural-language tasks in the manner of the **Understand** program. But the scheme for constructing spaces *en-passant* seems both elegant and important. If successful it should be a technique that can apply in all situations (yielding strongly synergistic effects) and it should challenge the basic **Soar** design in fundamental ways. This project is in an early phase of its development, with no significant results yet to report.

## LEARNING OF MACRO-OPERATORS

Korf [183] has recently shown that any problem that is serially decomposable — that is, there is some ordering of the subgoals in which each subgoal is dependent only on the preceding subgoals, and not on the succeeding ones — can have a *macro table* defined for it. Once the macro table is created (by a search-based preprocessing phase), it enables efficient solutions from any initial state of the problem to a particular goal state. For the Eight Puzzle, a macro table can be created if the goals are, in order: (1) place the space in its correct position; (2) place the space and the first tile in their correct positions; (3) place the space, the first tile, and the second tile in their correct positions; etc. Each goal depends only on the tiles already in position and on where the one new tile currently is. The macro table is a simple two dimensional structure in which each row represents a goal, and each column represents the position of the new tile. Each macro specifies a sequence of moves that can be made to satisfy the goal, given the current position of the new tile. The macro-operator technique proves especially useful in tasks such as the Eight Puzzle (and Rubik's cube) in which traditional problem solving techniques such as means-ends analysis and hill climbing are ineffective.

This work is of particular relevance to **Soar**. It describes a new weak method, interacting with the work in **Soar** on a *universal weak method* [188]. It shows how a large degree of generality can be obtained

with a simple caching scheme — the generality comes from using a single goal in many different situations. And it looks like **Soar's** chunking mechanism could learn macro tables in a straightforward manner. We thus set out to replicate Korf's results in **Soar**.

The current state of this project is that a macro table has been acquired by **Soar** for the Eight Puzzle [191]<sup>5</sup>. The implementation consists of two problem spaces, one containing the normal eight-puzzle operators (up, down, left, right), and one containing operators corresponding to the serially-decomposable goals. The work goes beyond Korf's by showing how: (1) the technique can be used in a general, learning problem solver without the addition of new mechanisms; (2) macro tables can be learned incrementally during problem solving, rather than requiring a preprocessing phase; and (3) even more generality can be obtained via transfer of learning between macros in the table.

## DATA CHUNKING

**Soar** has one long-term memory — production memory — and a learning mechanism (chunking) that is capable of adding new rules to that memory. This combination works well for representing and acquiring procedural knowledge. But this is clearly not enough; a complete architecture must be capable of representing and acquiring long-term declarative (fact) knowledge as well, a problem that we call *data chunking*. In general this problem is solved in almost all systems simply by having a separate mechanism for adding information to a declarative long term memory. For instance, in **Act\*** there is simply a fixed probability of making a new fact a permanent part of declarative memory [6]. Though the same approach could be taken with **Soar**, there are several reasons to suppose that the current mechanisms in **Soar** ought to be adequate on their own: (1) a rule can be viewed as the combination of a fact (the rule's action), and an access path to that fact (the conditions); (2) the rules created by chunking are based on the information in the system's working memory, a short-term repository of facts; and (3) the original formulation of chunking was as a model of declarative memory.

The goal of the data-chunking project is to establish whether **Soar's** current mechanisms are adequate for the learning of declarative information. We have begun by looking at simple paired-associate learning. Given a list

<sup>5</sup>The searches on which this table was based were hand directed to reduce the amount of computation time required.

of stimulus-response pairs to be memorized (such as, *wub* → *ba*), the task is to be able to recall the correct response given a stimulus. On first glance it appears that chunking should easily be able to accomplish this type of learning because it directly learns stimulus-response pairs (that is, productions). The approach would be to examine the stimulus, and then generate the response, hoping to yield a rule with the stimulus in the condition and the response in the action. The problem is that chunking does not learn arbitrary pairs; what is learned is determined by the problem solving performed. The example does not work because the response must be examined in order for it to be generated. Thus, the response will be one of the conditions, making it impossible to retrieve the response given just the stimulus.

One approach to this difficulty is to modify the rule matcher to do some form of partial matching. Another approach is to figure out what problem solving would be required in order to learn the desired rule. It turns out that the correct rule can be learned if the rule described above (the one with both the stimulus and response in its conditions) is used as a goal test in a search that tries to generate the response without examining it. This leads to the two-process theory of recall — search combined with a recognition test — familiar in psychology (see, for example, Anderson & Bower [8]). This project has made it to the initial proof-of-concept stage, with a demonstration that the desired rules could be learned in small test cases. More work is required before this can be turned into a general facility for remembering facts.

## GENERALIZATION AND ANALOGY

Anderson [7] showed how analogical problem solving could lead to the learning of generalized structures, and Greiner [134] showed how generalized structures could be used to perform analogies. Which comes first, generalization or analogy? This project is an attempt to: (1) clarify the relationship between analogy and generalization; (2) investigate further the issues involved in learning more general rules than are automatically generated by chunking; and (3) investigate using chunking to learn new knowledge from the outside world. The approach we take is to look at how **Soar** can make use of the different types of information that an advisor can provide when an impasse is reached in problem solving.

The simplest possibility occurs when the advisor directly suggests a way of resolving the problem. For example, if the impasse occurred because it was unclear which of several acceptable operators should be applied, then the advisor would specify the correct operator. Rather than

take the advisor's word on faith, a goal is set up to ascertain whether the operator is indeed the right thing to do. The processing of this goal leads to the learning of a chunk which will produce this piece of advice in relevantly similar situations. This type of learning — using the verification of a fact as a means of creating more general knowledge — has been referred to variously as creating *justifiable generalizations* and as *explanation-based generalization* (see, for example, [256]).

A type of advice that is one step removed from directly giving the answer, is advice about a similar but simpler problem, and of what the right step is in that problem. The approach to using this kind of advice in **Soar** is to solve the simpler problem, using the advice about what to do. Then, if the two situations are similar enough in the relevant aspects, a chunk should be learned which directly transfers to the original problem. This is a type of *analogy by generalization*. However, if the problem specified by the advisor is too different from the problem currently being solved, the chunks learned will not directly transfer. Problem solving must then be done to transfer the lessons learned between the two problems. If chunking behaves like the similar mechanism of knowledge compilation [7], then chunking this transfer process should lead to a form of *generalization by analogy*.

This project is proceeding stepwise through these three types of advice-taking, with work currently in progress on the second step — learning from a closely related problem (or analogy by generalization).

## ABSTRACTION PLANNING

Planning is a general capability of an intelligent agent. It should be possible to plan in any problem space, although whether it helps or not certainly depends on the details. Two major types of planning have been explored in AI. One is *abstraction planning*, epitomized by **GPS** planning [267] and **Abstrips** [308], where the system plans by solving a problem in a simplified space formed by abstracting away from some details of the original space. The states and operators of the planning space are abstract versions of the states and operators of the original space. The other is *deliberate planning*, epitomized by **Noah** [309] and **Molgen** [361], where planning occurs in a problem space whose states are partial plans.

Abstraction planning appears to be a natural uniform activity in problem solving [267] and it appears to translate into a natural uniform activity in **Soar**. In fact, **Soar** already engages in a form of (non-abstract) planning. When it does not know what to do next — because, for example, there is more than one operator that can be applied to the

current state — it performs a search to determine which of the alternatives will work (or work best). While searching, it learns chunks that efficiently apply the complex operators to be executed later in the task, and it learns search-control chunks that tell it which operators to use in its future performance. The search-control chunks embody a plan for **Soar's** performance on the remainder of the task. Thus, though planning is normally not considered to be closely related to learning, the use of chunking as the means of storing plans, makes learning an integral component of this project.

The first step towards abstraction planning in **Soar** is to convert the look-ahead searches into searches of abstracted problem spaces. Both **GPS** and **Abstrips** abstracted by ignoring certain things that had to be done, **GPS** by ignoring *differences* and **Abstrips** by ignoring *preconditions*. Initially in **Soar**, we are abstracting by ignoring *operators*. It is possible to ignore whole operators, or to work with abstract operators by taking advantage of the fact that a complex operator is implemented via a problem space (part of the goal/problem-space hierarchy alluded to earlier). By ignoring some of the operators in this subspace, the original operator becomes abstracted — it pays attention to only some of the features of the original situation, and generates only some of the original actions.

The second step towards abstraction planning is to learn chunks from the abstracted search. The third and final step is to apply those chunks to the original problem. If a chunk suggests an operator that can be applied, it will be applied. If a chunk suggests an operator that cannot yet apply, **Soar** should fall back on operator subgoaling to find a state to which it can apply. If a chunk suggests an operator that turns out to be incorrect — abstraction problem solving is by definition only approximate — **Soar** should back up and try another approach.

This project has so far demonstrated the ability to delete operators from problem spaces, search in the abstracted spaces, learn chunks, and transfer them to the original problem. More work is required on the issues of filling in gaps in the plan, filling in gaps left by the execution of abstract operator-implementation chunks, backtracking on failure, understanding the constraints that must be placed on problem spaces so that abstraction will work in any problem space, and deciding what aspects of the problem space to abstract.

## CONCLUSION

This article has described five projects that are working towards the goal of having a single system that can do all forms of learning. These

projects exhibit three themes. The first theme is that learning and problem solving interact to a surprising extent, and to their mutual benefit. Learning helps problem solving by acquiring the initial problem space so that problem solving can occur, by storing plans, by acquiring macro-operators and search-control information, and by transferring knowledge from analogous problems. Problem solving helps learning by isolating regions of time over which it is useful to learn (the span of processing of a goal), by focusing on what is important (leading to properly general chunks), by performing abstraction (leading to more generality), by performing difficult analogical mappings (leading to even more generality), and by arranging for data chunking to work. These strongly synergistic effects are part of the pay-off from building a complete architecture.

The second theme, which is closely related to the first, is that the degree to which what is learned is general is critically dependent upon the kinds of problem solving done. The search for generality in **Soar** does not involve experiments with induction algorithms such as AQ11 [230] and Version Spaces [245]<sup>6</sup>. Instead, the learning mechanism provides only a basic ability to abstract away aspects of the world not relevant to the problem solving. The different types of problem solving — whether straightforward search, search in an abstracted space, or analogical problem solving — provide experiences of different generality to the learner. It is an interesting open question as to whether this is all that is required to learn suitably general information.

The third theme is that, though chunking has been developed as a skill-acquisition mechanism (that is, to speed up what the system can already do), and not as a knowledge-acquisition mechanism, there is no apparent reason why it cannot also perform knowledge acquisition. The projects on problem-space creation, data chunking, and analogy are all investigating how chunking can be used to learn new knowledge from outside of the system.

## ACKNOWLEDGMENTS

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contracts F33615-81-K-1539 and N00039-83-C-0136, and by the Personnel and Training Research Programs,

---

<sup>6</sup>We have implemented a version-space problem space in *Soar*, but it has languished because of a lack of synergy with the rest of the system [316].

Psychological Sciences Division, Office of Naval Research, under contract number N00014-82C-0067, contract authority identification number NR667-477. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the Office of Naval Research, or the US Government.

# **LEARNING CONCEPTS IN A COMPLEX ROBOT WORLD**

**Claude Sammut and David Hume**

Department of Computer Science, University of New South Wales  
P.O. Box 1, Kensington NSW, Australia 2033

## **ABSTRACT**

An overview is presented of a collection of programs which simulate and learn concepts in a complex, three dimensional robot world. Objects in the world may be highly structured. There may be several agents active in the world simultaneously. The task of some of these agents is to learn the structure of the world and how to plan sequences of actions. Several different learning systems are currently under development by members of the AI group at the University of New South Wales.

## **INTRODUCTION**

Learning programs have often been restricted to limited, well-behaved environments. Induction of concepts in such environments is relatively easy since the state of the world changes very little. The learning programs described here operate in a very rich simulated robot world. Objects and actions in Magrathea [154] can be very complex. Objects may have articulated joints. They can be fixed, mobile or motile (i.e. able to move themselves) and there may be more than one actor in the world apart from the learning system itself. The world also simulates physical laws, collisions and perception from different view points. The simulation program is written in Prolog and C and runs on a SUN workstation.

Current work involves the development of learning elements which are capable of performing experiments which will enable them to deduce the laws that govern the simulated world and how to plan actions in the world. Some of the problems involved in learning in the robot world are presented below.

- Because the learning system does not have complete control over its environment, it cannot assume that instances of concepts will be seen in a ideal setting, therefore, a concept or theory can only be considered an approximation to the truth. Later evidence may refute or modify the theory. In that case,

the system must be able to cope with a knowledge base which is not provably correct. It must also be able to revise the knowledge base in the light of new evidence.

- The system learns new concepts by observing the world and attempting to match objects and events with patterns in the system's memory. Observations may include configurations of objects or the actions of another being in the world. There is a "parent robot" which can perform actions which are observed and imitated by the "child robot" (learning system).
- The learning element must plan experiments in order to test theories. These experiments must be well chosen so as to avoid unnecessary searching of incorrect theories. In general it is not possible to construct an experiment which will confirm a theory beyond doubt. However, an attempt must be made to perform an experiment which will yield significant new knowledge.
- Since there may be other active beings in the world, the state of the world may change, not as a result of actions by the learning system. The presence of more than one actor opens the possibilities of learning cooperative behavior.

## ARCHITECTURE

The world model and the various learning programs are intended to be as independent of each other as possible. To achieve this independence, the system is designed to consist of a number of Unix processes, each one running one module. Currently there are five modules either in existence or now under development:

- Magrathea (the world model).
- A natural language front end.
- Marvin (a program which learns by imitation).
- Galatea (an autonomous learning program).
- STANLIE (a language learning program).

The robot world maintains a data base which describes the world at any instant in time. This state is also displayed on a graphics display. Other processes can interact with the world by sending messages to actors in it. Processes communicate via Unix pipes. For example, if a robot called fred has been created, another process may send a command such as "fred: stand behind chair and touch it". Fred's movement is then shown on the screen and the state of the world is updated. At present, commands are in a very stylized form, however, a natural language front end to the simulation is under development [170] so that human trainers can interact with the system easily.

## MAGRATHEA

The representation of objects and events in the robot world is central to the operation of the system. Objects are declared to be either fixed, mobile or motile. Each object is given a name, colour, 3-D position, orientation, shape and dimensions. If the object is motile, then the type of movement possible is also given. If an object is part of a larger structure, the method of attachment to the rest of structure can be specified. Attachment may be fixed as a leg to a table or free as an arm to a body. In the latter case, the degrees of freedom of the object must also be specified. In this way a limb with joints can be specified.

Each agent in the world has simple forms of sight, hearing and touch. For example, to allow fred, the robot, to see, it must be given an eye. The declaration for the eye includes the field of view. An observing process may request a list of objects which the robot can see at any time. This list can be "taxonomic", i.e. the name of each object is given, or "geometric", i.e. only the shape and orientation is given. In the latter case, the observing processes must work out which object it is looking at. Whenever a change occurs in the world, signals are sent to the other processes in the system. These changes may trigger responses from those processes.

## LEARNING

Learning can occur in a number of different ways. A parent robot may perform a sequence of actions and a "child" robot attempts to imitate and then generalise that sequence. This program follows the learning pattern of Marvin [311]. The learner generalises the description of a structure or action and tests this generalisation by performing an experiment. The experiment may involve creating a new structure or

carrying out a plan. Criticism of the generalisation comes from the world itself. For example, the learner's generalisation may result in the construction of an unstable structure. Since the experiment fails because the structure cannot be built this indicates that the generalisation was incorrect. As with Marvin, once a concept is learned, it remains in the program's memory. That concept may then be used to build more complex concepts. All structures and events are represented in predicate logic. This includes sequences of events which represent a plan.

Another learning system being developed by Brebner [40] attempts to autonomously discover theories about the robot world. The program uses ideas of paradigm, paradigm-shift and falsification of theories from the philosophy of scientific discovery. The program, called Galatea, begins life with a simple paradigm which is used to explain the surrounding world. Based on this paradigm, the system predicts the behavior of the world. If a prediction fails, ad hoc hypotheses are added to explain the failure. If all possible theories under a paradigm are falsified then a "paradigm shift" may occur. A paradigm shift is a change in perception of the world.

Powers [281] is developing a language learning system called STANLIE. The idea behind this system is that children do not learn language in isolation from the world around them. A parent will often point to an object and say word, e.g. "ball". That is, language learning always takes place in the presence of a number of stimuli. Therefore it is interesting to put a language learning program into a rich world so that it may benefit from additional information just as a human child does.

## ACKNOWLEDGMENTS

The robot world, Magrathea, was originally defined by David Powers [281] in connection with his work on learning language. He is now at Macquarie University, Sydney. Paul Brebner's work on Galatea was begun as a Master's thesis at the University of Waikato in New Zealand.

# **LEARNING EVALUATION FUNCTIONS**

**Patricia A. Schooley**

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

## **ABSTRACT**

This research proposes a scheme for learning state evaluation functions for heuristic search problem solving systems. The proposed scheme employs both empirical and analytical learning techniques. Characterizations of states which are  $n=1,2,\dots$  steps from the goal are learned. These characterizations are then used to define a state evaluation function.

## **EMPIRICAL VS ANALYTICAL METHODS**

Empirical learning methods usually lead to quite tractable (i.e. easily evaluated) state evaluation functions. However, the accuracy of the functions derived by purely empirical methods has been disappointing. Samuels [313] used empirical techniques to discover features of a checkers board state that were relevant to determining the distance to the goal the game. Samuels' system often proposed features that had little relevance.

One analytical learning technique is based on the back propagation of the goal state and the operator preconditions through the solution trace, as was described by Mitchell. [253] This back propagation provides sufficient conditions for a state to be  $n$  steps from the goal. Analysis of additional solution traces may provide other sets of sufficient conditions for a state to be  $n$  steps from the goal. The disjunction of these sets of sufficient conditions provides an increasingly more accurate description but a less tractable evaluation function.

## **DEFINING AN EVALUATION FUNCTION**

Given CHAR1, CHAR2,...CHARn, the characterizations of what it means to be 1, 2,...,n steps from the goal, an evaluation function, f, can be defined as:

$$f(S) = \min\{m / S \text{ matches } CHAR_m\}$$

That is, f evaluated on state S is the minimum m such that S matches

the characterization of the set of states which are  $m$  steps from the goal.

Problem solver solution traces provide positive examples of states which are a particular distance from the goal. Essential to learning from these examples is a generalization language in which to describe the sets of states which are distance  $n$  from the goal. The description language will determine the relevant features used to describe sets of states. [246]

## DIFFERENCE CHARACTERIZATIONS

For any  $n$  (not equal 0) a state which is  $n$  steps from the goal will exhibit differences from the goal, regardless of what representation scheme is used to represent problem states. Therefore, a language which describes the differences between a state and the goal could be used to characterize the set of states which are distance  $n$  from the goal. This difference characterization approach is prompted by several observations.

- Human problem solvers appear to use differences between the current state and the goal as a guide in problem solving.
- Solutions to some fairly complicated problems (such as Rubik's cube) are often stated in terms of differences between a state and the goal. Sequences of operators are given which will resolve particular types of differences between the current state and the goal.
- The concept of differences between a state and the goal is a meaningful one in any problem solving domain.
- Differences between any two states, not just between a state and the goal can be described allowing the use of the same characterizations for problems in the same domain which have different goals. [101]

## AN INCREMENTAL ALGORITHM

The simplest difference characterization is obtained by simply counting the number of syntactic differences between a solution trace state and the goal. The set of states which are  $n$  steps from the goal could initially be described as those states having the same number of differences from the goal as does the solution trace state which is  $n$  steps from the goal. If states (including the goal state) are represented by a list of predicates, the

number of predicates differing between a solution trace state and the goal would simply be counted.

For improved accuracy of the evaluation function these characterizations can be refined by noting the particular types of solution trace predicates that differ from the goal and in what variables these predicates differ. Then, if in succeeding solution traces a difference in a particular type of predicate does not appear, that difference can be removed from the difference characterization since that difference is not always essential for states which are  $n$  from the goal. The elimination of such predicates from the difference characterization helps to control the complexity of the resulting evaluation function.

A further revision of the difference characterizations, based on an analysis of the solution trace, is possible. The preconditions of the operators in the solution trace are constraints on the differences between the solution trace states and the goal. Back propagating the operator preconditions through the solution trace provides constraints on the differences in the difference characterizations for each value of  $n$ .

Of particular importance are the "static" preconditions of the operators. These are the precondition predicates which cannot be created or altered by the application of any operator. These "static" precondition constraints are essential for a state which is some particular distance  $n$  from the goal.

At this point the difference characterization for each  $n$  consists of a set of constrained differences between a solution trace state and the goal. Succeeding solution traces may contain a state which is  $n$  steps from the goal, but which does not match the current difference characterization for  $n$ . This would lead to repeating the above procedure on this new solution trace to obtain a set of constrained differences between this state and the goal. The new difference characterization for  $n$  will then become the disjunction of these two constrained differences descriptions.

## PROGRESS OF THE RESEARCH

The proposed learning scheme has been tested in a hand example on the 8 tiles puzzle. Hand examples in other domains are in progress. The current plan is to implement the learning scheme to facilitate experiments in more complex domains.

Further research is planned to investigate the issue of the accuracy vs the tractability of learned evaluation functions. Of particular interest is the contributions made toward these twin goals by empirical vs analytical techniques for learning.

# **LEARNING FROM DATA WITH ERRORS**

## **Jakub Segen**

AT&T Bell Laboratories  
Holmdel, N.J. 07733, USA

### **ABSTRACT**

An inference criterion representing a trade-off between "fit" and simplicity leads to a general method for constructing concept descriptions from a class of descriptors, given examples with errors. In one application, this method is combined with a specific descriptor generator into a program learning structural descriptions of shape.

### **INTRODUCTION**

A question of major importance for learning from data with errors is the choice of the preference criterion for ranking competing hypotheses or generalizations. The problem with statistical criteria based purely on the degree of "fit" arises when the language used to express hypotheses allows one to improve an imperfectly fitting hypothesis by creating a more complex one with a better fit. For example a high degree polynomial will usually fit better to a given set of measurements of voltage and current, than the Ohm's law. If two hypotheses provide equal fit, then it might be appropriate to choose the simpler one. However, significant errors in data usually result in a monotonic trade-off between fit and simplicity of a hypothesis. A general inference criterion, called minimal representation criterion, has been proposed [323], to reflect this trade-off. It combines the measure of fit with a measure of complexity of a hypothesis, model, or a description. This criterion is derived by relating both measures to the length of the shortest program generating the observed data. The minimal representation criterion in its general form requires minimizing the expression:

$$S(P(y)) - \log_2 P(y^n)$$

where  $y^n$  is a sequence of observations, and  $P(y)$  is a probability distribution describing both model and errors. The first term is the

number of bits needed to specify the probability distribution  $P(y)$ , the second term represents the degree of fit. This criterion has been applied to discover patterns in a continuous signal and in a symbol sequence, in signal modeling to choose the order of a model, and in clustering to select the best number of clusters [323, 314].

## LEARNING CONCEPT DESCRIPTIONS

The minimal representation criterion has been used to derive a preference function for learning concept descriptions from positive and negative training examples with errors [324]. The assumed objective is to find a set of functions, or descriptors  $f_1, f_2, \dots, f_k$ , that jointly provide the best discrimination of a concept C against all other concepts. The resulting preference function is of the form:

$$Q(C, f_1, f_2, \dots, f_k) = D(C|f_1, f_2, \dots, f_k) + \sum_{i=1}^k S(f_i)$$

where  $\text{math}(S(f))$  is the complexity of a descriptor, and  $D$  is a negative log likelihood computed from the set of training examples. The value  $-D$  represents the overall fit, or agreement of the description with training data. The best description is found by minimizing  $Q$ .

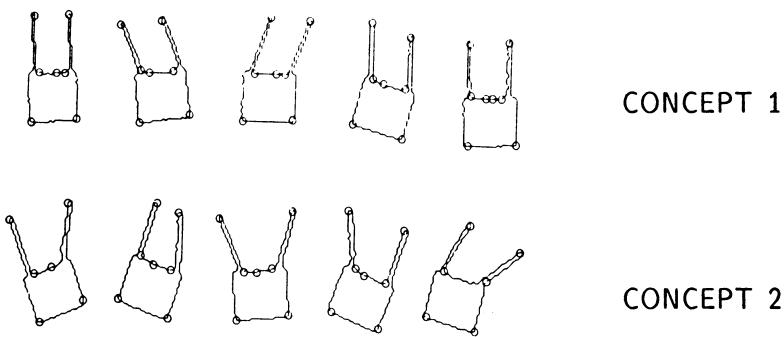
The power of this criterion comes from the relatively unrestricted form of descriptors. A descriptor can be a predicate, an arithmetic expression, or generally a program. The complete learning system consists of two modules: a descriptor generator and a description constructor. The descriptor generator is a domain dependent process that generates a stream of descriptors ordered by increasing complexity. The description constructor is a program that searches this stream for a set of descriptors minimizing  $Q$ . While it is shown that this search can be completed in a finite time, in most cases an exhaustive search for the optimal solution is too costly. Therefore, a fast, sub-optimal description constructor has been developed [325]. This constructor proceeds incrementally. First, it scans a given list of descriptors, if necessary requesting the generator to expand it, until it finds a single best descriptor. Then it seeks additional descriptors that improve  $Q$ , until it becomes certain that no further improvement will be achieved by adding another descriptor.

## LEARNING STRUCTURAL DESCRIPTIONS OF SHAPE

Most of the current work on structural representations of shape concentrates on finding complete or characteristic descriptions, in a form of a graph. Since comparing two graphs is an NP-complete task, such descriptions are difficult to use when identification requires comparing an instance against a large number of descriptions. The method shown in the previous section provides a basis for learning incomplete but more efficient descriptions, by capturing only the most discriminating structural characteristics. A program learning such descriptions has been described in [325]. Its outline is provided below.

A shape is extracted from an image by reducing it to a set of curves, using edge or region extraction techniques. It is presented to the learning program as a set of critical points, that correspond to local extrema of curvature, (see Fig. 1). Each critical point is characterized by a position, curvature and local curve orientation. From a group of shape instances the descriptor generator derives symbolic primitives called structural patterns, that represent common configurations of critical points. These configurations are either single critical points, or binary structures such as pairs of points, pairs of pairs, etc. The language of structural patterns is built hierarchically, using patterns formed at one level to express the next level of binary structures. The simplest patterns are found by clustering the critical points in training instances, based on curvature. Each point is then identified by a label given to the corresponding pattern. The binary structures at the next level are built by pairing each labeled point with its closest neighbors, and characterizing each pair by the component labels, distance, and difference of orientations. The structural patterns at this level are formed by clustering the pairs, using these parameters. The higher levels of structures and structural patterns are generated in a similar way. For each structural pattern the generator creates a descriptor, which is a predicate indicating a presence of this pattern in an instance. Each request from the description constructor results in generating one level of descriptors, then the generator stops and waits for another request.

Descriptions that are built by this program are powerful enough to discriminate among non-rigid shapes. An example in Fig. 1 shows instances of two shape concepts that are identified correctly using the descriptions provided by the program.



**Figure 1:** Instances of two shape concepts that are distinguished using descriptions inferred by the learning program.  
Circles identify critical points.

# **EXPLANATION-BASED MANIPULATOR LEARNING**

## **Alberto Maria Segre**

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign  
Urbana, IL 61801

### **ABSTRACT**

This paper describes a robot manipulator system under development which learns from observation. The system observes manipulator command sequences that solve problems currently beyond its own planning abilities. General problem-solving schemata are automatically constructed via a knowledge-based analysis of how the observed command sequence achieved the goal. The acquired schemata serve two purposes: they allow the system to solve problems that were previously unsolvable, and they aid in the understanding of later observations.

### **INTRODUCTION**

The introduction of robot manipulators has had an enormous impact on the manufacturing community. The scope of this impact is indeed remarkable if one considers that today's manipulators do not exploit their full potential. This lacuna can be attributed to the fact that while manipulators are indeed *general purpose*, they are still far from *flexible*.

This distinction does not refer to any shortcoming in the actual design of the mechanical arm, but rather is meant to reflect the gap that exists between the theoretical capabilities of a manipulator and the difficulties involved in preparing it to perform some new task. We call this problem the *retraining problem*.

### **EXPLANATION-BASED MANIPULATOR LEARNING**

To correct this problem, we propose to replace robot teaching with robot learning: to shift the burden of designing new task sequences from the operator to the robot by endowing the manipulator with a *conceptual level* for task specification and learning [326].

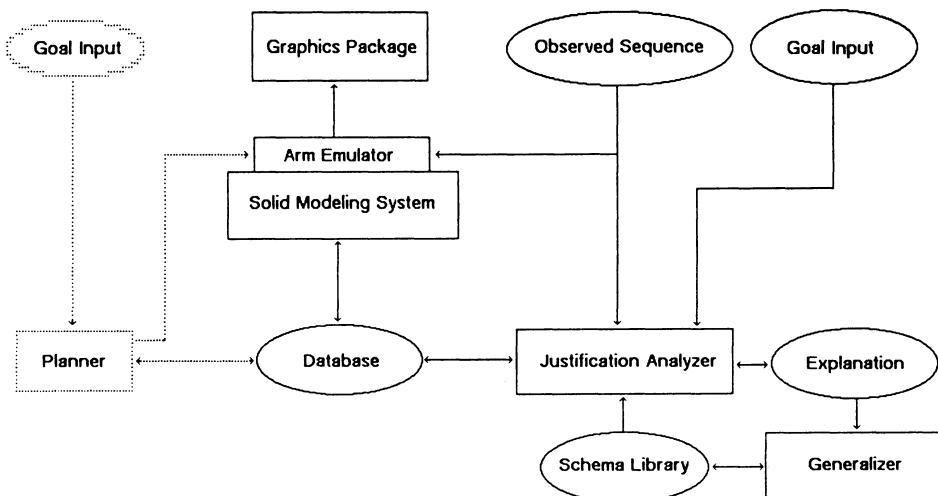
Our approach is based on work in *explanatory schema acquisition* [DeJong (this volume)]. Such explanation-based systems are capable of acquiring useful knowledge on the basis of a single problem-solving episode.

For this to be possible, these systems rely on extensive domain-specific knowledge both while observing the sample problem-solving episode, and in guiding the generalization process.

In our system, domain knowledge is encoded in *schemata* [240] which represent the system's knowledge of a particular topic. For example, a schema for a block would include its dimensions, grasping techniques, its current location, etc. In our system there are three kinds of schemata: physical object schemata, mechanism schemata, and task schemata which represent actions and action sequences.

## DESCRIPTION

The figure shows the architecture of our explanation-based system. Observed input is in the form of a sequence of primitive gripper commands, together with a goal specification.



The gripper commands are fed to both the arm emulator and the justification analyzer. A software arm emulator is currently being used, although a real robot arm is being interfaced with the system. The goal specification is given in terms of a mechanism schemata, which gives a functional (as opposed to physical) description of the assembly.

The justification analyzer constructs a causally complete *explanation* of how the observed gripper commands achieve the specified goal. The

justification phase is similar to the processing of other *understanding* systems ([107] among others), with one important difference: in addition to the causal links interrelating the input actions, the explanation includes the set of *constraints* crucial to the successful completion of the task.

The explanation is constructed from task schemata in a bottom-up fashion. As the number of schemata in the system increases, we are quickly faced with the combinatorially explosive *frame selection problem* [65]. Our solution to the frame selection problem is embodied in the system's schema suggestion/activation/cancellation/closure mechanism [326].

By applying domain-specific background knowledge to the explanation constructed, the generalizer produces a new task schema which does not reflect the peculiarities of the observed problem-solving episode. It is independent of the initial piece placements and is free of redundant steps. Any observed gripper command which does not contribute to the specified goal is removed before generalization and thus does not appear in the new schema.

The resulting schema can also be used to plan functionally similar assemblies using new pieces of differing size and shape. These new pieces must, however, share the same function in the new assembly with the corresponding pieces in the original example. This is because the generalizer is also applied to the pieces which fill the roles in the new schema. The constraints are used to limit the extent of this generalization so that the resulting schema is causally correct (e.g., a square peg does not fit a round hole).

The planner applies schematic knowledge from the schema library when presented with a new situation. The dotted lines in the figure indicate the planner and justification analyzer do not operate at the same time. Schemata from the schema library are indexed by the goals they achieve. Once a schema is selected, the planner formulates its preconditions as a series of subgoals to be achieved.

## RESULTS

A first implementation of the system was applied to an example consisting of a peg inserted into a bored block through a loose washer. The system observed a non-optimal 19 step assembly sequence from which a new task schema was created. The observed sequence was crucially dependent on the presence of a redundant piece in the workspace.

The new task schema was then applied to a number of different initial piece placements. For the placement in the original example, a 17 step assembly sequence was generated which did not rely on the presence

of redundant pieces in the workspace. For a simpler initial piece layout, the generated sequence was of only 6 gripper commands, reflecting the power of this approach in taking advantage of serendipitous situations during planning. Finally, when applied to a more complex initial piece placement, the generated sequence was of length 22, demonstrating the ability of the system to untangle the more complicated initial layout while still successfully accomplishing the construction task.

## CONCLUSION

The approach we have discussed describes a system which can improve its effectiveness while greatly decreasing the human programming time of robot manipulators. A manipulator can be taught by leading it through a task. The observed sequence need not be a perfect sequence: it must simply be effective in accomplishing the stated goal. The system acquires a general solution which not only enables it to react to variations and exceptional conditions (an ability previously bestowed only by trained robot programmers through hours of work), but also attempts to provide a more efficient solution by removing redundant or needlessly inefficient steps. This combines the ease of instruction and cost-effectiveness of teaching systems with the flexibility of manipulator-level programming language systems; and as an added bonus shifts the programming burden from the human teacher to the robot learner.

## ACKNOWLEDGMENTS

This work was supported in part by the Air Force Office of Scientific Research under grant F49620-82-K-0009 and in part by the National Science Foundation under grant NSF-IST-83-17889.

# LEARNING CLASSICAL PHYSICS

## Jude W. Shavlik

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign  
Urbana, IL 61801

### ABSTRACT

A computational model of learning in a complex domain is being developed. This model supports knowledge-based acquisition of problem-solving concepts from observed examples, in the domain of classical physics. The current implementation of the model learns about momentum conservation, in a psychologically plausible fashion, from a background knowledge of Newton's laws and the calculus. Using its physical and mathematical knowledge, the system determines *why* the human-provided solution to a specific problem suffices to solve the problem, and then extends the solution technique to more general situations.

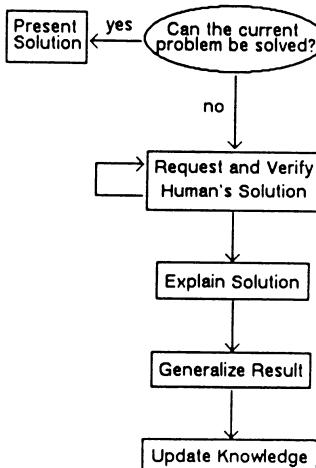
### INTRODUCTION

*Explanation-based learning* [87] is a knowledge acquisition method that utilizes deep domain models. This type of learning involves generalizing a single problem solution into a form that can be later used to solve conceptually similar problems. The generalization process is driven by the *explanation* of why the solution worked. It is the deep knowledge about the domain that allows the explanation to be developed and then extended. This approach to learning has much in common with [254, 243, 342, 401]. See [90] for a full discussion.

A system has been designed and implemented in which explanation-based learning is applied to classical physics. Initially this system is capable of performing many of the mathematical manipulations expected of a college freshman and is provided a representation of Newton's laws. Physical concepts taught in a first semester physics course are to be acquired; hence the name of the system, **Physics 101**. Newton's laws - which constitute the initial domain model - suffice to solve all problems in classical mechanics, but the general principles that are consequences of Newton's laws are interesting for their elegance as well as their ability to greatly simplify the solution process. Since the system's physical knowledge rests on the strong foundation of Newton's laws, only its mathematical sophistication will limit the physical concepts it can acquire.

## SYSTEM OVERVIEW

The operation of the system can be seen in figure 1. Attention is currently focused on learning from the example solutions of a human user. After a physical situation is described and a problem is posed, **Physics 101** attempts to solve the problem. When the system cannot solve a problem, it requests a solution from the human user. The solution provided must then be verified; additional details are requested when steps in the human's solution cannot be understood. The process by which the system understands an example is divided into two phases. First, using its current knowledge about mathematics and physics, the system verifies that the solution is valid. At the end of this phase the system has determined that the human's solution solves the current problem, but it does not have any understanding of *why* the user chose these steps to solve the problem. During the second phase of understanding, **Physics 101** determines a reason for the structure of each expression in the user's solution. Especially important is understanding new formulae encountered in the solution. After this phase the system has an understanding of how and why this solution solved the problem at hand. At this point it is able to profitably generalize any new principles that were used in the solution process, thereby increasing its knowledge of classical physics.



**Figure 1:** The Operation of Physics 101

The current implementation of the model learns the physical concept

of momentum conservation. More detailed descriptions of this process can be found in [331] and [332]. In this example, conservation of momentum is used - by a human - to solve a one-dimensional, two-body collision problem that the system could not solve. (In this problem no external forces act upon the two objects.) The human's solution is verified and then generalized, during which the system discovers that momentum is only conserved in the absence of any net external force. A general equation describing how external forces affect momentum is then derived by the system. Equation 1 presents the result obtained.

$$\sum_{i=1}^N \text{mass}_i \text{velocity}_i = \int \sum_{i=1}^N \text{force}_{\text{external},i} dt \quad (1)$$

This formula says: *The total momentum of a collection of objects is determined by the integral of the sum of the external forces on those objects.* A second problem, which involves three bodies under the influence of an external force, has been solved by **Physics 101** using this generalized result.

One interesting issue encountered in the collision problem involves the generalization from a specific situation containing two objects to the general situation, which can contain *any* number of interacting objects. The system analyzes the human's solution and detects that summing the two objects' momenta eliminates from the calculation the force each object exerts upon the other, regardless of the details of these forces. (This is a consequence of Newton's third law, which says that every action has an equal and opposite reaction.) Since each object in a physical situation potentially exerts a force on every other object, in the general case cancelling the net inter-object force upon an object requires summing the momenta of *all* the objects. This is the line of reasoning **Physics 101** follows when generalizing the human's solution technique.

Much research on learning involves relaxing constraints on the entities in a situation, rather than generalizing the number of entities themselves. Nonetheless, many important concepts require generalizing number. Explanation-based learning provides a solution to a major problem, namely, how do you know when it is valid and proper to generalize the number of entities? For example, compare the concept of *tripod* with *bicycle wheel*. Both concepts contain a number of repeated components. Suppose a three-legged tripod and a 25-spoked wheel are observed. An explanation-based system can build a general concept for each. The general tripod concept will contain precisely three legs, as any other number of legs is unstable. The general wheel concept, however, will allow a variable number

of spokes. The explanation of a component's functionality dictates when it is valid and proper to generalize its number.

## CONCLUSION

By analyzing a worked example, the current implementation of **Physics 101** is able to derive a formula describing the temporal evolution of the momentum of any arbitrary collection of objects. This formula can be used to solve a collection of complicated collision problems. Other physical concepts to be learned include work, friction, conservation of energy, simple harmonic motion, and conservation of angular momentum. As these additional concepts are learned, previously-learned concepts will have to be refined. This work will also investigate how the system can learn to estimate which features of a problem can be *ignored* when solving the problem, an important trait possessed by experts.

Others [124, 197, 202] have investigated the learning of physical *concepts*. Larkin's **ABLE** system learns how to *algebraically* apply principles of physics. Langley's work involves the analysis of multiple data measurements and does not utilize deep knowledge about the physical world. Forbus and Gentner consider learning *qualitative* physics by analogical reasoning. Unlike any of these systems, **Physics 101** reasons with calculus to explore the implications of Newton's laws.

This research contributes to the theory of machine learning, and should prove applicable to knowledge-acquisition in expert systems. From a psychological perspective, it demonstrates the computational consistency of a mechanism that may underlie human learning in a complicated domain. This work also has implications for intelligent computer-aided instruction, in that it advances a learning model for a complex domain involving both symbolic and numerical reasoning.

## ACKNOWLEDGMENTS

This work was supported in part by the Air Force Office of Scientific Research under grant F49620-82-K-0009 and in part by the National Science Foundation under grant NSF-IST-83-17889.

# VIEWS AND CAUSALITY IN DISCOVERY: MODELLING HUMAN INDUCTION

Jeff Shrager<sup>1</sup>

Carnegie-Mellon University, Dept. of Psychology  
Pitts., PA, 15213

## ABSTRACT

Our recent efforts have involved modelling "instructionless learning". This is a sort of induction that human learners exhibit when they are asked to "figure out" complex systems (e.g., programmable toys) without instructions or advice. Instructionless learners often undertake major reorganizations of their theories but may not, in the process, lose the syntax and semantics of particular parts of their theories. How do these reorganizations take place? They do not think very carefully about changes in their theories and therefore rely heavily upon the analysis of errors. How does error recovery work?

Our computer model is composed of several theory change mechanisms, an experiment constructor, and causal reasoner. The principal theory change mechanism is "view application" which merges abstract schemas into the current theory in a process similar to script inference. View application provides the framework in which all other mechanisms work. It may result in large changes in part of the current schema while leaving specific contents intact. "Discovery" results from successive merging of views into the current theory.

The causal reasoner is invoked when errors take place in the prediction of (confirmatory) experimental results. The reasoner first tries to interpret the observations in the current theoretical framework (assigning blame and then patching the theory). If this fails, it tries to interpret the observed (unpredicted) behavior in terms of a new view (calling on view application to merge this new view into the current theory). Blame assignment relies on a "mental model" of the target. The content of this model is learned by view application.

We think that these mechanisms (and others embodied in the model) underlie instructionless learning (natural induction), and are principle mechanisms in scientific reasoning.

---

<sup>1</sup>The author's current address is: c/o Xerox PARC, 3333 Coyote Hill Rd., Palo Alto, CA 94304.

## MODELLING INSTRUCTIONLESS LEARNING

Our interest in the intuitions which underlie scientific reasoning led to the study of "instructionless learning". In this task, subjects are asked to "figure out" a BigTrak programmable toy without instructions or advice. They form hypotheses about the device and run explorations and experiments to refine and confirm their hypotheses. The findings from this psychological work are presented in Shrager & Klahr (1983 and in preparation) and Shrager (1984 and 1985).

IE (the Instructionless Experimenter) is a learning model which can simulate subjects learning about the BigTrak. IE relies upon several independent learning mechanisms to change the current schema (which represents the learner's current theory of the target). The mechanisms communicate by setting goals in response to various conditions. IE focuses on learning problem solving operators for the target device, although it also acquires "model" knowledge in the course of learning.

The various theory change mechanisms in the program are: operator analogy, rote association of inputs with behaviors of the device, the instantiation of operator types into particular operators, a process called "view application" (which I shall describe shortly), the generalization of numeric terms into variables, and search.

## VIEW APPLICATION PROVIDES LEARNING FRAMEWORKS

These independent learning mechanisms work within a framework which defines particular primitives and their interpretations, and the interpretation of particular knowledge structures (e.g., operators). We call these abstract frameworks "views". In contrast to typical discovery learners, IE is not bound to a particular framework. When failure explanation mechanisms cannot interpret observations within the current framework, they may force a shift in the framework of analysis by calling for the application of a new view. New learning thus takes place as a result of the combining of views throughout the learning session.

Views are similar to scripts, and the view application mechanism is similar to the mechanisms of script application [318, 319]. However, views represent objects in a structural theory rather than sequences of events. Views and theories in IE contain operators for problem solving in the domain, objects that constitute the terms of reference and their structure, and various physical features of the domain. Thus they are specifically suited to reasoning about complex systems. Although the view application heuristics are not specifically bound to this representation, the goals that

the learner uses to direct experiment construction, etc revolve around the operators in the representation.

## LEARNING MECHANISM MAKE HEURISTIC DECISIONS

One result of our psychological work is the observation that human learners do not take into account contradictory evidence in making learning decisions, and do not very carefully perform experiments (e.g., they almost always perform confounded confirmatory experiments). They do not always slowly accumulate information which constitutes the theory but sometimes prefer to completely reorganize far-reaching aspects of their current knowledge about BigTrak.

We believe that this is a result of the high rate of interaction and low cost of error in this task. For instance, we would not expect this to be true of people instructionlessly learning to run a real air-traffic control system. Nonetheless, our subjects succeed in learning about the somewhat complex BigTrak in about half an hour, performing about 50 experiments during that period. How can a learning engine which doesn't carefully seek optimal theory changes succeed in so short a time?

## IMPORTANT LEARNING TAKES PLACE AT POINTS OF FAILURE

IE performs various sorts of experiments with the BigTrak (via a simulator). Like CD learning models, but unlike most search models, the events which are most important to learning take place when experimental predictions *fail*. IE never backtracks to a previous theory. When a failure occurs, the observed behavior of the target device is analyzed in order to learn *positively* from the failure. This may take two general forms: within theory credit (blame) assignment, or extratheoretical learning by activating the view applier to combine in a new view. These are discussed in the next two sections.

## THE CAUSAL REASONER USES THE MODEL TO ASSIGN BLAME

The first thing that IE does in the face of an experiment failure is to try to interpret the observed behavior in terms of the current theory. In order to assign blame for the error, and appropriately patch the theory to account for the observed behavior, IE relies upon the learner's device

model. This contains objects such as memories, switches in various states, and internal operators. Internal operators are called upon by the processing of other (external or internal) operators in order to accomplish action in the device model. Thus, for instance, IE's understanding of the function of pressing a particular key on the device may be that it applies several internal operators in order to update the memory in a particular way. For instance, if I press the CLR (clear) button, a loop is activated which erases the actions stored in each of the cells of the program memory.

Blame assignment works by backchaining through the operators applied during the present (failed) interaction until one is found which can be understood as having lead to the observed behavior. For instance, suppose that the present interaction is that I press "CLR RIGHTARROW 1 GO" and expect the BigTrak to move to the right one foot. In fact, it instead turns to the right 6 degrees. We begin by asking whether pressing GO could have caused this behavior. The current understanding of what GO does is that it runs the contents of a memory. Therefore, the causal reasoner knows to look, instead, for some operator that set that memory. It looks next at the previous RIGHTARROW-n operator. This operator's function indicates that it sets the memory to a particular action (believed to be a rightward movement for n-feet). The reasoner is able to interpret the right turn as a possible new function for this operator, and so changes the expected function of that operator to account for the new observation.

## **FAILURE IN BLAME ASSIGNMENT MAY LEAD TO NEW VIEW APPLICATIONS**

If the causal reasoner is not successful at assigning blame for the observed behavior, it tries to interpret the observed behavior in terms of a known view. Views are indexed by their function (and by other means). A search takes place for a view that could account for the observed behavior of the device. If one is found, then view application is invoked to merge the selected view into the current theory. Selection takes place without regard for the current contents of the device theory. However, view application may or may not preserve some of the learner's current knowledge about the device, depending upon how much is specified in the view.

## **WHY DOES IE SUCCEED?**

We asked how the learner is able to discover an essentially correct theory of this somewhat complex programmable device, with relatively few

experiments and given the limitations of somewhat shallow decision making in learning. The answer is that (a) the BigTrak is "reactive" -- experiments do not simply yield a correct/incorrect signal. The action of the device provides some information which can be analyzed in order to learn. And, (b) the BigTrak has an underlying model which can be understood as some "simple" combination of a small amount of the learner's prior knowledge (as stored in views).

## DISCUSSION AND GOALS

The IE model makes a strong claim that all learning takes place within frameworks defined by views and that these views change as a result of the analysis of failures. (There are other means of view activation.) Views help to fill a gap in the theory of discovery programs like BACON [194, 197]: Where does the structure of the search space come from? IE also begins to address the question of the use of structural models (i.e., "mental models" in the case of devices) in learning and discovery. They help perform appropriate blame assignment in cases of error. None of these concepts is particularly new individually -- view application is like script inference, and LEX (and other systems) performs theory-based credit assignment. However, we have shown how they can be combined into a working learning engine and have provided evidence to suggest that this is an appropriate model of human learning in some cases.

Our current goals are to work toward more reasoned discovery within this framework. That is, to use the structural theory (in the form of a device model or structural model of, for instance, some physical system) to both do reasoned blame assignment and view indexing and application. We hope that this, combined with better models of the other learning mechanisms mentioned here, will enable us to construct a learner that is able to use structural theories to discover in more complex scientific domains.

## ACKNOWLEDGMENTS

David Klahr has contributed extensively to this research. We also thank John Anderson and Herbert Simon for their advice.

# LEARNING CONTROL INFORMATION

Bernard Silver

GTE Laboratories Inc  
40 Sylvan Rd, Waltham MA 02254

## ABSTRACT

Precondition Analysis is a technique for learning control information from a single example. Unlike many other analytic learning techniques, it can be used in domains where operator inversion is difficult or impossible. Precondition Analysis has been implemented in the domain of equation solving. The author is currently extending this work in various directions.

## INTRODUCTION

Precondition Analysis is a technique for learning control information from a single example, [340, 339, 341]. Using the terminology of Mitchell, [253], it is a mainly analytic technique. However, unlike many other analytic approaches, such as LEX2, [249], it is capable of working in domains that are ill-behaved in various ways. For example, the back propagation used by LEX2 requires that the problem-solving operators involved are *reversible*. (Operators are reversible if it is possible to reconstruct state *A*, given an operator *O* and state *B*, produced by *O* from *A*.) This requirement is not always met. The operators may also be ill-behaved in other ways. In the equation solving domain used by the author, the preconditions of the operators are necessary conditions but not sufficient. This leads to additional difficulties in learning control information.

## THE LP PROGRAM

Precondition Analysis has been implemented in LP [340, 341], an equation solving program, built on PRESS, (book,meta), that learns new ways to solve equations from worked examples. The equations used are symbolic, transcendental and non-differential, such as

$$\cos(x) + 2 \cdot \cos(2 \cdot x) + \cos(3 \cdot x) = 0$$

LP is presented with a worked example that is simply a list of states, beginning with the original equation and ending with the solution. LP uses three types of object:

1. **Rewrite Rules:** Facts of Algebra.
2. **Methods:** Operators that apply the **rewrite rules**. They contain control information that suggests how and when the rules should be applied.
3. **Schemas:** A plan containing **methods**, that indicates reasons for applying each method. They are used as an inexact plan, executed in a flexible way.

LP is capable of learning methods and schemas from the worked example. It is also able to detect the use of a rewrite rule that it does not know. In such cases, LP prompts the user with a specific instance of the rule and asks the user for the general rule.

The newly learned methods and schemas, and the newly acquired rewrite rules increase the ability of LP to solve new equations. It should obviously be able to solve the equation presented in the worked example, but the new structures should be applicable in other circumstances as well.

The schemas are perhaps the most interesting part of LP. A schema is basically a list of the methods that were used in a particular worked example. In addition, the schema also indicates the "reasons" for the application of each method. Basically, LP considers that a method is applied in order to satisfy the some preconditions of a following method, with the final method solving the equation. Thus the reasons recorded in the schema are just conditions. The reasons for each method are divided into two parts; those conditions that are already satisfied (and which the method application will preserve), and the new conditions that will become satisfied because of the method application.

The schemas are used when LP is given a new equation to solve. In simple terms, the program would like to apply exactly the same sequence of methods as was used in the worked example. This usually does not work, and the program arrives at a point where it cannot apply the method that comes next in the schema. At this point, the program examines the reasons for the application of that method, and attempts to "patch" the schema. For example, it might find that it cannot apply method *M*. Method *M* was used in the worked example to satisfy conditions *C* so that the next method *N* could apply. LP looks through the methods it knows, and may find that another method, *M'* is also known to sometimes be able to produce the satisfaction of *C*. In this case, LP

may try to apply  $M'$  instead of  $M$ . With luck, this may produce the desired effect and  $N$  can then be applied. In such a case, LP has patched the schema by substituting  $M'$  for  $M$ .

Generally, the problem is more complex, for many reasons. The analysis in the schema is inexact; LP has a limited vocabulary for expressing conditions, and it may be that a certain method was used in a worked example for a variety of reasons, including some that LP cannot express. In such a case, LP may be able to substitute one method for another, but then find that it is unable to apply the following method. Also, the fact that the preconditions of a method are only necessary and not sufficient means that even if all the preconditions of a method have been satisfied by previous steps, the method may still not apply.

## RESEARCH DIRECTIONS

LP has been quite successful in learning new equation solving techniques. However, an experiment reported in [341] demonstrated that the time taken to solve an equation (or to fail to solve it) increases significantly as the program learns more schemas. The problem is that LP tries very hard to use all the schemas that *might* be applicable. If in fact the equation cannot be solved by LP, the program spends a considerable time discovering this. This is not so undesirable. More importantly, LP will sometimes try to use an inappropriate schema first, before choosing a more useful one. This tends leads to an increase in average solution time as learning progresses. (Of course, the number of problems that can be solved increases as well.) One of the directions of the current research is to make the schema selection process cleverer.

Minton is also interested in the problem caused by learning "too much". His MORRIS program, [242], tackles the problem earlier by not storing all learned macros. This is a somewhat orthogonal approach to improving the schema selection process.

The other major research direction is to evaluate Precondition Analysis in other domains. In particular, the author is interested in discovering the extent to which the domain influences the utility of the various schema patching processes. For example, under certain circumstances LP is allowed to omit an operator. This may turn out to be a very poor idea in other domains.

# AN INVESTIGATION OF THE NATURE OF MATHEMATICAL DISCOVERY

Michael H. Sims

Departments of Mathematics and Computer Science  
Rutgers University, New Brunswick, NJ 08903

## ABSTRACT

Our research has indicated that in some domains of mathematics a purely empirical approach is insufficient to model the discovery process. In this paper we discuss an analytic technique which can be coupled usefully with empirical techniques of discovery.

## RESEARCH PROBLEM

One area of my dissertation research is the **interplay between empirical and analytical techniques in mathematical discovery**. I am developing a computer program named IL (for Imre Lakatos [192]) to investigate discovery in mathematics. In this paper I will describe some current efforts in proof transformation within IL.

Previous investigations in applying AI to mathematical discovery have explored empirical and analytic techniques independently. The major piece of research is Doug Lenat's AM program [83]. Considering the success of AM it is surprising that it wasn't followed by other mathematical discovery systems<sup>1</sup>. AM used what have been called empirical techniques. By *empirical* we mean that the reasoning was driven by *examples*, rather than by more *deductive* or *analytic* reasoning.

A mathematician attempting to discover a new fact might begin by investigating a simple example of some property, and then verifying that it is valid. One might then consider a more general case and attempt to extend the verification of the simpler case to the new problem. As part of this process one may discover important new restrictions which weren't evident in the simpler problem. The importance of this research is that by combining known techniques in a novel way there is a synergy which can

---

<sup>1</sup>Lenat did investigate mathematical discovery further as an application of his Eurisko system, but that particular application has not had a major impact on our understanding of the issues.

lead to the discovery of new theorems. In the following example, we will make more explicit this reasoning process in a five step process leading to a discovery.

## EXAMPLE OF A THEOREM DISCOVERY

Let's consider a plausible discovery of a theorem about Conway numbers, my primary domain of application. First we present a little background [80]. Conway numbers are recursively defined objects with Left and Right sets, i.e.,  $[L|R]$ , where L and R are sets of Conway numbers. The Conway numbers which correspond to 0,1,-1, in the usual number system are  $0 = [\{\}|\{\}]$ ,  $1 = [\{0\}|\{\}]$ , and  $-1 = [\{\}|0\}]$ . This means that the right set of 1 is the empty set and the left set is the set with one element, namely 0. Now consider the recursively defined relationship of  $\leq$ .

**Definition 1:** We say that  $x \leq y$  iff (no  $y' \leq x$  and  $y \leq$  no  $x'$ ).  
[See Figure 1]



**Figure 1:**  $x$  and  $y$ , with typical elements of their left and right sets.

Note that  $y = [\{\dots,y',\dots\}|\{\dots,y',\dots\}]$  and  $x = [\{\dots,x',\dots\}|\{\dots,x',\dots\}]$ . Intuitively the idea of the definition is that all the  $y'$ 's are to the right of  $x$ , and that all the  $x'$ 's are to the left of  $y$ .

I can now state the theorem that I would like my system to discover:  
**Theorem:**  $-1 \leq y$ , where  $y = [\{\dots,y',\dots\}|\{\}]$ . How might this realistically be discovered? Consider a five step approach:

- (a) Discover property or fact (*In our case: Discover conjecture  $-1 \leq 1$* )
- (b) Verify property (i.e., *Prove  $-1 \leq 1$* )
- (c) Heuristically modify property (a) (i.e., *Generalize  $-1 \leq 1$  to  $-1 \leq y$  by syntactic mutation*)

(d) Verify more general property (c) (i.e., *Try to prove*  $-1 \leq y$ )

(e) Modify property (c), guided by any failure in attempted verification (d) (i.e., *Proof in (d) will require adding the restriction*  $y = [\{...y'\,...\}|\{\}\}]$ )

Steps (a) and (c) can be assumed to be similar to the processing of AM and will not be discussed in this paper. We are next going to discuss the proof (b), which can reasonably be taken as the output of a theorem prover subsystem. We will then discuss our implementation of (d) and (e).

**Theorem 1:**  $-1 \leq 1$

This theorem seems obvious, but for Conway numbers there is a little that needs to be verified. In particular we must use the definitions of  $\leq$ , 1 and  $-1$ .

**Proof:** In the first three lines we will simply restate what we mean by the theorem.

(1)  $-1 \leq 1$  means that:

(2)  $[\{\}| \{0\}] \leq [\{0\}| \{\}]$ . Clarifying what we mean by the inequality we have:

(3)  $x \leq y$  iff (no  $y' \leq x$  and  $y \leq$  no  $x'$ ). Here  $x = -1 = [\{\}| \{0\}]$ , and  $y = 1 = [\{0\}| \{\}]$ . (Or equivalently,  $y' \in \{0\}$ ,  $y' \in \{\}$ ,  $x' \in \{\}$ , and  $x' \in \{0\}$ .)

(4) We now use the (trivial) fact that if the right set of  $y$  is empty, there is no  $y'$  such that  $y' \leq x$  (since there is no  $y'$  at all). We conclude one half of what we need to conclude --i.e., that no  $y' \leq x$ .

(5) Similarly we can conclude  $y \leq$  no  $x'$ .

(6) Hence,  $-1 \leq 1$ .

This proof only required the expansion of definitions, their instantiations, and the application of one rule. As a mathematician one realizes that the crux of this proof is that if the right set of  $y$  is empty and the left set of  $x$  is empty, then  $x \leq y$ . We want IL to do a similar analysis. One strategy would be to generalize the theorem and see if the

proof still goes through (or can be patched up). For example, we might incrementally generalize each of the terms in the theorem's statement<sup>2</sup>. In the above theorem we can generalize one or more of  $-1, 1$ , or  $\leq$ . We might arbitrarily generalize just the  $1$ . Leading to:

**Conjecture 1:**  $-1 \leq y$ ,  $y$  is a number. This is not quite true, but it will get patched up in the process of the proof generalization. The conjecture is the result of an exploration along the *is-a* dimension (step (c) above). IL will use the proof to Theorem 1 to help find the proof of this conjecture. Replaying the proof we have:

**Proof:**  $y$  is a number so  $y = [\{..., y', ...\} | \{..., y', ...\}]$ .

So Step (2) of the above proof becomes  
 $[\{\} | \{0\}] \leq [\{..., y', ...\} | \{..., y', ...\}]$ .

(3)  $x \leq y$  iff ( $\text{no } y' \leq x$  and  $y \leq \text{no } x'$ ). Here  $x = -1 = [\{\} | \{0\}]$ . The second part, " $y \leq \text{no } x'$ ", is okay since still there is no  $x'$ .

On the other hand, " $\text{no } y' \leq x$ " doesn't follow through. Using the reasoning of the first proof it would be true if there were no  $y'$ , in other words if  $y$ 's right set was empty. This follows from examining the precondition to the rule used in step (4). So modifying the theorem's statement to incorporate the constraint we have:

**Theorem 2:**  $-1 \leq y$ , where  $y = [\{..., y', ...\} | \{\}]$ .

The generalization procedure can now proceed to consider further generalizations. We might also want to consider specializations of some other part of the theorem.

In the current Prolog version of IL, this proof transformation procedure has been partially implemented. This procedure allows for the generalization of and-or solution trees, and has been tested on generalizing a Tower of Hanoi solution, as well as the application of this paper. Each step in our proof can be thought of as an operator which takes a domain-state into an range-state, with the final range state being the given

<sup>2</sup>One might use information about the intended use or goal to direct which parts or in what order to generalize. Alternatively, it might be appropriate to find a maximal generalization of the theorem consistent with the proof, or to trade off specializing one part of the proof statement for generalizing another part.

theorem. The proof transformation process consists of choosing the form of our theorem (usually a generalization) and then back propagating that state through the proof operators.<sup>3</sup> Having back propagated through the entire tree, we gather up the constraints which have been generated and appropriately constrain the statement of our theorem. The back propagation or goal regression is mathematically the process of finding preimages to various sets [269, 371].

## RELATED WORK AND PLANS

In related work on mathematics, Bledsoe [26] has explored how an example can be helpful in guiding a search for a proof, and Bundy [49] has investigated how the analysis and correction of a faulty proof can lead to the invention of new terms. IL's procedure of proof generalization is however more closely related to the analytical techniques used by Utgoff, DeJong and others [371, 86, 221, 243, 249]. IL uses a relatively standard application of goal regression, except that IL's choice of an operator to back propagate over can depend on the *goal concept* [249, 166] of the problem. IL allows constraint intersection after each operator, and we aren't necessarily trying to do maximal generalization. Most other applications of goal regression have generalized their operators simply by variabilizing constants. We have shown an application of these goal regression techniques to a theorem discovery and its proof as part of our five step discovery process. This process is heuristic; there are no guarantees that for a given related expression the proof transformation will carry through. We have, however, demonstrated that this is a viable approach for certain problems.

We plan to continue with the implementation of our five step discovery process as well as demonstrate its viability in several case studies. This work represents part of the research for an interdisciplinary dissertation on mathematical reasoning under the direction of S. Amarel.

## ACKNOWLEDGMENTS

My thanks to J. Mostow, S. Amarel, S. Kedar-Cabelli, P. Spool and P. Riddle for suggesting improvements on an earlier draft. This research

<sup>3</sup>Additionally we allow arbitrary constraints to be imposed on (i.e., intersected with) the *derived* constraints after each back propagation through an operator. This is to facilitate the treatment of zero-sum games.

has been supported in part by the National Science Foundation under grant DCS83-51523, and in part by the Rutgers Laboratory for Computer Science Research.

# **LEARNING HOW TO REACH A GOAL: A STRATEGY FOR THE MULTIPLE CLASSES CLASSIFICATION PROBLEM**

**Henri Soldano and Hélène Pigot**

Institut Curie, laboratoire Curie  
11 rue P & M. Curie 75005 Paris, France

## **INTRODUCTION**

We are interested in the task of Knowledge acquisition in addressing real problems as the interpretation of complex objects (speech recognition, scene analysis, etc . . . ).

Solving such problems consists in managing a set of goals, each of them having to be reached in executing a set of deductive processes. These processes are built either using empirical and theoretical knowledge or acquiring knowledge through examples and background knowledge. The main points are:

- How to find a structure determining the whole process?
- How to reach the goals?

We focus here on the second point considered as a learning task.

Let **U** be the set of possible objects and **C** be the set of concepts relevant to the addressed problem. A concept **C** is then a subset of **U**. A set of examples of **C** is a subset of **U** included in **C**. A concept may be considered as a subset as well as a logical proposition.

Let us define the following task: learning how to reach a goal **B** which is expressed as a set of concepts and a set of relations between these concepts, provided that we have a set of examples for each concept, and background knowledge.

We propose the following scheme:

1. Applying two kinds of elementary learning from examples mechanisms:
  - single concept learning

- discrimination learning, i.e. searching discriminant features between two concepts.
2. The result of a learning mechanism is a “deductive process” which is able to give a partial answer about any object of  $\mathbf{U}$ , starting with the description of this object.
  3. Finding a strategy in order to organize this set of deductive processes in order to reach the goal.

In an actual application we are particularly concerned with the confidence which we may have about the whole process, and thus about each executed deductive process, when an unknown object is tested. This leads to the following notions:

- *Evaluation mechanisms*: a deductive process is valid if it verifies an evaluation criteria, i.e. if it makes few errors [345].
- *Silence*: a deductive process gives an answer or not. The only way to insure a high confidence in a deductive process is to allow him to answer “I don’t know”.

The goal-learning task for which we propose a strategy is the multiple classes classification problem: an object has to be classified in one class belonging to an exhaustive set of  $n$  classes, i.e.:

$$\mathbf{A}_1 \cup \mathbf{A}_2 \cup \dots \cup \mathbf{A}_n = \mathbf{U} \quad (1)$$

$$\forall i, j \quad \mathbf{A}_i \cap \mathbf{A}_j = \emptyset \quad (2)$$

For simplicity purpose we will not discuss here the more general classification problem in which the classes may overlap (i.e. the condition (2) is dropped).

We have checked this strategy in the following speech recognition problem: a vowel, described by a power spectrum, has to be labelled with one of the twelve vowel labels of the french phonetic system [276]. We have used statistico-logic methods as learning mechanisms. These methods, which use symbolic and numeric features, are particularly efficient for such pattern recognition problems [283, 355].

## ELEMENTARY LEARNING MECHANISMS

Let  $E_A, E_B, \dots$  be respectively sets of examples of the concepts  $A, B, \dots$  and  $\bar{A}$  the complementary set of  $A$  in  $U$  (which is equivalent to the negation  $\neg A$  of  $A$ ).

### Single Concept Learning

In order to learn  $A$  we use  $E_A$  (examples of  $A$ ) and  $E_{\bar{A}}$  (examples of  $\bar{A}$  i.e. counter-examples of  $A$ ). Thus we use knowledge about  $A \cup \bar{A} = U$ .

$(E_A, E_{\bar{A}})$  leads to build a deductive process  $d_A(u)$  such that:

- if  $u$  is an object of  $U$ , then the answer  $d_A(u)$  is one of the following:
  - Refutation of A:  $u$  does not verify  $A$ .
  - Silence about A: no possible answer with respect to  $A$ .
  - Assertion of A:  $u$  verifies  $A$ .

### Discrimination Learning

In order to build a deductive process related to the discrimination between  $A$  and  $B$ , we use  $E_A$  and  $E_B$ . Since we use knowledge only about  $A \cup B$ , a discrimination learning mechanism uses partial knowledge about  $U$ .

$(E_A, E_B)$  is used to build a deductive process  $d_{A/B}(u)$  such that:

- if  $u$  is an object of  $U$ , then the answer  $d_{A/B}(u)$  is one of the following:
  - Refutation of A:  $u$  does not verify  $A$ , with respect to the acquired knowledge about  $A$  and  $B$ .
  - Refutation of B:  $u$  does not verify  $B$ .
  - Silence

One can find a simple reason why the process gives refutation answers: usually, we do not know if an object  $u$  belongs or not to  $A \cup$

B; thus two situations may occur:

- either  $u$  belongs to  $A \cup B$ , then  $d_{A/B}$  is relevant since it has been built using knowledge about  $A$  and  $B$ , and refuting  $A$  is equivalent to assert  $B$ .
- or  $u$  does not belong to  $A \cup B$ , then the process is not relevant. However, whatever the answer of the process is, it can't be wrong.

This means that the process may be executed even if we have no a priori information about the object.

Let us consider a simple example.  $U$  is a set of vehicles split up into four classes (bicycles, cars, trucks, planes). The following discriminant process  $d_{C/B}$  dealing with cars and bicycles has been built:

- If  $u$  has two wheels then it is *not a car*.
- If  $u$  has a steering-wheel then it is *not a bicycle*.

Let  $b, c, t, p$  be respectively a bicycle, a car, a truck, a plane. Applying  $d_{C/B}$  we obtain the following answers:

- $b$  is *not a car*
- $c$  is *not a bicycle*
- $t$  is *not a bicycle*
- *Silence* about  $p$

Let us note that, in the third case, although the discriminant process was theoretically not relevant, it gave useful information about the vehicle  $t$ .

A more formal justification of giving refutation answers is obtained considering the following definition of a discrimination learning mechanism: finding a concept representation  $f$ , such that:

- all examples of  $A$  verify  $f$ , i.e.  $E_A \Rightarrow f$
- no examples of  $B$  verify  $f$ , i.e.  $E_B \Rightarrow \neg f$

Then, whatever the used learning method is, we assume that  $f$  is such that:

$$\begin{array}{ll} A \Rightarrow f & (1) \\ B \Rightarrow \neg f & (2) \end{array}$$

In order to apply  $f$  to unknown objects we use the equivalent forms for (1) and (2):

$$\begin{array}{ll} \neg f \Rightarrow \neg A & \\ f \Rightarrow \neg B & \end{array}$$

i.e., if  $f$  is false, then  $A$  is refuted, if  $f$  is true, then  $B$  is refuted.

## **A STRATEGY FOR THE MULTIPLE CLASSES CLASSIFICATION PROBLEM**

During the learning step we build the  $n$  deductive processes  $d_{Ai}$  corresponding to the  $n$  classes, and the  $n \times (n-1)/2$  discriminant processes corresponding to the couples of classes  $(A_i, A_j)$ .

Let us examine now how to use the processes when an unknown object is provided and has to be tested.

### **Using Single Concept Learning**

We start with applying the  $n$  processes  $d_{Ai}$  to the unknown object to be tested. The result is that some classes are refuted and few or none of them are asserted. In practice the assertions are often not reliable enough, thus we use an elimination strategy: The list  $Nref$  of non-refuted classes is established.

### **Using Discrimination Learning**

The second part of the elimination strategy consists in applying the  $p \times (p-1)$  discriminant processes related to the  $p$  classes belonging to  $Nref$ . Each of these processes uses knowledge about two classes and is likely to refute one of them.

For each class  $A$  of  $Nref$  the results of the  $(p-1)$  deductive processes likely to refute  $A$  are observed. If the class is refuted by one or more processes, then it is eliminated. When the  $p$  classes are checked this way, we obtain a more restricted list of non-refuted classes.

If the list is empty the tested object is not classified. If the list contains one or more classes we have a high confidence in the fact that the correct class belongs to the list.

## Some Remarks

We have been applying this strategy to the vowel recognition problem mentioned above. This led to the following remarks:

- If, testing an object, we still have several possible classes at the end of the whole process, then it is useful to state an order on the non-refuted classes. This may be done using the strength of each assertion or refutation when such information is available.
- When managing a lot of deductive processes the occurrence of errors is unavoidable. This leads to the use of recovering mechanisms as the following: "a class of Nref is actually refuted, only if it is refuted by several discriminant processes".
- The description of a complex object needs a large quantity of information. But each deductive process to be applied (or learned) only needs part of this information. It is then useful to have a specific description for each of the deductive processes, especially if they rely on logic-based description.

# **CONCEPTUAL CLUSTERING OF STRUCTURED OBJECTS**

**R. E. Stepp**

Coordinated Science Laboratory  
The University of Illinois at Urbana-Champaign  
1101 W. Springfield Avenue, Urbana, Illinois 61801

## **ABSTRACT**

A summary of this author's current research on conceptual clustering of structured objects is presented along with some avenues for future investigation.

## **INTRODUCTION**

The observed physical world contains interconnected and interrelated objects. Some objects may be constituent parts of larger objects or systems of objects, or they themselves may be systems composed of subparts, or both. Suitable representation schemes for describing structured objects include networks or expressions in predicate calculus. Such representations can be manipulated by programs and thus provide an ability to perform automated conceptual clustering (including concept formation, inductive inference, generalization, etc.) involving both the characteristics of parts of objects and the interrelationships between parts.

Clustering over characteristics of parts is done by discovering useful relationships between multivalued part descriptor functions (e.g., "color of part-1") and a subset of function values (e.g., "red or blue"). Automated attribute-based conceptual clustering has been the goal of several programs including CLUSTER/2 [236], DISCON [199], RUMMAGE [120], and some related programs such as GLAUBER [200].

Clustering over interrelationships between parts of objects is a more difficult task. Some progress in this direction has been made in the program CLUSTER/S [362]. A fundamental difficulty in attempting to find conceptual clusters in a collection of structured objects stems from the lack of a finite representation space spanning all possible object descriptions. For example, an intelligent program may find a conceptually pleasing way to partition objects into those that "have a blue triangle on top of a red square" and those that "have a red part on top of a green part." For

certain collections of objects, this conceptual clustering can unambiguously divide the collection into distinct classes. The difficulty lies in the potential for objects that match both descriptions simultaneously, such as an object with three parts: a blue triangle on a red square on a green rectangle.

The two class descriptions above are not mutually disjoint because their membership sets have a non-empty intersection. When the entities within the intersection are some of the given objects (given objects will be called *observations*) the intersection is termed *strong*. When the entities within the intersection are unobserved hypothetical objects, the intersection is termed *weak*. Thus, a pair of descriptions is either *disjoint*, *weakly intersecting*, or *strongly intersecting*.

## CURRENT RESEARCH

Current research involves developing better algorithms for performing conceptual clustering on structured objects for two distinct types of tasks: building a concept hierarchy to describe a collection of structured objects, and building a concept hierarchy to describe object substructure.

### **Building a concept hierarchy to describe a collection of structured objects.**

Two methodologies for performing conceptual clustering of structured objects were proposed by Michalski and Stepp [237]. The two methods are: a data-driven approach that employs repeated discrimination (repeated applications of a *learning from examples* paradigm), and a model-driven approach that uses background knowledge to find and evaluate good classifying attributes. An implementation of the method based on repeated discrimination is incorporated in the program CLUSTER/S [362].

CLUSTER/S has two strategies for clustering structured objects. One is an adaptation of the conceptual clustering algorithm used in CLUSTER/2. That algorithm is briefly: perform the following steps for several values for K (the number of clusters to form).

1. select K seed objects;
2. form a conceptual cluster surrounding each seed;
3. evaluate the clustering according to a given goal (user-specified evaluation function);

4. repeat (1) through (3) until the hill climbing appears stalled (until a given number of iterations yield no upward advance towards the goal as measured by the evaluation function).

As demonstrated by CLUSTER/S, this approach will partition the collection of objects into conceptual classes and provide a conjunctive-form description for each class. The class descriptions are either disjoint weakly intersecting.

The other strategy in CLUSTER/S decomposes the structured object clustering problem into two parts: generating simplified unstructured attribute-based descriptions, and then performing attribute-based conceptual clustering on the generated attributes using the mechanisms of CLUSTER/2. The approach is to build a maximally specific characteristic description for the entire collection of objects. This description is generalized just to the extent that it covers all objects while retaining as many object part interrelationships as possible. This covering characteristic description serves as a template to identify corresponding parts of individual objects and permits a template-ordered sequence of attribute values to be extracted from each object description. The derived attribute-based object descriptions are processed by CLUSTER/2. The benefit is that the resulting class descriptions are disjoint over the universe of objects that match the template description. The disadvantage is that only those object part interrelationships and part attributes found in the template description can be used to form the cluster descriptions. For further discussion see Ref. [362].

#### **Building a concept hierarchy to describe object substructure.**

An interesting but different type of conceptual clustering problem involves trying to discover a decomposition of the internal structure of one object or common units of substructure prevalent in a collection of objects. For example, given a few organic chemical molecules viewed using the simplistic "ball and stick" model, an interesting internal structure might be an aromatic ring of carbon and hydrogen atoms. This substructure occurs one or more times in a variety of molecules. A general characteristic of substructure units and a reasonable heuristic for identifying candidate units is that they be made of highly interconnected parts with few connections to other substructures. Another general characteristic for good substructures is that they join together and with other substructures to form the whole object.

An ability for recursive conceptual data analysis can be brought into the picture. With a mechanism for retaining useful concepts and inferences from one conceptual clustering application to another, it is possible to

discover potentially relevant internal substructures in a few selected examples and then utilize the discovered substructure definitions to simplify the other object descriptions (by decomposing the objects into substructure units). Research in this area is just beginning.

## FUTURE DIRECTIONS

People perform conceptual clustering with relative ease. Studies (e.g., Ref. [226]) indicate that people may prefer classifications based on simple conjunctive concepts. It also appears likely that people utilize a large amount of general and problem-specific background knowledge to help build relevant categories.

New research is aimed at improving the current capabilities of CLUSTER/2 and CLUSTER/S for applying background knowledge for the synthesis of candidate class descriptions. The background knowledge consists of a network of goals of the classification, inference rules and heuristics for deriving new object descriptors, definitions of descriptor value sets and types, and the criterion for evaluating the goodness of one candidate classification versus another. This avenue of research builds on the model-driven approach outlined in [237] and [362].

## ACKNOWLEDGMENTS

This work is supported in part by the Air Force Office of Scientific Research under grant F49620-82-K-0009 and by the National Science Foundation under grant NSF DCR 84-06801.

# LEARNING IN INTRACTABLE DOMAINS

## Prasad V. Tadepalli

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

### ABSTRACT

The domain knowledge used in Explanation Based Learning to explain and generalize examples sometimes gives rise to prohibitively complex and verbose explanations. This problem is most apparent in domains like chess. Here we illustrate the importance of recognizing the goal structure of the players to learn useful and tractable heuristics in chess.

### INTRODUCTION

Explanation Based Learning (EBL) has received considerable attention in recent years. In this paradigm, in addition to the training examples, the learning program is also given some *domain theory* which is used to constrain the possible generalizations to those consistent with the theory. Learning involves explaining an example in terms of the domain theory and then generalizing it so that the same explanation still holds [255]. The generalization is usually done by a technique called *goal regression* [269] which computes sufficient conditions for the same explanation to hold. In some domains like chess, programming and circuit design, it is often too complex to explain and generalize any non-trivial example. Even when the generalization is possible, the generalized expression is too complex to evaluate during the problem solving. This problem is called *the intractable theory problem* by Mitchell et al. [255]. In this research, we are exploring this problem in the domain of king and pawn endgames in chess.<sup>1</sup> We motivate the need for explicating the goal structure in the move sequence to generate meaningful and tractable, though approximate, generalizations. We identify some of the open problems in this approach and suggest future directions to pursue.

---

<sup>1</sup>This report describes research in progress but not an implemented system.

## RELATED WORK

EBL has been successfully used in different domains by several people [119, 86, 243, 255]. Pitrat [278] and Minton [243] use similar techniques to learn winning plans in chess. Pitrat uses domain specific heuristics to simplify and generalize the chess position. Minton circumvents the complexity problem to some extent by learning only plans in which all the moves of the system are *forced* - i.e. the system has only one valid move at each intermediate position.

## THE PROBLEM: Intractability

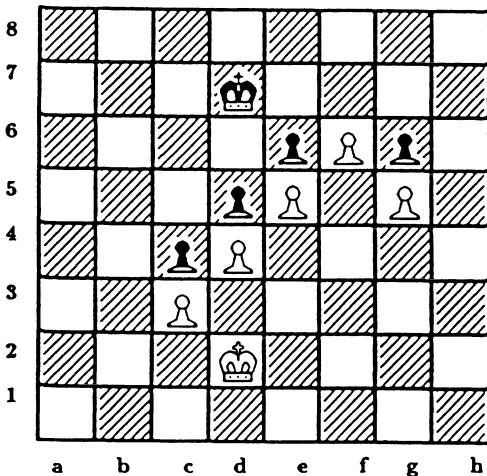
In two person games, an adequate explanation of a strategy involves showing that all the threats and defenses of the opponent are going to be met. Typically, since the opponent has several possible moves (say around 10) in each position, the explanation involves steps that increase exponentially in complexity with the length of the solution sequence. We adapted the *goal regression* technique to the two person games and worked it out by hand on a simple example. We were able to induce the *square of the pawn law*<sup>2</sup> from the example. Unfortunately, even in this simple case, we encountered huge expressions of predicate calculus, and had to do many simplifications and make some default assumptions (e.g. ignore the possibility of your own king getting into check while advancing the pawn to promote it). Unlike some domains like robot planning, the default assumptions in chess are very context sensitive [395]; i.e. the validity of the assumption depends very much on the particular board position. The problem needs to be approached in a different way to be generalizable to other complex examples.

## SOLUTION: Explicate the Goal Structure

Let us motivate the new approach by an example from [19]. Assume that the program is given the problem and its solution in Figure 1.

In addition to exponential complexity, there are two other important reasons why the standard goal regression method is not a useful one to follow. One reason is that there may not be anything *conceptually similar* in the thousands of positions that happen to require the same sequence of

<sup>2</sup>Square of the pawn law defines the board locations, in which the opponent's king could be so that he can take the pawn before it promotes - i.e. reaches the last rank.



1. WK(d2→c2), BK(d7→e8)
2. WK(c2→b2), BK(e8→d7)
3. WK(b2→a3), BK(d7→e8)
4. WK(a3→b4), BK(e8→d7)
5. WK(b4→c5), BK(d7→e8)
6. WK(c5→d6), BK(e8→f7)
7. WK(d6→d7), BK(f7→f8)
8. WK(d7→e6)XP, BK(f8→e8)
9. WK(e6→d6), BK(e8→f7)
10. WP(e5→e6), ...

**Figure 1:** A Problem and its Partial Solution

operators to achieve the same goals. On the other hand, two different move sequences may in fact be used to achieve the same goals. Their similarity should be made apparent by describing rules at a higher level of abstraction. A human player may begin to understand the above example in the following way.

White king moved from d2 to... d7 to e6 and took the pawn at e6. Apparently, White's goal was to take the pawn at e6. If I were playing White, I would follow the shortest route. But then, in this case, there is an obstruction to that path. White's path seems to be the shortest permissible. Meanwhile, Black king should stay in the vicinity of the White pawn at f6, so that it can not be promoted. But why did White take the pawn at e6? To advance the White pawn at e5? ...

The point is that people understand long action sequences, not directly in terms of each individual action and its effects on the following action, but by inferring the plausible high level goal structure of the agent [317, 354]. In order to produce meaningful and useful generalizations from an example, it is necessary to interpret the example at the proper level of abstraction. In our case, it involves interpreting the move sequence to make the goal structure of the players explicit. For example, the goal structure of White for the above problem contains, in addition to other goals, the

decomposition of *promote-pawn* goal to *remove-blockades* and *advance-pawn* goals. The *remove-blockades* goal is further decomposed into *approach-the-blockade* and *take-the-blockade*. Goals are thus decomposed until the level of individual moves. By inferring the goal structure of the players, the learning program can:

- Decompose the problem into several independent subproblems and learn the relevant pieces of knowledge separately. In the example, the plans for *approach-the-blockade* and *advance-pawn* can be independently learned. This is achieved by doing goal regression independently on each of the subproblems. One of the advantages of doing this is that the plans for individual subproblems can be combined in novel ways during subsequent problem solving.
- Learn rules involving high level goals and actions which stand for apparently dissimilar action sequences at the lower level. In the above problem, one might learn a rule like "*If a pawn is not protected by any other pawn, then attack it*". This rule is applicable independent of the sequence of moves one may have to play to attack and take the pawn. The high level rules may be obtained by doing goal regression in an abstraction space.
- Facilitate knowledge integration by recognizing known subgoals and plans and learning only when a new goal is achieved or a goal is achieved in a novel way. In the example, if the system already knows how to achieve the *approach-the-blockade* goal, it can ignore that part of the game and focus on the remaining part. This takes the mystery away from the first six moves of White.

The price for these apparent benefits is that the rules the system learns are not guaranteed to work always. The heuristic nature of the knowledge involved in identifying the goal structure and decomposing a problem and the unexpected interactions of subproblems are some of the sources of uncertainty. Thus we still need to do some search before applying the rules to confirm their validity [243, 395].

## FUTURE WORK

I am working on an implementation of a system that demonstrates the validity of the above ideas. One major necessary condition to the

success of the above approach in chess has to do with the representation of high level concepts like "nearest path", "barricade" etc.

## ACKNOWLEDGMENTS

Many of the ideas presented here have evolved during numerous discussions with Smadar Kedar-Cabelli, Rich Keller, Sridhar Mahadevan, Tom Mitchell, Armand Prieditis and Keith Williamson. My sincere thanks go to all of them. I also thank Smadar Kedar-Cabelli, Sridhar Mahadevan, Tom Mitchell and Jack Mostow for comments on an earlier draft.

# ON COMPIILING EXPLAINABLE MODELS OF A DESIGN DOMAIN

## Christopher Tong

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

### ABSTRACT

The purpose of this paper is to draw from our work on design<sup>1</sup> (implemented in a program called DONTE), to tentatively lay out some implications for the compilation of explainable design domain models. Using a simple example, we develop a framework for viewing the incremental compilation of domain models that rely progressively less on *search* and more on *compiled foresight*.

### A SIMPLE EXAMPLE

A novice designer's initial specifications call for design of a Stack, capable of holding 50 elements, each element at most 3 bits "wide"; the stack is to be realized using TTL technology, and must use at most 200 gates. An expert designer examines the novice designer's initial specifications and immediately states that they are unrealistic. When asked why, he explains "Each of the 50 Stack elements requires 3 bits; each of the bits can be realized using a TTL flip-flop; a minimal-gate flip-flop (a latch) requires at least 2 gates. Thus your design will require at least  $50 \times 3 \times 2$ , or 300 gates."

### RELATED IDEAS

1. *Search is a useful metaphor for the design process.* In DONTE, design problems like the one in the above example are solved using a top-down refinement process; this process begins with the designer's abstract, functional specification for a Stack, and incrementally elaborates the specification until it is transparently realizable in the target technology, TTL

---

<sup>1</sup>Tong, "Knowledge-based circuit design", forthcoming PhD thesis, Stanford University.

(e.g. the specification is in terms of gates). This incremental refinement process may be viewed as search in a space of partial specifications; from a given partial specification, more complete specifications - its *refinement alternatives* - can be generated. Decomposition and implementation are the "macro" steps in this design space - an abstract Stack component is decomposed into 50 element-holding components; each element-holding component is further decomposed into 3 bit-holding components; finally, each bit-holding component is implemented as a TTL flip-flop.

*2. To facilitate a model of design as search, specifications are usefully and operationally distinguished as functional specifications or goals.* Circuits are examples of functional artifacts, that is, artifacts designed for particular purposes. One of the nice properties of many functional design domains (e.g. the circuit design domain) is that often a space of partial functional specifications can be described economically by using a *library* of abstract or concrete *component classes* and *component interconnection classes*, and indexing functional refinement or implementation alternatives off of these class definitions. Search then proceeds via incremental generate-and-test. The *Generator* creates refinement alternatives for instances of library classes by creating instances of other library classes, as directed by the original instances' library class definitions. The *Tester* then selects one or more alternatives whose continued refinement is worth exploring.

Not all of the designer's initial specifications can be easily expressed as library class instances. In our example, that the Stack be capable of storing 50 elements is representable by making the Stack specification refer to an instance of a "List data structure" class, with a "maximum length" parameter set to 50. However, that the Stack be realized using at most 200 gates is not easily expressible in this fashion; in the generate-and-test model of design, such "left-over" specifications are used by the Tester to help select among refinement alternatives. We will refer to these later kinds of specifications as *goals*.

*3. Evaluation consists of determining whether what has been specified is realizable. Explainable evaluation demonstrates how specifications can (or cannot) be realized.* The Tester uses goals as tests by trying to evaluate them against the partial specifications. Evaluation can be performed by inspection, compiled foresight, or search.

For example, when the Stack specification has been completely refined, and implemented as a description of 150 flip-flops (each composed of 2 gates), the goal, "The stack must be realized using at most 200 gates", can be evaluated by *inspection*, and recognized as being unsatisfied

by this particular implementation choice. When the novice designer initially lays out his specifications, he may not know whether his "gate-limiting" goal is satisfiable, and may need to *search* the design space to create implementations that can be evaluated by inspection. In contrast, when the expert designer views the novice's initial specifications, he immediately applies *compiled foresight* in the form of an *evaluation function*:

$$\# \text{ elements in Stack} \times \# \text{ bits per element} \times \# \text{ gates per bit} \quad (\text{EF})$$

to arrive at a lower bound on the number of gates, and an immediate assessment that the goal as stated is unrealistic; satisfying the goal is a possible recommendation.

Evaluation by inspection is explainable by definition. Evaluation by search is indirectly explainable, in that it ultimately produces nodes that are evaluable by inspection. A particular, explainable form of compiled foresight, the *evaluation function*, maps abstract metrics into concrete metrics (e.g. number of elements and bits/element into number of gates); an explanation for such an evaluation function can be created if the evaluation function can be viewed as a compiled *qualitative simulation* of the design process. For example, the function EF is actually derivable from a sequence of known generic refinement steps:

Memories are decomposable into element-holding components.

Element-holding components are decomposable into  
bit-holding components.

Bit components are implementable as flip-flops. (EJ)

Saving such explanations along with the evaluation functions they explain provides a means for describing *how* the goals may (or may not) be satisfied; in other words, EJ constitutes an abstract *design plan* for refining a memory.

*4. A goal for learning research in the context of design is to create a learning cycle that incrementally compiles an illcharted domain model (yielding a design process that relies heavily on search) into one whose associated design process is based mostly on use of explainable, compiled foresight. In our search metaphor, a model of a functional design domain consists of a space of partial functional specifications, a set of goal types (e.g. constraints on the number of gates), and a (possibly incomplete) method for evaluating nodes in the design space with respect to particular types of goals. Design domain models vary in their *chartedness*, the strength of their knowledge connecting what is specifiable with what is realizable. *Generative chartedness* is the strength of the knowledge for*

recognizing whether a particular partial specification ultimately can be refined into a complete specification (and providing guidance for how to do it). *Evaluative chartedness* is the strength of the knowledge for testing partial specifications against goals. The discussion in this paper presumes the domain is currently generatively well-charted, but evaluatively ill-charted; the learning cycle would attempt to improve the evaluative chartedness of the model by converting past experience into explainable compiled foresight, used in the future to avoid search. This compilation process does not broaden the class of solvable design problems, but rather increases the efficiency of the design process for the currently solvable problems.

**5. Evaluation functions can be constructed by:** creating problem-specific, explanation-level design plans using a qualitative domain model; justifying them in that specific context using a complete domain model; generalizing them; and reformulating them as evaluation functions. As a bridge between an inefficient design process based purely on search, and one that relies more heavily on compiled foresight (e.g. evaluation functions), DONTE makes use of a *qualitative design space* that allows it to plan in terms of abstract and possibly incomplete refinement steps. For example, the qualitative space would allow steps involving refinement of abstract functionality (a Stack is a Memory; the refinement of Memories in general entails expanding the Memory's data structure - e.g. a list - into components holding constituent data structures - e.g. the list's elements). In contrast, a complete model would insist that all the specific characteristics of a Stack be accounted for in refining a Stack. We conjecture that qualitative design plans make better explanations than complete design plans.

DONTE exploits any flexibility in the possible order of execution of its design steps by using a *nonlinear design plan* as a data structure. Design steps are distinguished by *purpose*; some are purely for the purpose of functional refinement, while others have the additional purpose of impacting a goal in some way. Thus, the complete plan for refining the Stack contains many more steps than just abstract memory expansion steps (list memory => element memory => bit memory => flip-flops). However, the memory expansion steps: (1) have the greatest impact on the "number of gates" goal; (2) can be executed fairly independently of other steps, as indicated by the nonlinear design plan; and (3) are sufficient for demonstrating the unrealistic nature of the goal.

DONTE's design process consists of a dialectic between a Designer and a Critic. The Designer makes a move in the design space. The Critic creates a design plan that suggests whether the goals can still be satisfied, and (abstractly) how they can (or cannot) be satisfied, via moves in the

qualitative design space. Such a projected plan constitutes a *plausibly explainable* evaluation, and is used to focus the attention of the Designer. The Designer eventually verifies or falsifies this evaluation, based on its movement in the *complete* design space.

As a next step in our research, we are considering how the results of performing qualitative and detailed search can be compiled into explainable evaluation functions. If the Critic's design plan indicating moves in the qualitative design space is verified by the Designer, that design plan constitutes an evaluation that has been justified in the context of the specific design problem.

Design plans appear to be an appropriate intermediate representation for operations we would like to standardly perform on compiled foresight. A design plan can be *generalized* by using such standard techniques as turning constants into variables (100 elements into n elements), or by exploiting an existing functional component hierarchy (e.g. a plan that has been justified for a Stack may be justifiable for the more general Memory). Also, design plans can be *merged* or *modified* somewhat more easily than can evaluation functions.

To avoid search in making evaluations, we would like to reformulate *process-oriented* design plans (such as EJ) into *state-oriented* evaluation functions (such as EF). Reformulation may not always be simple; however, we can see in our example that the multiplication of parameters in EF is a direct consequence of the occurrence of repeated components in the decomposition hierarchy produced by applying EJ. We are currently looking for similarly general mappings for reformulating design plans into evaluation functions.

# WHAT CAN BE LEARNED?

## L.G. Valiant

Aiken Computation Laboratory, Harvard University  
Cambridge, MA 02138

Both human and animal learning often exhibit fast rates of convergence. From relatively few examples an individual can induce the concept that is exemplified. Furthermore distinct individuals appear to converge to essentially the same concept even when they have learned from different sets of examples.

This phenomenon is commonplace and experimentally reproducible (e.g. [145]). Surely it must be amenable to scientific explanation. The emergence of computational complexity theory as a discipline offers hope in this quest. By ruling out hypotheses that are computationally intractable it should help to prune down the search space of possible learning mechanisms and thus increase our chances of discovering the true ones.

The old philosophical problem of induction does have to be faced. In any nontrivial setting inducing a concept with certainty from a few examples is logically impossible. The following more relaxed notion of induction does appear to us to be not only a plausible model of the real world phenomenon but also a workable one [374, 375, 376]. Suppose that examples and counterexamples of a concept occur in the real world with time invariant distributions  $D^+$  and  $D^-$  respectively.  $D^+$  defines the relative frequency with which the exponentially large or infinite variety of examples of the concept occur in nature.  $D^-$  defines the counterexamples. Learning or induction from examples consists of observing sets of identified examples and of identified counterexamples drawn randomly from these two distributions and deducing from these an algorithm  $f$  which has, with large probability, the following property. If a new example is drawn randomly from either  $D^+$  or  $D^-$  then  $f$  will correctly distinguish with large probability whether this was  $D^+$  or  $D^-$ .

The above notion of learning allows for two sources of error, both of which appear to be plausible and necessary. First the training set may be so atypical that the resulting deduction has no validity. Second, even if the training set was typical, the new example may be atypical enough that the training examples did not prepare for it. The probability of both sources of error can be reduced by increasing the number of training examples.

A learning strategy cannot be evaluated by trying it on a single concept. For no rigorous distinction can be made between whether the

concept is being learned or whether it has been preprogrammed into the strategy. Hence we have to discuss the behavior of a learning algorithm on whole classes of concepts.

The above observations suggest the following methodology for investigating learning: choose a style  $S$  of knowledge representation and find the largest class  $X$  of concepts expressible in this style that can be learned feasibly. For simplicity we are assuming passive learning protocols for now [374].

The most interesting knowledge representation  $S$  for which we have positive results is that of propositional expressions in disjunctive normal form or DNF (i.e. disjunctions of conjunctions). If the maximum length  $k$  of conjunctions that occur is fixed then these can be learned in the above sense in time polynomial in the number of propositional predicates and the size of the expression to be learned. The result holds for all distributions  $D^+$  and  $D^-$ . The simplest elimination algorithm suffices. Furthermore the algorithm requires no knowledge of the distributions.

The class of learnable DNF expressions can be extended if we are satisfied with learning for restricted distributions. A favourable case occurs when we are guaranteed that a universe of conjunctions that contains the ones occurring in the expression can be generated in polynomial time from a random sample of positive examples. One such method of generation is to observe which pairs, triples, etc., of predicates occur together with high correlation and choosing these conjunctions as the universe. With respect to any such method of generation one can define classes of concept-distribution pairs  $(f, D^+)$  that are learnable [376].

It is reasonable to ask whether our learning algorithms are suited to implementation on a connectionist model [1, 117]. It turns out that for reasonable classes of  $D^+$  this is possible for disjunctions of conjunctions where the length of conjunctions is bounded. A characteristic of the particular method that emerges is that the conjunctions are learned in unsupervised mode while the final disjunction is done in supervised mode.

Once we have a precise model of learning and some positive results numerous further questions can be asked. How robust is the algorithm to errors in the data? Can the learning be made more effective by having a more active learner? In what directions can the knowledge representation be extended without sacrificing computational tractability? For what representations is learning provably intractable? To some of these we have partial answers.

Although we do consider DNF to be an interesting representation richer or alternative styles may turn out to be even more interesting. It so happens, however, that most examples of successful concept learning to date are expressible as DNF expressions with short conjunctions (e.g.

[232, 327]). Furthermore it is striking that if we extend this class even slightly then learning appears to require exponential time. For example, no polynomial time algorithm is known for learning unrestricted DNF expressions either in our particular probabilistic sense, or in any other alternative plausible sense.

# **LEARNING HEURISTIC RULES FROM DEEP REASONING**

**Walter Van De Velde**

AI-Lab., Vrije Universiteit Brussel  
Pleinlaan 2, B-1050 Brussels, Belgium

## **INTRODUCTION**

The heuristic knowledge an expert has about his domain of expertise may be seen as a compilation of his past expertise. The expert has to fall back on his deep understanding of the problem-domain if his heuristic knowledge is inadequate to solve a new problem. A new heuristic may be learned from this experience and the old heuristics have to be adapted to incorporate it in a concise way.

This summary introduces research which aims at realising such a scenario within expert systems [359]. Current technology in knowledge representation allows for the expression of heuristic and deep knowledge in a single system. Methods are available for 'shallow' (e.g.. heuristic) and 'deep' reasoning. The combination of both, reveals a whole landscape of opportunities for discovering heuristics.

## **EFFICIENT HEURISTICS**

The ultimate goal of learning from deep reasoning is to start from a deep (e.g. causal) model of a problem-domain, and gradually 'compile' it into an efficient set of heuristics. Our approach is to let this process be driven by the events that actually occur in the everyday usage of the expert system. Thus, only relevant rules will be learned. It is conjectured that these are only a small subset of all those that are theoretically possible. This is probably true for diagnosis which we will use as an example of an application.

Different kinds of rules are needed in an efficient set of heuristics. There should be solution-rules which associate symptoms with solutions. There should be focus-rules which pinpoint the problem in some particular context (e.g.. component, function or problem-kind): each context has a set of relevant rules associated with it and focus-rules lead from one context to another. Moreover the number of questions to the user should be small and there may not be excessive double-checking of properties.

## LEARNING THROUGH PROGRESSIVE REFINEMENT

The basic techniques for rule-learning are the initial abstraction of a rule from a deep reasoning result, and its integration in the set of previously learned rules.

### Rule Abstraction

A deep reasoning process potentially investigates a lot of causal paths and asks for a lot of observations. To be worthwhile, heuristic rules must do the opposite: they should be as simple as possible, leaving out details and observations which did not directly contribute to the solution of the problem. Therefore, heuristic rules are learned under the following circumscriptive assumption: everything which is not mentioned in a rule is supposed to be normal or irrelevant to the anomaly to be explained.

From a deep reasoning sequence, a heuristic rule is learned which associates the anomaly to be explained (its 'primary-symptom') with the causes which reasoning deemed ultimately responsible (its 'corrected-items'). This rule may be tried whenever its primary-symptom is to be explained (and only then!).

For example, if a car does not start and deep reasoning determines that this was due to it not being in parking mode, then the initial rule extracted from this situation looks as follows:

Rule for not in parking

A Solution-Rule

Primary-symptoms: Not Car=Starts

Corrected-properties: Mode=Parking

### Rule Integration

The goal of rule integration is to add symptoms to the rules ('secondary-symptoms') so that they become mutually exclusive. Symptoms which can be added to a rule are the anomalies corresponding to its corrected-items, or symptoms known to be relevant (because they correspond to a corrected-item of another rule with the same primary-symptom) but not corresponding to a corrected-item of the rule (and thus were certainly normal in the situation the rule was initially learned from).

For example, suppose a new rule is abstracted which explains the malfunctioning of the car by an empty gastank, i.e.

Rule for no gas

A Solution-Rule

Primary-symptoms: Not Car=Starts

Secondary-symptoms: -

Corrected-properties: Gas=Present

then the rules will be adapted as follows:

Rule for not in parking

A Solution-Rule

Primary-symptoms: Not Car=Starts

Secondary-symptoms: Gas=Present

Corrected-properties: Mode=Parking

Rule for no gas

A Solution-Rule

Primary-symptoms: Not Car=Starts

Secondary-symptoms: Not Gas=Present

Corrected-properties: Gas=Present

The process of rule-integration is monotonic. A rule never becomes invalid though its applicability may be restricted. It should be stressed that this works well because of the circumscriptive way a rule should be read. The method is therefore not applicable to expert-provided heuristics because there is no control over the way such rules are to be read.

### **Strategic Integration**

The degree of freedom in the integration process can be exploited by using more knowledge about the domain. For instance, if a good way is found to associate costs with observations (dynamically), then learning techniques may use this knowledge to generate better rules.

Another optimisation would be to make rule order significant, allowing the incremental checking of symptoms.

### **OTHER LEARNING MECHANISMS**

Bases on similarities between rules or contexts, other learning mechanisms can augment the power of learned rules.

## **Learning through Generalisation**

The goal of learning through generalisation is to decouple learned rules from the particular context for which they were learned, i.e. to put them in more than one context. Rules may thus be applied to similar regions of the same or another device. A more sophisticated process of adaptation may take local variations into account.

For example, if a rule is learned which explains a malfunctioning sparkplug by a bad cable then this may be applied to all sparkplugs and their corresponding cable. Generalization requires sophisticated knowledge representation techniques.

## **Learning through Rule Combination**

Rules may try to explain different symptoms by the same corrected-items. The simultaneous occurrence of these symptoms points strongly to those corrected-items. So, these rules may be combined to a single one which forces to check the other symptoms as soon as one of them is observed. This mechanism turns out to be capable of generating strong rules and moreover, the rule-set shrinks. (Rule-combination should sometimes be prevented if costs are associated with observations).

Rule combinations accounts for learning heuristics such as checking whether the lights can burn to eliminate the possibility of a flat battery as explanation for a non-starting engine.

## **Learning New Concepts**

Whenever a bunch of symptoms has to be checked over and over again for different rules, it is more efficient to establish them as an intermediate conclusion. Criteria will have to be found to decide when it is interesting to introduce a new concept for such a conclusion.

## **Learning of Focus Rules**

The goal of learning focus rules is the division of rules in smaller sets according to the problem-type, component or function they relate to, and the learning of rules which lead from one set to another. This division may be based on another kind of model. For instance, focusing along the part-whole relation is learned by identifying in a topological model the part which is investigated by a set of rules.

## CONCLUSIONS

This summary introduced mechanisms to learn heuristics from deep reasoning. Some of them are implemented (abstraction, integration, combination and simple generalisation). Others require more research but seem feasible with current technology of knowledge representation.

Emerging from these experiments is the insight that a richer rule structure (e.g. distinction between primary and secondary symptoms) is important for rule learning. For example, without this structure which controls rule activation, the process of rule-combination would not be possible. This research may provide a better understanding of the semantics of heuristics.

For a more thorough description and extensive examples of the techniques presented here, as well as for comparison and references to other work on learning, see [360].

# **LEARNING A DOMAIN THEORY BY COMPLETING EXPLANATIONS**

**Kurt VanLehn**

Xerox Parc, Intelligent Systems Laboratory  
Palo Alto, CA. 94304

## **ABSTRACT**

There are many systems that learn by creating an explanation for some phenomenon. They draw upon a domain theory in order to generate the explanation. One approach to acquiring that domain theory is to analyze phenomena that *can't* be explained using the current domain theory. Any piece of knowledge that would complete the explanation is a candidate for addition to the domain theory. By examining several phenomena, the pieces of knowledge that have the most explanatory power can be located and added to the domain theory. A system based on this learning technique has successfully learned certain basic mathematical skills.

## **INTRODUCTION**

As the present volume testifies, a surprisingly large number of machine learning projects are investigating explanation-based learning. The shared feature of these investigations is that the learner comes equipped with a rich source of knowledge, often called the domain theory, that is used to analyze or explain some particular experience that the learner either creates or is given. For instance, the learner is given a particular story about kidnapping and explains it in terms of buying, selling, love, and other domain theoretical concepts. From the resulting explanation, the learner acquires new knowledge about kidnapping [87] [Mooney (this volume)]. Another illustration is Shavlik's work on physics (this volume). The learner is given the solution to a mechanics problems, and derives an explanation (proof of correctness) in terms of its current, incomplete knowledge of mechanics. From the explanation, it abstracts new principles of physics, e.g., conservation of momentum.

Unfortunately, there is some variation in how the term "explanation-based learning" is used. For instance, Mitchell et al. (this volume) seem to use it to refer to the kind of analytical generalization that is performed by LEX2 and its ilk (a technique often called back-propagation). The kind

of explanation-based learning considered here is characterized by hierarchical explanations of flat, non-hierarchical examples. The kidnapping story, for instance, has a non-hierarchical, almost linear structure. It is a translation of the linear text of the story into a more-or-less linear semantic representation. The explanation consists of grouping parts of the linear structure into composite structures (e.g., monetary exchanges) and finding relationships (e.g., causality) among them. The explanation, then, is a tangled hierarchy of composite structures, whose leaves are the parts of the example. In addition, there may be lateral relationships among some of the nodes in the hierarchy.

A critical issue in explanation-based learning is accounting for the acquisition of the domain theory that is used to formulate the explanations. Some explanation-based learning systems, as Tom Dietterich pointed out in his talk at the workshop, do not acquire new knowledge but rather transform their knowledge into a more efficient form. Such systems increase their performance, but their competence (i.e., their performance given infinite computational resources) remains at the same high level. The following comments describe a technique for learning new domain knowledge, that is, for augmenting the domain theory's competence.

## **EXPLANATION-BASED LEARNING OF ARITHMETIC PROCEDURES**

My research has concerned the psychology of skill acquisition [378, 377, 43]. In order to simplify the collection and analysis of experimental data, the research has been carried out with simple skills: ordinary, multicolumn arithmetic procedures. A technique was discovered for a type of learning that could be called *learning by completing explanations*. Although this is probably not how students learn arithmetic, it is how Sierra, the program that generates the theory's predictions, learns arithmetic [379].

In arithmetic, the procedure plays the role of the domain theory. It is a "theory" of action sequences. An action is a visible event, such as writing a digit or crossing one out. The procedure/theory can "explain" an action sequence by simply parsing it, using the procedure as if it were a grammar, but a grammar with data flow, conditional tests, etc. The explanation is a hierarchical trace (parse tree) erected over the action sequence. A partial explanation for an individual action in a sequence can be read off the parse tree by walking upwards from the action: e.g., action A was done in order to satisfy the goals of its caller (which is the next node above it in the trace tree), and the caller, which is a subprocedure,

was executed in order to satisfy its caller, and so on. A complete explanation for an action involves taking into account data flows and side-effects, so explicit links for these effects are added to the parse tree. Such links are analogous to the causal links that thread through the hierarchical structures of explanations of, e.g., kidnapping stories.

Learning a procedure corresponds to learning a domain theory. In Sierra, this learning proceeds by trying to explain/parse an action sequence. A simple kind of learning takes place when subprocedures must be generalized in order to complete a parse. For instance, if the subprocedure has an applicability condition that must be violated in order to parse the example, then the condition is generalized. This simple form of learning seems to be analogous to certain kinds of explanation-based learning (c.f., DeJong, this volume).

## LEARNING BY COMPLETING EXPLANATIONS

A more interesting kind of learning takes place when it is impossible to complete a parse, no matter how much the subprocedures are generalized. In this case, the learner's stock of subprocedures is fundamentally incomplete, and one or more new subprocedures must be invented. Solving this task is called learning by completing explanations. It is one way to incrementally acquire a domain theory.

There is, as far as I can see, only one feasible approach to learning by completing explanations. It is the approach followed in my work as well as two independent investigations [128, 350]. The approach is to parse the action sequence bottom-up as far as possible and top-down as far as possible. Any new subprocedure (or set of new subprocedures) that links the incomplete top-down parse to the incomplete bottom-up parse in such a way that a complete parse is yielded is a candidate solution to the learning task. Even for short action sequences, there can be millions of candidates. The challenge is to cope with this combinatorial space of candidates.

The solution used by Genesereth and Smith is to place such strong domain-specific constraints on the parsing that only one (or a few) of the possible candidates are generated. My solution is to (1) use unconstrained parsing, (2) assume that only one new subprocedure will be acquired, and (3) use a chart (sometimes called a well-formed substring table) to efficiently represent the space of possible candidates. This approach to learning by completing explanations makes it simple to perform induction across several action sequences. Essentially, each action sequence yields a space of candidate solutions, represented as a chart. Induction amounts to

intersection of these spaces, which can be efficiently implemented with a chart intersection algorithm.

Presumably, this technique can be used in any domain which learns hierarchical knowledge structures from sequential examples. Thus, it should extend to learning grammars from strings, learning story understanding schemata from stories, learning physics from exemplary solutions of physics problems, and perhaps other learning tasks as well.

From another viewpoint, this technique implements a kind of failure-driven learning. As Roger Schank pointed out in his talk at the workshop, it is plausible that people seek out and learn new ideas whenever they can't explain a phenomenon using their current knowledge. The ideas they seek are precisely those that will complete an explanation for the puzzling phenomenon. Sierra is a demonstration of the power of this technique in a particularly simple domain. It remains to be seen how it will fare in more complex ones.

## ACKNOWLEDGMENTS

This research was supported by Personnel and Training Research Programs, Psychological Sciences Division, the Office of Naval Research, under contract number N00014-82C-0067 and contract authority number NR667-477.

# **LEARNING IMPLEMENTATION RULES WITH OPERATING-CONDITIONS DEPENDING ON INTERNAL STRUCTURES IN VLSI DESIGN**

**Masanobu Watanabe**

C&C Systems Research Laboratories, NEC Corporation  
4-1-1 Miyazaki, Miyamae-ku, Kawasaki, 213 Japan

## **ABSTRACT**

The problem of expertise acquisition by observing user's solutions in the Learning Apprentice Systems [254] is here considered as learning implementation rules in VLSI circuit design. The representation and learning method of Operating-Conditions, depending on internal structures, in the Left-Hand-Side of the implementation rules are proposed.

## **MOTIVATION AND RESEARCH PROBLEM**

A Learning Apprentice System (LEAP) monitors user's solutions on circuit implementations in an interactive mode, and then generalizes and determines implementation rules by analyzing single training examples. The Left-Hand-Side of this implementation rule consists of both Operating-Condition, which assures that a well-defined output will be produced, and IO-Mapping (functional specification) to be implemented. In order to assure the generalized IO-Mapping derived by Verification-Based Learning [221], LEAP must determine the complete Operating-Condition. Otherwise, LEAP fails to formulate the implementation rule at an appropriate level of generality.

In some cases, it is difficult and unsuitable to describe the complete Operating-Condition by requirements on just input signals. For instance, let us consider a training example shown in Fig. 1 and an inferred rule shown in Fig. 2. In Fig. 1 both "Function to be implemented" and "Where Input Signals Satisfy" mean a module specification in terms of its IO-Mapping and Operating-Conditions, respectively. "User's Solution" means that the system has observed that the user decomposed the original module with the above specification by using both Pass-1 and Pass-Net-1.

Figure 2 shows an implementation rule with complete Operating-Condition which should be inferred from the training example shown in Fig. 1. It should be noted that its Operating-Condition ("Where Input Signals

Function to be implemented :

```

Inputs : Input1, Input2, Input3
Outputs : Output
Function : (Equals (Value Output (i))
                  (if (And (Equals (Value Input1 (i)) High)
                           (Not (Equals (Value Input2 (i))
                                         (Value Input3 (i)))))
                  Then Low
                  Else Tristated))

```

Where Input Signals Satisfy :

```

(Equals (Voltagelevel Input1 (i)) Switchable-High-Level)
(Equals (Voltagelevel Input2 (i)) Restoring-Logic-Level)
(Equals (Voltagelevel Input3 (i)) Restoring-Logic-Level)

```

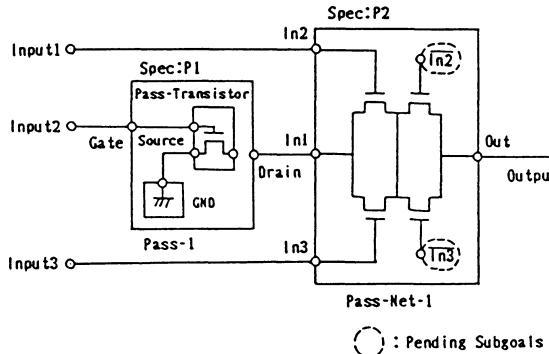
User's Solution :

Figure 1. A Training Example

Satisfy") is described by input signals (\*any\*) conditioned by the output signals (<any1> and <any2>).

If this Operating-Condition shown in Fig. 2 is described by requirements on only input signals, the Operating-Condition of input signals (\*any\*) can be derived as "(Equals (Voltagelevel \*any-input\*) Switchable-High-Level)", because of both a fact that Source port of Pass-Transistor within Pass-1 is connected to GND, and a fact that Operating-Condition for Gate of Pass-Transistor is "(Equals (Voltagelevel Gate) (IF (Equals (Voltagelevel Source) Passing-Low-Level) Then Switchable-High-Level Else Restoring-Logic-Level))". This leads us to an overgeneralized rule shown in Fig. 3(b).

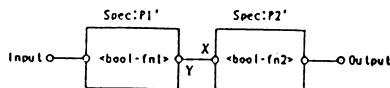
The reason, why this rule shown in Fig. 3(b) is overgeneralized, is that this rule is applicable to a module specification shown in Fig. 3(a). Even though the module specification should be eventually implemented by using Pull-Up-Networks, which have VDD as a voltage-source and produce

If the function to be implemented is of the form:

```
Inputs : <anyt>
Outputs : Output
Function : (Equals (Value Output (1))
  (If (And <bool-fn1> <bool-fn2>)
    Then <any1> Else <any2>))
```

Where Input Signals Satisfy:  
 $\text{(Equals (Voltagelevel } \text{Input1}) \text{ anyt)}$   
 $\text{ (If (Equals (Voltagelevel } \text{any1})$   
 $\text{ (Voltagelevel } \text{any2})$   
 $\text{ Passing-Low-Level)}$   
 $\text{ Then Switchable-High-Level}$   
 $\text{ Else Restoring-Logic-Level))}$

THEN one possible implementation is:



With Specifications of the two modules as follow:  
P1': (Equals (Value Y (1))  
 (If <bool-fn1> Then <any1> Else <any2>))  
P2': (Equals (Value Output (1))  
 (If <bool-fn2> Then (Value X (1)) Else <any2>))

Figure 2. Inferred Rule

Function to be implemented:

```
Inputs : Input1, Input2, Input3
Outputs : Output
Function : (Equals (Value Output (1))
  (If (And (Equals (Value Input1 (1)) High)
    (Not (Equals (Value Input2 (1))
      (Value Input3 (1)))))
    Then High Else Tristated))
```

Where Input Signals Satisfy:

(Equals (Voltagelevel Input1 (1)) Switchable-High-Level)  
(Equals (Voltagelevel Input2 (1)) Switchable-High-Level)  
(Equals (Voltagelevel Input3 (1)) Switchable-High-Level)

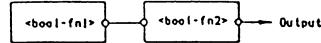
(a) Specification Misapplied by Ovvergeneralized Rule

If the function to be implemented is of the form:

```
Inputs : <anyt>
Outputs : Output
Function : (Equals (Value Output (1))
  (If (And <bool-fn1> <bool-fn2>)
    Then <any1> Else <any2>))
```

Where Input Signals Satisfy:  
 $\text{(Equals (Voltagelevel } \text{Input1}) \text{ anyt) Switchable-High-Level)}$

THEN one possible implementation is:



(b) Ovvergeneralized Rule

Figure 3. Ovvergeneralized Rule and Its Misapplication

"High" as its output. Operating-Conditions of the Pull-Up-Networks ("Voltagelevels of input signals should be Restoring-Logic-Levels") can not be satisfied with Operating-Conditions in Fig. 3(a) ("Voltagelevels of inputs are Switchable-High-Levels"). However, the rule shown in Fig. 3(b) can apply to this module specification shown in Fig. 3(a). This means that this rule is ovvergeneralized.

The functional specification in Fig. 3(a) should be implemented by Pull-Up-Networks. However, Pull-Up-Networks require that the voltage levels of input signals should be Restoring-Logic-Levels. Therefore, the requirements ("The voltage levels of input signals should be Switchable-High-Levels") shown in Fig. 3(b) are too weak, where the relationship between these voltage levels is shown in Fig. 4. That is, Switchable-High-Level includes Restoring-Logic-Level in terms of logical high level.

## RESULT

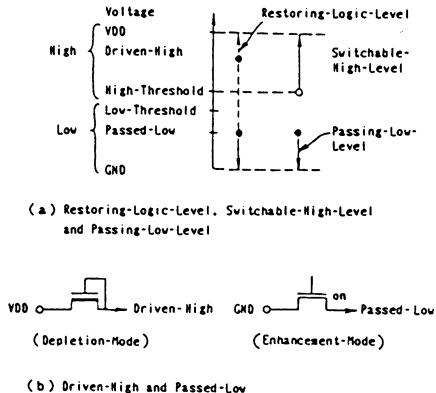


Figure 4. Classification of Voltage-Levels in MOSFET

## Reviewing Operating-Conditions

There are two solutions for describing the complete Operating-Conditions depending on internal structures. The first is exporting internal ports. For example, Source port of Pass-Transistor in Fig. 1 is exported to its supermodule (Pass-1). As a result, a class of an input signal to be connected with this Source port is explicitly restricted in the original specification. However, this means that the user, who describes the module specification, has already decided its internal structure by which to implement the module. This assumption is not always suitable, because the module specification is not necessarily complete in the early stage of design. Therefore, this solution does not seem to be good. The second is describing requirements on a class of input signals conditioned by output signals instead of internal structures. In some cases, the internal structures can be represented by the output values in the functional specification. In the above example, the output values of "Passed-Low" or "High" implicitly means that Source port is connected with GND or VDD, respectively. The specific output values (voltage levels of output signals) in the Mapping should be utilized to restrict classes of input signals. This approach suggests that Operating-Conditions should be described by both input and output signals.

## Representing Operating-Conditions depending on internal structures

The above consideration suggests the following representation method: when there are specific output values, which are not the values of input signals, in the functional specification, the specific values should not be generalized into arbitrary variables if the specific values do not cover all expected output values. For instance, as the specific output values "(Passes-Low, Tristated)" does not cover all expected output values "(Passed-High, Passed-Low, Tristated, ...)", they should not be generalized into arbitrary variables. According to this method, Operating-Condition of Pass-1 in Fig. 1 should be represented as "(Equals (Voltagelevel Gate (i)) (If (Equals (Voltagelevel Drain (i)) Passing-Low-Level) Then Switchable-High-Level Else Restoring-Logic-Levels))".

## Learning Operating-Condition depending on internal structures

A Learning method to derive Operating-Conditions of the composed module from Operating-Conditions of several submodules is here explained. LEAP [254] obtains these Operating-Conditions by backpropagating the Operating-Conditions of downstream submodules in to upstream submodules. When Operating-Conditions are described by using output values, the Operating-Conditions should be forwardly propagated. For example, Operating-Conditions of Input in Fig. 1 can be obtained by propagating constraints on output values of Drain of Pass-1 into Out of Pass-Net-1 through In1 of Mapping of Pass-Net-1, where the Mapping of Pass-Net-1 is "(Equals (Value Out (i)) (If (Not (Equals (Value In2 (i)) (Value In3 (i)))) Then (Value In1 (i)) Else Tristated))", and furthermore into Output port of the composed module. As a result, Operating-Condition of Input1 becomes "(Equals (Voltagelevel Input (i)) (If (Equals (Voltagelevel Output (i)) Passing-Low-Level) Then Switchable-High-Level Else Restoring-Logic-Level))". Complete Operating-Conditions derived by this proposed method can assure the Mapping function of the composed module generalized by Verification-Based method [221].

## CONCLUSION

This paper has presented a modified representation of Operating-Conditions, depending on internal structures, by using both input signals and output signal, and how to learn the Operating-Conditions from Operating-Conditions of submodules. This modified Operating-Condition can be considered "strong" conditions, compared with Operating-Conditions

based on just input signals ("weak" conditions) in LEAP [254].

## **ACKNOWLEDGMENTS**

The author wishes to express his sincere appreciation to Professors Tom Mitchell and Lou Steinberg of Rutgers University for their fruitful guidances when he was at Rutgers as a visiting researcher. He would like to thank Dr. Katsuya Hakozaki, Dr. Masahiro Yamamoto and Mr. Nobuhiko Koike of NEC Corporation for their encouragement.

# **OVERVIEW OF THE ODYSSEUS LEARNING APPRENTICE**

**David C. Wilkins, William J. Clancey, and Bruce G. Buchanan**

Stanford University, Department of Computer Science  
Stanford, CA 94305

## **ABSTRACT**

Human specialists employ impressive learning methods during their apprenticeship training period to augment their fledgling expertise. We describe an apprentice learning system under development that allows an expert system to use some of these same methods. These methods aid an expert system in transferring expertise to and from its knowledge base (i.e., in knowledge acquisition and intelligent tutoring).

Our approach to apprenticeship learning is embodied in a computer program, Odysseus, that watches the observable actions of a specialist. Justifications are created for each action of the specialist via a process of differential modeling between the specialist and the expert system. A learning opportunity occurs when no action justification is judged sufficiently plausible. This paper describes the three phases that Odysseus uses to learn via differential modeling: setting the stage for differential modeling by expanding the initial rule base and deriving rule justifications, detecting knowledge base differences by observing actions of a specialist and ranking proposed action justifications, and effecting knowledge base repair by rationalizing discrepancies and postulating new rules.

## **INTRODUCTION**

An apprenticeship learning period is an important phase on the path to master expert status for human specialists.<sup>1</sup> During this phase, an apprentice specialist *learns by watching master specialists* and *learns by doing problem solving* under the supervision of master specialists. Our research investigates how to give an expert system the benefits of an

---

<sup>1</sup>By *specialist*, we mean a problem solver whose abilities are at the novice or master level, and who is either a human or an expert system.

apprenticeship period.

Our method of apprenticeship learning is embodied in a computer program, Odysseus, that learns by watching specialists in the domain of medical diagnosis. The central task of Odysseus is to *rationalize* each observable action of a specialist during problem solving sessions. In medical diagnosis, these actions consist of all data requests made by a physician and the final diagnosis. Actions are rationalized by a process of *differential modeling* between the expert system and the specialist. Failure to find an adequate rationalization signals a possible deficiency in the expert system's domain or strategy knowledge. Using a taxonomy of deficiencies in conjunction with theoretical and experiential knowledge of the application domain, Odysseus automatically generates and tests conjectures to explain its inability to justify a specialist's action.

Odysseus is designed to work in conjunction with Heracles, an expert-system shell for solving heuristic classification problems, that was created by removing the medical knowledge from Neomycin [72]. Neomycin is a reorganization of the Mycin expert system that simulates the diagnostic process of medical experts, via a large body of abstract domain-independent strategy knowledge for hypothesis-directed reasoning. This strategy knowledge is used by Odysseus as a framework for detecting differences between the domain knowledge of a Heracles-based expert system and of a specialist. Odysseus has an abstract strategy language that allows comparison of the strategic behavior of the expert system and of a specialist [394].

## ODYSSEUS' METHOD

### Expanding Rule Base and Deriving Rule Justifications

There are two ways in which an existing expert system must be augmented before differential modeling of a human specialist can commence. First, the set of heuristic rules must be expanded via induction over past problem solving cases. The original set of rules is adequate for problem solving but, in our experience, is too impoverished to model the alternate problem solving behavior of other specialists in an apprentice context. Second, rules should be justified from first-principle knowledge or experience. Rule justifications allow a learning system to reason about the rules during the process of rationalizing discrepancies. The Leap learning apprentice for circuit design justifies rules in terms of circuit theory, a strong theory of the domain [254]. By contrast, only a weak theory generally underlies medical diagnosis, and Odysseus's justifications for rules

rely strongly on their empirical predictive power.

The induction subsystem of Odysseus is principally concerned with searching the space of rules of the form  $\text{lhs} \rightarrow \text{hypothesis}(\text{cf})$ , where  $\text{cf}$  is a Mycin-type certainty factor. A constrained rule generator and a candidate rule evaluator find all  $\text{lhs}$  forms that meet given constraints of minimal rule generality (coverage), minimal rule specificity (discrimination), maximal rule collinearity (similarity), and maximal rule simplicity (number of conjunctions and disjunctions). The rule evaluator always gives preference to collinear forms of heuristic rules contained in the original rule base. The expanded rule set produced by the induction subsystem is necessarily incomplete; however, it bootstraps the differential modeling process that leads to its refinement. Later, we will discuss how the induction subsystem suggests missing rules to the repair subsystem during the process of rationalizing discrepancies.

### Observing Actions and Detecting Knowledge Base Differences

Odysseus must decide whether an action of the specialist suggests a significant domain or strategic knowledge difference between the specialist and the expert system. For each observed action of the specialist, Odysseus generates an action justification set:  $J(A) = (j_1, j_2, \dots, j_n)$ . An action justification structure,  $j_k$ , relates an action  $A$  to an abstract strategic goal  $G$  via a skeletal rule path, that is,  $A \rightarrow R_1 \rightarrow R_2 \rightarrow \dots \rightarrow G$ . A typical goal might be the confirmation of a particular hypothesis. All skeletal rule paths beginning with  $A$  and leading to a goal are in the set  $J(A)$ ; thus the set delimits the possible interpretations that can be attributed to the specialist's action. Using the original Neomycin rule base, the average size of  $J(A)$  is 20 and the maximum size is approximately 400.

Action justification sets are posted on a blackboard, and a variety of knowledge sources (KSs) attach confirming and disconfirming evidence to individual action justifications. The more important KSs are as follows: The *Heracles simulator* KS processes the information obtained during the problem solving session and relates the current status of findings, hypotheses, and rules to individual action justifications. For example, if this KS believes that particular hypotheses have already been concluded, then it attaches negative evidence to all action justifications whose goal is to confirm one of these hypotheses. The *multiple interpretations* KS consists of heuristic rules that medical domain experts use to arbitrate between multiple interpretations. For instance, early in the consultation session with different action justifications confirming different hypotheses, the more general hypotheses are preferred. The *user model* KS records user

characteristics such as individual diagnostic style preferences, and employs these to help arbitrate between competing action justifications. For example, some problem solvers have a depth-first problem-solving style, meaning that they pursue a particular hypothesis as soon as there is weak evidence confirming it. Such a heuristic adds support to those action justifications consistent with this style. The *strategic distance* KS determines the similarity between the expert system's preferred strategic action and the strategic action associated with each action justification. The *patterns of interpretation* KS rates competing justifications according to the overall coherence they lend to the specialist's strategic plan.

After evidence has been gathered for and against members of  $J(A)$ , the action ranking subsystem must decide if any of the top ranked justifications are equal to the specialist's justification  $j_s$ . This is the most difficult task the apprentice learner faces, since there are often weakly plausible interpretations under which any action of the specialist is reasonable; yet to learn, the program must recognize when none of its action justifications obtain. Some apprentice systems do not need to confront this problem. In Leap, the specialist's design actions implicitly contain the domain knowledge to be acquired; while in Odysseus the medical specialist's actions only indirectly reflect the domain knowledge to be learned.

When the specialist being observed is Neomycin, Odysseus always selects the correct action justification from  $J(A)$ . However, when observing other novice and master medical specialists,  $j_s$  was not in the set of justifications generated from the original Neomycin knowledge base 75% of the time. This incompleteness was due to sparse domain knowledge, and motivated the initial induction phase to expand the rule base. Observing actual specialists also showed that more heuristics needed to be added to the KS's.

### Rationalizing Discrepancies and Postulating New Rules

A learning opportunity exists when Odysseus concludes that no member of  $J(A)$  adequately explains the specialist's action. At this point a repair subsystem, under implementation, engages the specialist in a dialogue to determine  $j_s$ . If  $j_s$  is not in  $J(A)$ , then there is a domain knowledge difference between the specialist and the expert system. If  $j_s$  is indeed in  $J(A)$ , then there is a domain or strategy knowledge difference, or a problem with the ranking heuristics. The evidence explicitly linked to the action justification structure by the KSs should allow Odysseus to isolate the cause of the difference.

Odysseus will also learn without engaging the specialist in a dialogue. Given an unexplained action, the action justification subsystem will provide the repair subsystem with the most likely goal(s) to which the specialist's action relates. The repair subsystem will perform a bidirectional search for a skeletal path between the action and the goal, calling on the induction subsystem to check for rules that could connect the two search frontiers.

## SUMMARY AND FUTURE WORK

This paper has provided an overview of the three phases used by Odysseus to automate the transfer of expertise for expert systems, and given the results of implementing the first two phases. We are currently implementing the method for rationalizing discrepancies and the expanding the heuristics associated with the KSs.

Odysseus is to be tested as a knowledge acquisition subsystem for the Heracles expert system, and also tested as a student modeling subsystem for a Heracles-based intelligent tutoring system (Guidon2). A third, more systematic validation exercise involves apprenticeship learning between two expert systems. In this exercise, a novice expert system will always have one less piece of knowledge than the master expert system. Each type of knowledge relation, such as trigger properties on rules, will be systematically removed from the knowledge base of the novice. For each removal, we will test whether the novice can learn the missing piece of knowledge in an apprentice setting. These multiple perspectives should aid us in becoming experts in the transfer of expertise.

## ACKNOWLEDGMENTS

The apprentice learning framework described here owes much to discussions with Avron Barr, Jim Bennett, Tom Dietterich, Paul Scott, and Derek Sleeman. Marianne Winslett Wilkins and Paul Rosenbloom gave insightful comments on earlier drafts. For critiquing Odysseus during its development, we are grateful to doctors Larry Fagan, Curt Kapsner, Randy Miller, Mark Musen, Roy Rada and Ted Shortliffe.

This work was supported in part by NSF grant MCS-83-12148 and ONR/ARI contract N00014-79C-0302. Computational resources were provided by SUMEX-AIM (NIH grant RR 0078) and Xerox PARC.

# LEARNING FROM EXCEPTIONS IN DATABASES

## Keith E. Williamson

Rutgers University, Department of Computer Science  
New Brunswick, NJ, 08903

### MOTIVATION

To utilize database management systems, a database designer must construct a *schema*, which is used to validate the data stored and help set up efficient access structures. Because database design is an art, and because the real world is irregular, unpredictable, and evolves, truly useful database systems must be tolerant of occasional deviations from the constraints imposed by the schema, including so-called "semantic integrity constraints," which are first-order assertions about consistent database states. For example, the integrity constraint

$$\forall e \in \text{employees}, \text{wages}(e) < \text{wages}(\text{supervisor}(e))$$

might not be true if an employee had a supervisor who was on a leave of absence. Techniques presented in [30] show how exceptional information can be accommodated in the general context of what are known as "semantic data models" (e.g. DAPLEX, SDM, TAXIS) [31]. An important aspect of this accommodation is resolving the inconsistency that results from exceptional information and the constraint it violates. This can be achieved by modifying the constraint so that it applies in a more restricted situation (by essentially ignoring each exceptional case). For example, if employees e45 and e52 earn more than their supervisors, then the above constraint could be modified to

$$\forall e \in \text{employees}, e = e45 \vee e = e52 \vee \text{wages}(e) < \text{wages}(\text{supervisor}(e))$$

Ideally, an induced description for the exceptions to a constraint can be used to do this in a more general fashion. The above constraint, for example, might be stated more appropriately as

$$\forall e \in \text{employees}, \text{status}(\text{supervisor}(e)) = \text{on-leave} \vee \text{wages}(e) < \text{wages}(\text{supervisor}(e))$$

## RESULTS

By generalizing from the exceptions to a constraint [32], we are able to *refine* the database schema to better characterize reality as it is reflected in the data encountered, including the exceptions. The types of schema refinements include: modifying constraints, adding attribute definitions to existing classes, and creating new classes and placing them into their correct location in the generalization partial-order. The benefits of modifying the schema in this more general way include: having fewer exceptions to encounter in the future, making the checking of constraints more efficient, making information retrieval more efficient, and giving the user a more refined vocabulary in which to pose queries and enter information. We see our algorithms as part of a "database administrator's assistant" - a computer system which can suggest modifications and additions to the schema.

We have considered two approaches to generalizing from exceptions to constraints in semantic data models. The first approach [32] is an empirical approach, in which a specific-to-general breadth-first search is used to induce a set of descriptions for a set of exceptions to a constraint. This syntactic method is augmented with a technique that attempts to judge which portions of a description are more *relevant* to the task at hand (describing exceptions to a particular constraint) so that irrelevant ones can be discarded. This is especially important since we use a generalization rule that leads to (potentially) exponential growth in the description size, or even worse, infinite recursion; this rule allows us to generalize a reference to an entity by introducing a description of its attributes and then recursively generalizing them (e.g. describing employees by describing attributes of their supervisors).

## CURRENT WORK

The second technique for generalizing from exceptions is an analytic approach [33], in which a detailed theory of the domain (of the database) is used to explain why the rationale for a constraint does not hold for a particular exceptional instance. The most general features of the instance that are necessary for the explanation to carry through are then determined by a process similar to constraint back-propagation (see [255] and [372]); we are currently investigating how the goal-regression procedure of [269] can be modified to regress a (generalized) goal through a derivation that uses horn-clause rules. An interesting aspect of this work is that, in general, our theory of the domain will be both incomplete and inconsistent

(default rules are used). The incompleteness forces us to occasionally fall back on empirical generalization techniques, while the use of default rules causes some concern about the consistency of our explanations and the correctness of our generalizations.

## FUTURE PLANS

Our immediate plans are to formalize our analytic generalization technique. While we will initially assume that our theories consist of horn-clause rules only, we hope to extend this to other forms as well. On a different note, we plan to address the problem of overwhelming the database administrator with suggestions for new classes to add to the schema. We intend to develop a mechanism for filtering through the numerous suggestions for new classes and coming up with class definitions that are amalgams of various suggestions.

# LEARNING APPRENTICE SYSTEMS RESEARCH AT SCHLUMBERGER

Howard Winston<sup>1</sup>, Reid Smith<sup>1</sup>, Michael Kleyn<sup>1</sup>, Tom Mitchell<sup>2</sup>, and Bruce Buchanan<sup>3</sup>

## ABSTRACT

We are developing a *Learning Apprentice System* (LAS) that partially automates the initial construction of a rule-based performance program from first principle domain knowledge and assists in refining the performance program's rules during routine use. A simple implementation has been constructed that demonstrates the feasibility of building such a system.

## INTRODUCTION

Work is currently underway on a prototype LAS that addresses two problems of knowledge-based system design: (*i*) how to exploit machine learning techniques to relieve the knowledge acquisition bottleneck; and (*ii*) how to reason from basic principles when rule-encoded approximate knowledge is inadequate to solve a problem.

Knowledge acquisition for expert systems is a time-consuming and painstaking process whereby the formalization of problem-solving knowledge, through discussions between a domain expert and a computer scientist (or *knowledge engineer*), is interleaved with coding and testing. Our LAS automates part of this process by both synthesizing an initial set of interpretation rules from basic domain knowledge and utilizing machine learning techniques to assist in their refinement during normal use.

Knowledge-based systems that cannot reason from basic principles do not exhibit a gradual degradation in their performance as they are presented with either: (*i*) increasingly complex problems within their domains, or (*ii*) problems increasingly removed from their domains. Our LAS can reason

---

<sup>1</sup>Schlumberger-Doll Research, Old Quarry Road, Ridgefield, CT, 06877

<sup>2</sup>Rutgers University, Department of Computer Science, New Brunswick, NJ, 08903

<sup>3</sup>Stanford University, Department of Computer Science, Stanford, CA, 94305

from basic principles to explain and correct mistakes made by the performance program. This reasoning is carried out by inspecting the record of assumptions and approximations that were made in deriving the performance program rules from an underlying domain model. If a rule error can be traced to an invalid approximation, that approximation can be corrected and the rule can be rederived and reapplied.

## BACKGROUND

This work is based on our experience with the *Dipmeter Advisor*\* knowledge-based system that interprets dipmeter (and other) logs to infer the geologic structures penetrated by an oil-well borehole. Dipmeter logs are records of the dip, or slope, of rock layers as a function of depth. Patterns in these logs can be used to identify subterranean formations. For example, an interval of increasing dip is referred to as a red pattern, and parts of formations that have been distorted (*i.e.*, distortion zones) characteristically give rise to red patterns on dipmeter logs.

The *Dipmeter Advisor* system partitions the interpretation task into a number of phases between which the user can alter intermediate conclusions reached by the system. One of the objectives of our LAS is to construct a system that can take advantage of this feedback to improve its performance on subsequent interpretation tasks.

## LAS ARCHITECTURE

Our proposed LAS contains five major sub-systems: (1) The *rule generation program* combines information about geologic structures and logging tools to automatically synthesize a set of predictive rules of the form (situation  $\Rightarrow$  manifestation). As a side-effect, the rule generation program builds a *justification structure* that records the inferences made in deriving the predictive rule from basic domain knowledge. (2) The *inversion program* converts predictive rules into interpretation rules of the form (manifestation  $\Rightarrow$  situation). (3) The *performance program* consists of the set of synthesized interpretation rules together with a rule interpreter that can apply them to input data. (4) When a conclusion reached by a performance program rule differs from one reached by a knowledgeable user, the *critic program* locates possible sources of this discrepancy by using the

---

\*Mark of Schlumberger

justification structures associated with the rules<sup>4</sup>. (5) The *rule editor* program then interactively pinpoints and repairs the faulty assertion in the justification structure and recompiles a new interpretation rule.

The rule generation program synthesizes a set of predictive rules from an underlying domain model. The rules that comprise this program are referred to as linkage rules, and a record of the firing of linkage rules is preserved in a justification structure. The justification structure is a network of *Beliefs* connected by *Justification Links*. Beliefs represent assertions and justification links record the derivational dependencies between beliefs. Each belief contains an assertion and information about the type, or source, of the belief<sup>5</sup> as well as its degree of belief. Each justification link contains a linkage rule and error propagation information. Linkage rules are rules of inference that enable a justified belief to be derived from its justifier beliefs, and the error propagation information specifies the way in which errors in justifier beliefs propagate to the justified belief.

The rule generation program contains an interesting situation generator (ISG) that discovers equivalence classes of situations, or interesting cases, that give rise to qualitatively distinct manifestations. The ISG partitions the range of possible values of each situation parameter into a set of subranges that correspond to qualitatively distinct manifestations. The ISG uses information about parameter range extrema and how local variations in situation parameters cause local variations in manifestation parameters to successively differentiate each situation parameter. This is continued until the entire situation parameter space has been partitioned into interesting cases. The output of the ISG is a hierarchical classification of these subcases.

The inversion program uses abduction to derive interpretation rules from predictive rules. An abductive inference allows the conclusion of B to be drawn from  $B \Rightarrow A$  and A. For example, from the predictive rule that distortion zones cause red patterns and an observation of a red pattern in a dipmeter log, abductive inference permits the conclusion that there exists a distortion zone. The inversion program represents this abductive inference in the form of the interpretation rule that red patterns suggest

<sup>4</sup>In (1) we present a detailed example of how justifications can be used by the critic to isolate candidate errors in the support of a performance program rule.

<sup>5</sup>The type of a belief can be either *definitional* (i.e., no further justification is required), *theoretical* (i.e., could be incorrect if the domain theory is incorrect), *statistical* (i.e., justified by statistical experience), or *default* (i.e., cannot be estimated without information either typically unavailable to the system or expensive to obtain).

distortion zones. Because red patterns are commonly associated with a number of other causes as well, the backward (interpretive) form of the predictive rule can only be used to *suggest* the cause when the manifestation is observed.

The performance program uses the set of synthesized interpretation rules. It is an interactive program that allows the user to correct errors made by the system. The user can delete, add, or modify system-drawn conclusions.

Based on user feedback, the critic locates parts of the underlying domain model that can be suspected of causing interpretation rule errors. The critic uses the error propagation information recorded in the justification structures of the interpretation rules to constrain the search for suspects. For each possible error in a belief, the supporting beliefs that could have caused the error are enumerated along with their suspected error types. These suspects are then pruned by attempting to determine the correctness of each suspect in the current situation and removing those shown to be correct. The remaining suspects are then ranked according to their type of belief, and those whose type of belief is definitional are removed. Finally, each remaining suspect is recursively investigated by this same method.

Given the set of suspects located by the critic, the rule editor corrects faulty domain knowledge and recompiles a set of interpretation rules that consistently reflects these repairs. The rule editor currently uses a mapping from error types to repair operations but should also be able to query the user for assistance in repairing the domain model.

## CURRENT STATUS

We have tested our proposed LAS architecture by generating and refining an interpretation rule that identifies late faults in dipmeter logs.

## FUTURE PLANS

Rule justifications provide support for the inferences made by the performance program. The content of these justifications is the deeper domain knowledge from which the rules are derived, together with the assumptions and approximations that have been used in their derivation. One might consider the construction of another layer of justification – support for the inferences made in linking beliefs in the rule justifications. We plan to examine this form of multi-layer justification.

## ACKNOWLEDGMENTS

Some of the ideas about how to generate error propagation information were contributed by Mike Barley. Ed Wisniewski developed the interesting situation generator.

*concept* cannot be produced from the composition of its constituents. Indeed, an interpretation of the phrase based on the meanings of its constituents often exists, but it carries a different meaning. We describe a space of phrases ranging in generality from fixed proverbs such as charity begins at home through idioms such as: lay down the law and phrasal verbs such as: put up with one's spouse and look up the name in the list, to literal verb phrases such as: sit on the chair. We suggest employing a *phrasal lexicon* to capture this entire range of language structures. Four issues must be addressed when learning phrases in context.

## DETERMINING SCOPE AND GENERALITY OF PATTERNS

The linguistic pattern of a phrase may be perceived by the learner at various levels of generality. For example, in My boss put his foot down, incorrect acquisition could yield patterns accepting sentences such as:

- He put his left foot down.
- He moved his foot down.
- He put down his foot.
- He put it.

Another pair of examples which demonstrate this problem is: (1) The show is yours. Take it away! as opposed to (2) This cheese stinks. Take it away!. In (2), take it away is a specific idiom, while in (1) it is an instance of a general verb phrase. The learner must use strategies [404] to arrive at the appropriate level of generality. Pattern acquisition is viewed as a search, driven by failures in parsing, through the partially ordered space of phrase patterns. As an example for the pattern-refinement process, consider this part of the dialog:

- |         |                     |
|---------|---------------------|
| RINA:   | David took on him.  |
| Native: | No. He took him on. |
| RINA:   | He took Goliath on. |

Initially, RINA had incorrectly acquired the pattern:

?x:person take:verb <on ?y:person>

in which the particle *on* is associated with the object (*?y:person*) as a *case marker*. Therefore RINA first generates: David took on him. However, when she hears from the native: No. He took him on, she detects a discrepancy regarding the location of the particle and corrects the pattern representation to:

?x:person <take:verb on> ?y:person

In this pattern the particle is associated with the verb as a modifier, supporting the generation of the correct sentence: He took Goliath on. In this example, the phrase has been generalized by allowing flexibility in its

word order.

## EXTRACTING MEANINGS FROM CONTEXT

The conceptual representation of the phrase must be extracted from the context which contains many concepts, both appropriate and inappropriate for hypothesis formation. We have developed strategies [403] for focusing on appropriate elements in the context, where the context may be given either as a reference to a pre-compiled story or a text which is currently being processed. For example, in explaining the meaning of *show on* (as yet a non-existing phrase), a native speaker might give this example: Remember Aesop's fable about the fox and the crow (where the fox convinces the crow to sing, thus gaining his piece of cheese? Well, the fox showed on the crow). Clearly, the guessed meaning of *show on* concerns story elements beyond the level of the mentioned animals (fox and crow) or the specific acts (singing, talking and eating). Rather, the meaning is taken to be the moral of the story, TAU-Deceipt [100]. Therefore, a typical guess would be that *show on* here means: "the fox tricked the crow," depicting that *thematic abstraction unit* (TAU).

Similarly, in the story of David and Goliath, referred to in the dialog above, the learner guesses the meaning from *story points* [391] given in the form of violated expectations. (First point: in spite of his physical inferiority, David **won** the fight; second point: in spite of his physical inferiority, he **decided** to fight.) A story point provides an abstraction of mundane details that appear in the original text of the story. Therefore, RINA guesses he won the fight. Second language speakers make this same **incorrect** guess.

As opposed to the previous situations, the learner might hear a new phrase before any such abstraction has been formed. Thus we need a strategy for extracting a concept from a partial context. For example, consider the sentence: Jenny wanted to go punk, but her father put his foot down, where the new phrase is put one's foot down. At the point that the phrase is encountered, the expectation is that the father will object to her plan, a hypothesis which is supported by linguistic clues in the sentence. Consequently, the phrase meaning is associated with that expectation; the general strategy being to associate the meaning with the active expectation.

## INCORPORATING EPISODES

Generation of examples is rina's way of conveying her conception of new phrases. For example, after participating in the dialog above, RINA will explain both the pattern and the concept of the phrase take on by generating: David took on Goliath (rather than spelling out the syntax and the semantics of the phrase). However, given only the concept of the phrase, the generation of such an example is a difficult task which involves scanning the entire database for an episode which satisfies the requirements of the definition. Therefore, in the process of learning a phrase, such episodes are indexed to the lexical entry as they are encountered. Episodes linked to a phrase also facilitate the reasoning process required in refining phrase concepts.

## REFINING PHRASE CONCEPTS

Initially, a phrase concept is established using strategies as described in Section 2. However, the concept is further refined on encountering additional examples. Consider one possible sequence of examples for the phrase take on as encountered by RINA:

- David took on Goliath.
- I took on my elder brother.
- He took on a new job.
- We took on a new systems programmer.

The dilemma at each encounter, is whether the new example is an instance of the existing phrase, or is it a new lexical entry. From the first example (David and Goliath), RINA constructs the initial concept: person ?x decides to fight a stronger person ?y. This concept holds well in the second encounter (my elder brother). A modification of the concept is required on the third example (his new job). The concept is generalized to: person ?x decides to accept a challenge ?y. The last example (the system programmer) is drastically different. No generalization is found, which encompasses the new situation as well as the previous situations, therefore a new lexical entry is initiated: management ?x decides to hire a new employee ?y. Notice that although a very general concept which encompasses all the given examples **could** be found (?x does something to ?y), such an over-generalized concept will diminish the effectiveness of the acquired phrase.

## CONCLUSIONS

The lexical-functional [41] approach to natural language processing emphasizes the role of the lexicon. Rather than maintaining a single "generic" entry for each word (take, for example), the lexicon contains many phrases (take to the streets, take to swimming, take it easy, take over, etc.). This approach proved effective both in parsing and in generation [369], however it requires the acquisition of a huge lexicon which cannot be done manually (considering the subtle meanings of phrases). Moreover, only by following the learning process can the lexicon capture correct phrase behavior. For example, understanding why to kick the bucket, as opposed to to bury the hatchet, cannot take the passive voice, may be captured only in the acquisition process [403].

Two processes are involved in learning: the search for the pattern in the space of patterns, which presents a well-defined search problem, and the construction of a concept from a given sequence of contexts. These two processes are not independent, as linguistic clues are used in constructing concepts and conceptual clues are used in forming phrase patterns. The task is further complicated by the operational need to parse the given text while the new phrase does not yet exist in the lexicon. It is our objective to relate this cognitive process to existing learning paradigms.

## References

- [1] Ackley, D., Hinton, G. and Sejnowski, T. "A Learning Algorithm for Boltzmann Machines." *Cognitive Science*. 9:1 (1985) 147-169.
- [2] Alkon, D. "Cellular Analysis of a Gastropod (*Hermissenda Crassicornis*) Model of Associative Learning." *Biological Bulletin*. 159 December (1980) 505-560.
- [3] Amarel, S., "On Representations of Problems of Reasoning about Actions," in *Machine Intelligence*, D. Michie, ed., Edinburgh University Press, 1968, 131-171, ch. 10.
- [4] Amarel, S. "Expert Behavior and Problem Representations", Technical report CBM-TR-126, Rutgers University, March 1982.
- [5] Anderberg, M. *Cluster Analysis for Applications*. Academic Press, 1973.
- [6] Anderson, J. *The Architecture of Cognition*. Harvard Press, 1983.
- [7] Anderson, J. "Knowledge compilation: The general learning mechanism." In *Proceedings of the 1983 Machine Learning Workshop*. R. Michalski, J. Carbonell, and T. Mitchell, eds., 1983.
- [8] Anderson, J., and Bower, G. "Recognition and Retrieval Processes in Free Recall." *Psychological Review*. 79 (1972) 97-123.
- [9] Angluin, D., and Smith, C. "Inductive Inference: Theory and Methods." *ACM Computing Surveys*. 15:3 September (1983) 237-269.
- [10] Bain, W. "Toward a Model of Subjective Interpretation". Technical report 324, Yale University, July 1984.
- [11] Bain, W., "Assignment of Responsibility in Ethical Judgments," in *Memory, Experience, and Reasoning*, Lawrence Erlbaum Associates, 1985, in press.
- [12] Balzer, R., Green, C. and Cheatham, T. "Software Technology in the 1990's Using a New Paradigm." *Computer*. November (1983) 39-45.
- [13] Banerji, R.. "Pattern Recognition, Structural Description Languages," in *Encyclopaedia of Computer Science and Technology*, Dekker, N.Y., 1979.

- [14] Banerji, R. and Ernst, G. "A Theory for the Complete Mechanization of a GPS-type Problem Solver." In *Proceedings of IJCAI-77*. Morgan Kaufmann, 1977, 450-456.
- [15] Barr, A., Cohen, P. and Feigenbaum, E. *The Handbook of Artificial Intelligence, Volumes I, II, and III*. Los Altos, CA: William Kaufmann, Inc., 1981 and 1982.
- [16] Barto, A. and Sutton, R. "Goal Seeking Components for Adaptive Intelligence", Technical report AFWAL-TR-811070, University of Massachusetts, 1981.
- [17] Becker, J. "Robot Computer Problem Solving System", Technical report TR-2316, Bolt Beranek and Newman Inc., September 1972.
- [18] Bennett, J. "ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System." *Journal of Automated Reasoning*. 1:1 (1985) 49-74.
- [19] Berliner, H. "Some Necessary Conditions for a Master Chess Program." In *Proceedings of IJCAI-73*. Morgan Kaufmann, 1973.
- [20] Berliner, H. "On the construction of evaluation functions for large domains." In *Proceedings of IJCAI-79*. Morgan Kaufmann, Tokyo, Japan, August, 1979, 53-55.
- [21] Berliner, H. and Campbell, M. "Using chunking to solve chess pawn endgames." *Artificial Intelligence*. 23:1 (1984) 97-120.
- [22] Biermann, A., Guiho, G. and Kodratoff, Y. *Automatic Program Construction Techniques*. New York: Macmillan Publishing Company, 1984.
- [23] Binford, T. "Visual Perception by Computer." In *Proceedings of the IEEE Conference on Systems and Control*. IEEE, December, 1971.
- [24] BIOCHIMIE, *Special Issue on Artificial Intelligence and Sequence Analysis*, 1985, volume 67 (in English).
- [25] Blair, D. and Maron, M. "An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System." *Communications of the ACM*. 28:3 March (1985) 289-299.
- [26] Bledsoe, W. "Using Examples to Generate Instantiations of Set Variables." In *Proceedings of IJCAI-83*. Morgan Kaufmann, 1983, 892-901.

- [27] Blum, L. and Blum, M. "Toward a Mathematical Theory of Inductive Inference." *Information and Control.* 28:2 June (1975) 125-55.
- [28] Bongard, N. *Pattern Recognition.* New York: Spartan Books, 1970.
- [29] Boose, J. "Personal construct theory and the transfer of human expertise." In *Proceedings of the National Conference on Artificial Intelligence.* Austin, Texas, 1984.
- [30] Borgida, A. "Language Features for Flexible Handling of Exceptions in Information Systems", Technical report LCSR-TR-70, Rutgers University, March 1985.
- [31] Borgida, A. "Features of Languages for the Development of Information Systems at the Conceptual Level." *IEEE Software.* 2:1 January (1985) 63-73.
- [32] Borgida, A. and Williamson, K. "Accomodating Exceptions in Databases and Refining the Schema by Learning from them." In *Proceedings of the Conference on Very Large Data Bases.* 1985.
- [33] Borgida, A., Mitchell, T. and Williamson, K., "Learning Improved Integrity Constraints and Schemas from Exceptions in Data and Knowledge Bases", to appear in *On Knowledge Base Management Systems.*
- [34] Bowen, K. and Kowalski, R. "Amalgamating Language and Metalanguage in Logic Programing". Technical report, School of Computer Science, Syracuse University, 1981.
- [35] Bower, G. and Winzenz, D. "Group Structure, Coding, and Memory for Digit Series." *Journal of Experimental Psychology Monograph.* 80 (1969) 1-17, (May, Pt. 2).
- [36] Bradshaw, G., *Learning to recognize speech sounds: A theory and model,* PhD dissertation, Carnegie-Mellon University, 1984.
- [37] Bradshaw, G., Langley, P. and Simon, H. "BACON 4: The discovery of intrinsic properties." In *Proceedings of the Third National Conference of the Canadian Society for Computational Studies of Intelligence.* Canadian Society for Computational Studies of Intelligence, Victoria, Canada, 1980, 19-25.

- [38] Bratko, I., Mozetic, I. and Lavrac, N., "Automatic Synthesis and Compression of Cardiological Knowledge," in *Machine Intelligence 11*, Hayes, J., Michie, D. and Richards, J., eds., Oxford University Press, Oxford, 1985, in press.
- [39] Brazdil, P., *A Model of Error Detection and Correction*, PhD dissertation, University of Edinburgh, 1981.
- [40] Brebner, P., "title not supplied," Master's thesis, University of Waikato, 1985.
- [41] Bresnan, J. *The Mental Representation of Grammatical Relations*. The MIT Press, 1982.
- [42] Brown, J., and Burton, R. "Diagnostic models for procedural bugs in mathematical skills." *Cognitive Science*. 2 (1978) 155-192.
- [43] Brown, J., and VanLehn, K. "Repair theory: A generative theory of bugs in procedural skills." *Cognitive Science*. 4 (1980) 379-426.
- [44] Buchanan, B., Sutherland, G. and Feigenbaum, E., "Toward an Understanding of Information Processes of Scientific Inference in the Context of Organic Chemistry," in *Machine Intelligence 5*, B. Meltzer and D. Michie, ed., Edinburgh University Press, Edinburgh, 1970.
- [45] Buchanan, B. and Mitchell, T., "Model-directed learning of production rules," in *Pattern-Directed Inference Systems*, Waterman, D. and Hayes-Roth, F., eds., Academic Press, New York, 1978.
- [46] Buchanan, B., Mitchell, T., Smith, R. and Johnson, C. Jr. "Models of Learning Systems." *Encyclopedia of Computer Science and Technology*. 11 (1978) 24-51, also Stanford report STAN-CS-79-692.
- [47] Buchanan, B., Barstow, D., Bechtel, R., Bennett, J., Clancey, W., Kulikowski, C., Mitchell, T. and Waterman, D., "Constructing an Expert System," in *Building Expert Systems*, F. Hayes-Roth, D. Waterman, and D. Lenat, ed., Addison-Wesley, New York, 1983.
- [48] Buchanan, B., and E. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley, 1984.
- [49] Bundy, A. "Poof Analysis : A Technique for Concept Formation." In *Proceedings of AISB-85*. Ross, P., ed., AISB, 1985, 78-86, also available as DAI Research Paper no. 198.

- [50] Bundy, A., Silver, B. and Plummer, G. "An Analytical Comparison of Some Rule Learning Programs". Technical report D.A.I. 215, University of Edinburgh, 1984.
- [51] Bundy, A. and Welham, B. "Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation." *Artificial Intelligence*. 16:2 (1981) 189-212.
- [52] Burstein, M. "A Model of Learning by Incremental Analogical Reasoning and Debugging." In *Proceedings AAAI-83*. Washington, D.C., August, 1983, 45-48.
- [53] Burstein, M., *Learning by Reasoning from Multiple Analogies*, PhD dissertation, Yale University, 1984.
- [54] Burstein, M., "Concept Formation by Incremental Analogical Reasoning and Debugging". *Machine Learning II*. Morgan Kaufmann, Publishers, forthcoming.
- [55] Caple, Balda, and Willis "How did Vetebrates take to the air." *Science*. July 1 (1983) .
- [56] Carbonell, J., *Subjective Understanding: Computer Models of Belief Systems*, PhD dissertation, Yale University, 1979.
- [57] Carbonell, J. "Derivational Analogy in Problem Solving and Knowledge Acquisition." In *International Machine Learning Workshop*. R. Michalski, ed., University of Illinois at Urbana Champaign, 1983.
- [58] Carbonell, J. "Derivational Analogy and Its role in Problem Solving." In *Proceedings AAAI-83*. Washington, D.C., August, 1983, 64-69.
- [59] Carbonell, J., Michalski, R. and Mitchell, T., "An Overview of Machine Learning." in *Machine Learning, An Artificial Intelligence Approach*, R. Michalski, J. Carbonell and T. Mitchell, eds., Tioga Press, Palo Alto, CA, 1983.
- [60] Carbonell, J., "Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition," in *Machine Learning II*, Michalski, R., Carbonell, J., And Mitchell, T., eds., Morgan Kaufmann, Los Altos, CA, 1985, (forthcoming).
- [61] Carey, S., "Cognitive Development: the Descriptive Problem," in *Handbook of Cognitive Neuroscience*, Gazzaniga, M., ed., Plenum Press, NY, 1984, 37-69.

- [62] Carlson, L. *Dialogue Games*. London: D. Reidel, 1983.
- [63] Carr, B. and Goldstein, I. "Overlays: A theory of modeling for Computer Aided Instruction", AI Memo 406, AI Laboratory, Massachusetts Institute of Technology, 1977.
- [64] Causey, R. "Derived Measurement, Dimensions, and Dimensional Analysis." *Philosophy of Science*. 36 (1969) 252-2699.
- [65] Charniak, E. "With a Spoon in Hand this Must be the Eating Frame." In *Theoretical Issues in Natural Language Processing*. Urbana, Ill., 1978, 187-193.
- [66] Chase, W., and Ericsson, K., "Skilled memory," in *Cognitive Skills and Their Acquisition*, J. Anderson, ed., Lawrence Erlbaum Associates, Hillsdale, NJ, 1981, 141-189, ch. 5.
- [67] Chase, W. and Simon, H. "Perception in chess." *Cognitive Psychology*. 4 (1973) 55-81.
- [68] Chauvet, J. "Genetic Epistemology." *AI Magazine*. 6 (1985) 18.
- [69] Checroun, N. *Pull Up a Chair*. Minneapolis, Minnesota: Lerner Publications Co., 1967.
- [70] Chilausky, R., Jacobsen, B. and Michalski, R. "An application of Variable Valued Logic to Inductive Learning of Plant Disease Diagnostic Rules." In *Proceedings of the Sixth Annual Symposium on Multiple Valued Logic*. Logan, Utah, 1976.
- [71] Clancey, W. and Letsinger, R. "NEOMYCIN: Reconfiguring a Rule-Based Expert System for Application to Teaching." In *Proceedings of IJCAI-81*. Morgan Kaufmann, Vancouver, British Columbia, 1981, 829-836.
- [72] Clancey, W. and Letsinger, R.. "Neomycin: Reconfiguring a rule-based system for application to teaching," in *Readings in Medical Artificial Intelligence*, Clancey, W. and Shortliffe, E., eds., Addison-Wesley, 1984, 361-381.
- [73] Clavieras, B., *Modification de la representation des connaissances en apprentissage inductif*, PhD dissertation, Universite de Paris-Sud, 1984.
- [74] Cochin, I. *Analysis and Design of Dynamic Systems*. New York: Harper and Row Publishers, Inc., 1980.

- [75] Cohen, B., *Understanding Natural Kinds*. PhD dissertation, Stanford University, August 1982.
- [76] Cohen, B. and Murphy, G. "Models of Concepts." *Cognitive Science*. 8 (1984) 27-58.
- [77] Coles, D. and Rendell, L. "Some issues in training learning systems and an autonomous design." In *Proceedings of the Fifth Conference of the Canadian Society for Computational Studies of Intelligence*. Canadian Society for Computational Studies of Intelligence, 1984.
- [78] Collins, A. and Gentner, D. "Constructing runnable mental models." In *Fourth Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 1982.
- [79] Collins, A. and Gentner, D. "Multiple models of evaporation processes." In *Fifth Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 1983.
- [80] Conway, J. *On Numbers and Games*. Academic Press, 1976.
- [81] Cullingford, R. "Script Application: Computer Understanding of Newspaper Stories", Technical report 116, Yale University, January 1978.
- [82] Davis, R., "Applications of Meta-level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases," in *Knowledge-Based Systems in Artificial Intelligence*, Randall Davis and Douglas Lenat, eds., McGraw-Hill, New York, 1982, PhD dissertation, also appears as Tech. memo 283, CSD/AIA Lab, Stanford University, 1976.
- [83] Davis, R. and Lenat, D. *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill, 1982.
- [84] DeGroot, A. *Thought and Choice in Chess*. The Hague: Mouton, 1965.
- [85] DeJong, G. "Prediction and Substantiation: a New Approach to Natural Language Processing." *Cognitive Science*. 3 September (1979).
- [86] DeJong, G. "Acquiring Schemata Through Understanding and Generalizing Plans." In *Proceedings of IJCAI-83*. Morgan Kaufmann, 1983.

- [87] DeJong, G. "Generalizations Based on Explanations." In *Proceedings of IJCAI-81*. Morgan Kaufmann, 1981.
- [88] DeJong, G.. "An Overview of the FRUMP System," in *Strategies for Natural Language Processing*, W. Lehnert and M. Ringle, ed., Lawrence Erlbaum, 1982.
- [89] DeJong, G. "An Approach to Learning by Observation." In *Proceedings, International Machine Learning Workshop*, 1983, 171-176, also appears as working paper 45, A.I. research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.
- [90] DeJong, G., Mooney, R., O'Rorke, P., Rajamoney, S., Segre, A. and Shavlik, J. "A Review of Explanation-Based Learning", Technical report, University of Illinois at Urbana-Champaign, 1985, in preparation.
- [91] de Kleer, J. "Multiple representations of knowledge in a mechanics problem solver." In *Proceedings of IJCAI-81*. Morgan Kaufmann, 1981.
- [92] Dietterich, T., London, B., Clarkson, K., and Dromey, G., "Learning and Inductive Inference," in *The Handbook of Artificial Intelligence*, William Kaufmann, 1982, ch. XIV.
- [93] Dietterich, T. and Michalski, R., "A Comparative Review of Selected Methods for Learning from Examples," in *Machine Learning*, Tioga, 1983, 41-82.
- [94] Dietterich, T. "Learning about Systems that Contain State Variables." In *AAAI-84*. William Kaufmann, 1984, 96-100.
- [95] Dietterich, T.. *Constraint Propagation Techniques for Theory-Driven Data Interpretation*, PhD dissertation, Stanford University, Dept. of Computer Science, December 1984, also appears as Technical report STAN-CS-84-1030, Stanford University, 1984.
- [96] Dietterich, T.. "The Methodology of Knowledge Layers for Inducing Descriptions for Sequentially Ordered Events," Master's thesis, University of Illinois, May 1980, also available as tech report no. 1024.

- [97] Dietterich, T. and Buchanan, B. "The Role of Experimentation in Theory Formation." In *Proceedings of the International Workshop on Machine Learning*. Univ. of Illinois at Urbana-Champaign, June, 1983. 147-155.
- [98] Dietterich, T. and Buchanan, B. "The Role of the Critic in Learning Systems." In *Adaptive Control of Ill-Defined Systems*. Oliver G. Selfridge, Edwina L. Rissland, and Michael A. Arbib, ed., Plenum Press, NATO Advanced Research Institute on Adaptive Control of Ill Defined Systems, June 21-26, 1981, 1984, 127-147.
- [99] Differding, J. "The OPAL Interface: General Overview". Working Paper, Stanford University Oncocin Projects, August 1984.
- [100] Dolan, C. and Dyer, M. "Learning Planning Heuristics through Observation." In *Proceedings of IJCAI-85*. Morgan Kaufmann, August, 1985.
- [101] Doran, J., "An Approach to Automatic Problem Solving," in *Machine Intelligence*, Collins, N. and Michie, D., eds., American Elsevier, New York, 1967.
- [102] Doyle, R. "Hypothesizing and Verifying Causal Models", Technical report AI Lab Memo 811, MIT, December 1984.
- [103] Duda, R., Gaschnig, J. and Hart, P., "Model design in the Prospector consultant system for mineral exploration," in *Expert Systems in the Micro-electronic Age*, Michie, D., ed., Edinburgh University Press, Edinburgh, 1979.
- [104] Duda, R., Hart, P. *Pattern classification and scene analysis*. New York: Wiley, 1973.
- [105] Duerr, J. and Quinn, W. "Three *Drosophila* mutations that block associative learning also affect habituation and sensitization." *Proceedings of the National Academy of Sciences*. 79 June (1982) 3646-3650.
- [106] Dufay, B. and Latombe, J., "An Approach to Automatic Robot Programming on Inductive Learning," in *Robot Research*, M. Brady and P. Richards, ed., Massachusetts Institute Technology Press, 1984.
- [107] Dyer, M. *In-Depth Understanding* . Cambridge, Mass.: Massachusetts Institute Technology Press, 1983.

- [108] Ellman, T. "Generalizing logic circuit designs by analyzing proofs of correctness." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, 1985.
- [109] Ernst, G. and Goldstein, M. "Mechanical Discovery of Classes of Problem-Solving Strategies." *Journal of the ACM*. 29:1 January (1982) 1-23.
- [110] Evans, T., "A Program for the Solution of a Class of Geometric Analogy Intelligence Test Questions," in *Semantic Information Processing*, Minsky, M., ed., MIT Press, Cambridge, Mass, 1968, 271-253.
- [111] Fahlman, S. "A Planning System for Robot Construction Tasks." *Artificial Intelligence*. 5:1 (1974) .
- [112] Falkenhainer, B. "ABACUS: Adding Domain Constraints to Qualitative Scientific Discovery", Technical report ISG 84-7, Intelligent Systems Group, University of Illinois at Urbana-Champaign, 1984.
- [113] Fang, J. *Towards a Philosophy of Modern Mathematics*. New York: Hauppauge, Paideia series in modern mathematics, Vol. 1, 1979.
- [114] Feigenbaum, E. "The simulation of verbal learning behavior." In *Proceedings of the Western Joint Computer Conference*. 1961, 121-132.
- [115] Feigenbaum, E., and Simon, H. "EPAM-like models of recognition and learning." *Cognitive Science*. 8 (1984) 305-336.
- [116] Feldman, J. "Some Decidability Results on Grammatical Inference and Complexity." *Information and Control*. 20:3 April (72) 244-261.
- [117] Feldman, J. and Ballard, D. "Connectionist Models and Their Properties." *Cognitive Science*. 6 (1982) 205-254.
- [118] Fickas, S., *Automating the Transformational Development of Software*, PhD dissertation, University of California at Irvine, 1982.
- [119] Fikes, R., Hart, P., and Nilsson, N. "Learning and Executing Generalized Robot Plans." *Artificial Intelligence*. 3 (1972) 251-288.
- [120] Fisher, D. "A Hierarchical Conceptual Clustering Algorithm". Technical report, University of Calif., at Irvine, 1984.

- [121] Fisher, D. and Langley, P. "Approaches to Conceptual Clustering." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, Ca., 1985, 691-697.
- [122] Forbus, K. "A study of qualitative and geometric knowledge in reasoning about motion". Technical report AI TR615, MIT, 1981.
- [123] Forbus, K. "Qualitative Process Theory." *Artificial Intelligence*. 24:1-3 (1984) 85-168, also appears as MIT technical report AI TR664 1982.
- [124] Forbus, K. and Gentner, D. "Learning Physical Domains: Towards a Theoretical Framework." In *Proceedings of the 1983 International Machine Learning Workshop*. University of Illinois, Urbana, Ill., 1983, 198-202.
- [125] Fourman, M. "Compaction of Symbolic Layout using Genetic Algorithms." In *Proceedings of the International Conference on Genetic Algorithms and Their Applications*. Carnegie-Mellon University, 1985, 141-153.
- [126] Fu, L., *Learning Object-level and Meta-level Knowledge for Expert Systems*, PhD dissertation, Stanford University, 1985.
- [127] Fu, L., and Buchanan, B. "Enhancing Performance of Expert Systems by Automated Discovery of Meta-Rules." In *Proceedings of the IEEE Conference, Denver, November 1984*. IEEE, November, 1984.
- [128] Genesereth, M.. "The role of plans in intelligent teaching systems," in *Intelligent Tutoring Systems*, Brown, J. and Sleeman, D., eds., Academic Press, New York, 1982.
- [129] Gentner, D. "Structure Mapping: A Theoretical Framework for Analogy." *Cognitive Science* 7(2). April-June (1983) 155-170.
- [130] Glass, A., Holyoak, K. and Santa, J. *Cognition*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1979, Chapter 10: The Structure of Categories.
- [131] Gold, E. "Language Identification in the Limit." *Information and Control*. 10 (1967) 447-474.

- [132] Granger, R., Schlimmer, J. and Young, M.. "Contingency and latency in associative learning: computational, algorithmic and implementational analyses," in *Brain Structures, Learning and Memory*, Newburg, R., ed., , 1985, also appears as :Computer Science Department Technical Report 85-10. University of California, Irvine.
- [133] Greiner, R. and Genesereth, M. "The Role of Abstractions in Understanding Analogy". Working Paper HPP84-8, Stanford University, Dept. of Computer Science, April 1984.
- [134] Greiner, R., *Learning by Understanding Analogies*, PhD dissertation, Stanford University, 1985, forthcoming.
- [135] Haggerty, J., "REFEREE and RULECRITIC: Two Prototypes for Assessing the Quality of a Medical Paper," Master's thesis, Stanford University, Dept. of Computer Science, 1984.
- [136] Halasz, F. and Moran, T. "Analogy Considered Harmful." In *Human Factors in Computer Systems*. National Bureau of Standards, Gaithersburg, Maryland, March, 1982.
- [137] Hall, R., "Analogical reasoning in artificial intelligence and related disciplines", unpublished, University of California, Irvine.
- [138] Hall, R., Wenger, E., Kibler, D. and Langley, P. "The effect of multiple knowledge sources on learning and teaching". Technical report TR-85-11, University of California, Irvine, 1985.
- [139] Hart, P., Nilsson, N. and Raphael, B. "A formal basis for the heuristic determination of minimum cost paths." *IEEE Transactions on Systems Science and Cybernetics*. 4:2 (1968) 100-107.
- [140] Hawkins, R. and Kandel, E. "Is There a Cell-Biological Alphabet for Simple Forms of Learning?" *Psychological Review*. 91:3 (1984) 375-391.
- [141] Hayes, J. and Simon, H. "Understanding written problem instructions." In *Knowledge and Cognition*. Gregg, L., ed., Erlbaum, Potomac, MD, 1974.
- [142] Hayes-Roth, F., "The Role of Partial and Best Matches in Knowledge Systems," in *Pattern-Directed Inference Systems*, Waterman, D. and Hayes-Roth, F., eds., Academic Press, New York, 1978, 557-574.

- [143] Hayes-Roth, R. and McDermott, J. "An Interference Matching Technique for Inducing Abstractions." *CACM*. 21 (1978) 401-411.
- [144] Hayes-Roth, B., and Hewett, M. "Learning Control Heuristics in a Blackboard Environment". Technical report HPP-85-2. Stanford, Ca.: Stanford University, 1985.
- [145] Herrenstein, R., Phil. Trans. Royal Society London 308, pages 129-144.
- [146] Hesse, M. *Models and Analogies in Science*. Notre Dame: University of Notre Dame Press, 1966.
- [147] Hirsh, H.. "Modeling problem solving performance," Master's thesis, Stanford University, June 1985.
- [148] Holland, J. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [149] Holte, R., "title not supplied," in *Advanced Digital Information Systems*, Prentice Hall, 1985, 309-498.
- [150] Holte, R., "A Conceptual Framework for Analyzing and Comparing Practical Algorithms for the Machine Learning Task of Concept Identification", forthcoming Ph.D. thesis.
- [151] Holyoak, K. "Analogical Thinking and Human Intelligence", Technical report 40, University of Michigan, April 1982.
- [152] Horning, J. "A Study of Grammatical Inference". Technical report CS 139, Stanford Artificial Intelligence Project, August 1969.
- [153] Hoyle, G. "Learning, using Natural Reinforcements, in Insect Preparations that Permit Cellular Neuronal Analysis." *Journal of Neurobiology*. 11:4 (1980) 323-354.
- [154] Hume, D.. Honours Thesis, University of New South Wales..
- [155] Hunt, E. *Experiments in Induction*. New York: Academic Press, 1966.
- [156] Hunt, G. "Admissible Hypotheses and Enhanced Learning." In *Proceedings of IJCAI-83*. Morgan Kaufmann, 1983, 444-446.
- [157] Iba, G. "Learning by Discovering Macros in Puzzle Solving." In *Proceedings of IJCAI-85*. Morgan Kaufmann, 1985.

- [158] Jardetzky, O., Lane, A., Lefevre, J., Lichtarge, O., Hayes-Roth, B. and Buchanan, B. "Determination of Macromolecular Structure and Dynamics by NMR." In *Proceedings of the NATO Advanced Study Institute, 'NMR in the Life Sciences'*, June-17-27, 1985. Plenum Publishing Corp., 1985. forthcoming.
- [159] Johnson, N., "Organization and the concept of a memory code," in *Coding Processes in Human Memory*, Melton, A. and Martin, E., eds., Winston, Washington, D.C., 1972.
- [160] Kahn, G., Nowlan, S. and McDermott, J. "Strategies for knowledge acquisition." *IEEE Pattern Recognition and Machine Intelligence*. September (1985) .
- [161] Kahn, G., Nowlan, S. and McDermott, J. "MORE: An intelligent knowledge acquisition tool." In *Proceedings of IJCAI-85*. Morgan Kaufmann, 1985.
- [162] Kandel, E. *Behavioral Biology of Aplysia*. San Francisco: W.H. Freeman, 1979.
- [163] Kandel, E. and Schwartz, J. "Molecular Biology of Learning: Modulation of Transmitter Release." *Science*. 218 October (1982) 433-443.
- [164] Kedar-Cabelli, S. "Purpose-Directed Analogy." In *Proceedings of the Cognitive Science Society*. Irvine, CA, August, 1985.
- [165] Keller, R. "Learning by Re-expressing Concepts for Efficient Recognition." In *Proceedings AAAI-83*. Washington, D.C., August, 1983, 182-186.
- [166] Keller, R.. "Sources of Contextual Knowledge for Concept Learning". Unpublished doctoral dissertation proposal, July 1984, Rutgers University Department of Computer Science.
- [167] Keller, R., *Contextual Learning: A Performance-based Model of Concept Acquisition*, PhD dissertation, Rutgers University, 1986, forthcoming.
- [168] Kelly, G. *The Psychology of Personal Constructs*. New York: Norton, 1955.
- [169] Kibler "Perturbation: A Means for Guiding Generalization." In *Proceedings of IJCAI-83*. William Kaufmann, 1983, 415-418.

- [170] Kneipp, W., Honours Thesis, University of New South Wales..
- [171] Ko, H., "Controlling Attention in Inference Machine: Prolog", to be published as ISG report, University of Illinois at Urbana-Champaign.
- [172] Kodratoff, Y., "Generalizing and Particularizing as the Techniques of Learning," in *Computers and Artificial Intelligence*, , 1983, 417-441.
- [173] Koestler, A. *The Act of Creation: A Study of the Conscious and Unconscious in Science and Art*. Macmillan, 1964, either this edition or the Dell (1964) edition as other editions do not contain the important Book Two.
- [174] Kokar, M. "Learning Arguments of Invariant Functional Descriptions", Technical report, Northeastern University, College of Engineering, year not supplied.
- [175] Kokar, M. "The Use of Dimensional Analysis for Choosing Parameters Describing a Physical Phenomenon." *Bulletin de l'Academie Polonaise des Sciences, Serie de Sciences Techniques*. XXVII:5/6 (year not supplied) 249.
- [176] Kolodner, J. "Towards an Understanding of the Role of Experience in the Evolution from Novice to Expert." *International Journal of Man-Machine Systems*. November (1983) .
- [177] Kolodner, J. *Retrieval and Organizational Strategies in Conceptual Memory: a computer model*. Hillsdale, NJ: Lawrence Erlbaum, 1984, also appears as PhD dissertation, Yale University, 1980.
- [178] Kolodner, J. and Simpson, R. "Experience and problem solving: a framework." In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*. the Cognitive Science Society, 1984.
- [179] Kolodner, J., Simpson, R. and Sycara-Cyranski, K. "A Process model of case-based reasoning in Problem Solving." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, Calif., 1985, 284-290.
- [180] Konolige, K., "A First-order Formalization of Knowledge and Action for a Multi-agent Planning System," in *Machine Intelligence*, J. Hayes, ed., Ellis Horwood, 1982.
- [181] Korf, R. "A Program that learns to solve Rubik's Cube." In *AAAI-82*. Morgan Kaufmann, 1982, 164-167.

- [182] Korf, R. "Macro-operators: A weak method for learning." *Artificial Intelligence*. 26 (1985) 35-77.
- [183] Korf, R. "Depth-first iterative-deepening: An optimal admissible tree search." *Artificial Intelligence*. 27 (1985) .
- [184] Korf, R. "Toward a Model of Representation Changes." *Artificial Intelligence*. 14 (1980) 41-78.
- [185] Korf, R., *Learning to Solve Problems by Searching for Macro-Operators*, PhD dissertation, Carnegie-Mellon University, July 1983.
- [186] Kowalski, R. *Logic for Problem Solving*. North-Holland, 1979.
- [187] Krasne, F. and Bryan, J. "Habituation: Regulation through Presynaptic Inhibition." *Science*. 182 (1973) 590-592.
- [188] Laird, J., and Newell, A. "A Universal Weak Method", Technical report #83-141, Carnegie-Mellon University Computer Science Department, June 1983.
- [189] Laird, J., Newell, A., and Rosenbloom, P., "Soar: An Architecture for General Intelligence", In preparation.
- [190] Laird, J., Rosenbloom, P., and Newell, A. "Towards chunking as a general learning mechanism." In *Proceedings of AAAI-84*. Morgan Kaufmann, Austin, 1984, (Also available as part of Carnegie-Mellon University Computer Science Tech. Rep. #85-110).
- [191] Laird, J., Rosenbloom, P., and Newell, A., "Chunking in Soar: The anatomy of a general learning mechanism". In preparation.
- [192] Lakatos, I. *Proofs and Refutations*. Cambridge University Press, 1976.
- [193] Lakoff, G. and Johnson, M. *Metaphors We Live By*. Chicago: The University of Chicago Press, 1980.
- [194] Langley, P. "BACON: A Production System That Discovers Natural Laws." In *Proceedings of IJCAI-77*. Morgan Kaufmann, 1977.
- [195] Langley, P. and Neches, R. "Prism User's Manual", Technical report, Department of Computer Science, Carnegie-Mellon University, 1981.

- [196] Langley, P., Nicholas, D., Klahr, D. and Hood, G. "A Simulated World for Modeling Learning and Development." In *Proc. 3rd Annual Conference of the Cognitive Science Society*. 1981.
- [197] Langley, P. "Data Driven Discovery of Physical Laws." *Cognitive Science*. 5:1 (1981) 31-54.
- [198] Langley, P. "Learning effective search heuristics." In *Proceedings of IJCAI-83*. Morgan Kaufmann, 1983, 419-421.
- [199] Langley, P., and Sage, S. "Conceptual Clustering as Discrimination Learning." In *Proceedings of the Fifth Conference of the Canadian Society for Computational Studies of Intelligence*. Canadian Society for Computational Studies of Intelligence, 1984.
- [200] Langley, P., Zytkow, J., Simon, H., and Bradshaw, G., "The Search for Regularity: Four Aspects of Scientific Discovery," in *Machine Learning*, Tioga, 1983.
- [201] Langley, P., Ohlsson, S., and Sage, S. "A machine learning approach to student modeling", Technical Report CMU-RI-TR-84-7, Robotics Institute, Carnegie-Mellon University, 1984.
- [202] Larkin, J., "Enriching Formal Knowledge: A Model for Learning to Solve Textbook Physics Problems," in *Cognitive Skills and their Acquisition*, J. Anderson, ed., Lawrence Erlbaum, Hillsdale, N.J., 1981, 311-334.
- [203] Lawler, R. *Computer Experience and Cognitive Development*. Chichester, England and New York: Ellis Horwood Ltd. and John Wiley, 1985.
- [204] Lawler, R.. "The Internalization of External Processes", unpublished.
- [205] Lawler, R., and Selfridge, O. "Learning Concrete Strategies through Interaction." In *Proceedings of the Cognitive Science Society Annual Conference*. Cognitive Science Society, 1985.
- [206] Lebowitz, M., *Generalization and Memory*, PhD dissertation, Yale University, 1980.
- [207] Lebowitz, M. "Generalization from natural language text." *Cognitive Science*. 7:1 (1983) 1 - 40.

- [208] Lebowitz, M. "Concept learning in a rich input domain." In *Proceedings of the 1983 International Machine Learning Workshop*. Champaign-Urbana, Illinois, 1983, 177 - 182. To appear in *Machine Learning II*.
- [209] Lebowitz, M. "Integrated learning: Controlling explanation." *Cognitive Science*. (1985) , to appear.
- [210] Lenat, D., "AM: Discovery in Mathematics as Heuristic Search," in *Knowledge-Based Systems in Artificial Intelligence*, Randall Davis and Douglas B. Lenat, eds., McGraw-Hill Book Co., New York, 1982. Based on Phd thesis, Stanford University, Stanford, CA, 1977.
- [211] Lenat, D. "The nature of heuristics." *Artificial Intelligence*. 19 (1982) 189-249.
- [212] Lenat, D. "Theory Formation by Heuristic Search. The Nature of Heuristics II: Background and Examples." *Artificial Intelligence*. 21:1-2 (1983) 31-59.
- [213] Lenat, D., Hayes-Roth, F. and Klahr, P. "Cognitive Economy in a Fluid Task Environment." In *International Machine Learning Workshop*. R. Michalski, ed., University of Illinois at Urbana Champaign, 1983.
- [214] Lenat, D. "EURISKO: A Program That Learns New Heuristics and Domain Concepts. The Nature of Heuristics III: Program Design and Results." *Artificial Intelligence*. 21 (1983) 61-98.
- [215] Lenat, D., and Brown, J. "Why AM and EURISKO Appear to Work." *Artificial Intelligence*. 23:3 (1984) 269-294.
- [216] Lewin, K. *A Dynamic Theory of Personality*. New York: McGraw-Hill, 1935. Translated by Adams and Zener.
- [217] Lindsay, R., Buchanan, B., Feigenbaum, E., and Lederberg, J. *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project*. New York: McGraw-Hill, 1980.
- [218] Luce, R. "Dimensionally Invariant Physical Laws Correspond to Meaningful Relations." *Philosophy of Science*. 45 (1978) 1-16.
- [219] MacDonald, A.. *Personal Construct Theory: A Computational Perspective*, PhD dissertation, Brunel University, 1984.
- [220] Mackie, J. *The Cement of the Universe: A Study of Causation*. Oxford University Press, 1974.

- [221] Mahadevan, S. "Verification-Based Learning: A Generalization Strategy for Inferring Problem-Decomposition Methods." In *Proceedings of IJCAI-85*. Morgan Kaufmann, 1985. also appears as Technical report LCSR-TR66, Rutgers University, January 1985.
- [222] Mauldin, M. "Maintaining Diversity in Genetic Search." In *Proceedings of AAAI-84*. Morgan Kaufmann, 1984.
- [223] Mayer, R. "Frequency norms and structural analysis of algebra story problems." *Instructional Science*. 10 (1981) 135-175.
- [224] McCarty, L. and Sridharan, N. "The Representation of an Evolving System of Legal Concepts II. Prototypes and Deformations." In *Proceedings of IJCAI-83*. Morgan Kaufmann, Vancouver, B.C., Canada, 1981, 246-253.
- [225] McCarty, L. and Sridharan, N. "A Computational Theory of Legal Argument", Technical report LPR-TR-13, Laboratory for Computer Science Research, Rutgers University, January 1982.
- [226] Medin, D., Wattenmaker, W., and Michalski, R., "Constraints in Inductive Learning: An Experimental Study Comparing Human and Machine Performance", submitted to *Cognitive Science*.
- [227] Michalski, R., "Variable-valued logic and its applications to pattern recognition and machine learning," in *Computer science and multiple-valued logic: Theory and applications*, D. Rine, ed., North-Holland, Amsterdam, 1975, 506-534.
- [228] Michalski, R. and Chilausky R. "Knowledge Acquisition by Encoding Expert Rules Versus Computer Induction from Examples: A Case Study using Soybean Pathology." *International Journal for Man-Machine Studies*. 12:63 (1980) .
- [229] Michalski, R., "A Theory and Methodology of Inductive Learning," in *Machine Learning*, Tioga, 1983, 83-134.
- [230] Michalski, R., and Larson, J. "Selection of most representative training examples and incremental generation of VL<sub>1</sub> hypotheses: the underlying methodology and the description of programs ESEL and AQ11", Technical Report 867, Computer Science Department, University of Illinois, 1978.

- [231] Michalski, R. "Variable-valued Logic: System VL<sub>1</sub>." In *Proceedings of the 1974 International Symposium on Multiple-Valued Logic*. West Virginia University, Morgantown, West Virginia, May 29-31, 1974, 323-346.
- [232] Michalski, R., "personal correspondence to L. Valiant".
- [233] Michalski, R., Carbonell, J., and Mitchell, T. *Machine Learning: An Artificial Intelligence Approach*. Tioga, 1983.
- [234] Michalski, R., and Chilausky, R. "Learning by Being Told and Learning from Examples: an Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis." *Policy Analysis and Information Systems*. 4:2 (1980) .
- [235] Michalski, R., Ko, H. and Chen, K. "Symbolic Process Prediction: A Method and a Program SPARC/G", Technical report ISG 85-12, UIUCDCS-F-85-942, University of Illinois, 1985.
- [236] Michalski, R., Stepp, R., "Learning from Observation: Conceptual Clustering," in *Machine Learning*, Tioga, 1983.
- [237] Michalski, R., and Stepp, R., "How to Structure Structured Objects," in *Proceedings of the International Machine Learning Workshop*, University of Illinois, at Urbana-Champaign, 1983, 156-160.
- [238] Miller, G. "The magic number seven plus or minus two: Some limits on our capacity for processing information." *Psychological Review*. 63 (1956) 81-97.
- [239] Minsky, M. and Papert, S. *Perceptrons*. Cambridge, Mass.: MIT Press, 1969.
- [240] Minsky, M.. "A Framework for Representing Knowledge." in *The Psychology of Computer Vision*, P. Winston, ed., McGraw-Hill, New York, 1975, 211-277.
- [241] Minsky, M., "Learning Meaning", unpublished.
- [242] Minton, S. "Selectively Generalizing Plans for Problem-Solving." In *Proceedings of IJCAI-85*. Joshi, A., ed., Morgan Kaufmann, 1985.
- [243] Minton, S. "Constraint Based Generalization- Learning Game Playing Plans from Single Examples." In *Proceedings of AAAI*. 1984.

- [244] Mitchell, T. "Version spaces: A candidate elimination approach to rule learning." In *Proceedings of IJCAI-77*. Morgan Kaufmann, 1977, 305-310.
- [245] Mitchell, T., *Version Spaces: An approach to concept learning.*, PhD dissertation, Stanford University, 1978.
- [246] Mitchell, T. "The Need for Biases in Learning Generalizations". Technical report CBM-TR-117, Rutgers University, 1980.
- [247] Mitchell, T., Utgoff, P., Nudel, B. and Banerji, R. "Learning Problem-Solving Heuristics through Practice." In *Proceedings IJCAI-7*. Morgan Kaufmann, Vancouver, B.C., Canada, August, 1981, 127-134.
- [248] Mitchell, T. "Generalization as Search." *Artificial Intelligence*. 18:2 March (1982) 203-226.
- [249] Mitchell, T. and Keller, R. "Goal Directed Learning." In *Proceedings of the Second International Machine Learning Workshop*. Urbana, Illinois, June, 1983.
- [250] Mitchell, T. "Learning and Problem Solving." In *Proceedings IJCAI-8*. Morgan Kaufmann, Karlsruhe, West Germany, August, 1983, 1139-1151.
- [251] Mitchell, T. "An intelligent aid for circuit redesign." In *Proceedings of the Third National Conference on Artificial Intelligence*. Washington, DC, 1983.
- [252] Mitchell, T., Steinberg, L., and Shulman, J. "A Knowledge-Based Approach to Design." In *Proceedings of the IEEE Workshop of Principles of Knowledge-Based Systems*. IEEE, December, 1984, 27-34. Revised version to appear in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September, 1985.
- [253] Mitchell, T., "Toward Combining Empirical and Analytic Methods for Learning Heuristics," in *Human and Artificial Intelligence*, Elithorn, A. and Banerji, R., eds., Erlbaum, 1984, also appears as Technical report LCSR-TR-27, Rutger's University, 1982.
- [254] Mitchell, T., Mahadevan, S. and Steinberg, L. "Leap: A learning apprentice for VLSI design." In *Proceedings of IJCAI-85*. Morgan Kaufmann, August, 1985. Also appears as Rutgers University technical report LCSR-TR-64, 1985.

- [255] Mitchell, T., Keller, R. and Kedar-Cabelli, S. "Explanation-Based Generalization: A unifying view." *Machine Learning*. 1:1 January (1986) .
- [256] Mitchell, T., Utgoff, P., and Banerji, R.. "Learning by experimentation: Acquiring and refining problem-solving heuristics." in *Machine Learning: An Artificial Intelligence Approach*, R. Michalski, J. Carbonell, T. Mitchell, eds., Tioga Publishing Co., Palo Alto, CA, 1983, 163-190.
- [257] Mooney, R., "Generalizing Explanations of Narratives into Schemata," Master's thesis, University of Illinois at Urbana-Champaign, May 1985. Also appears as Technical Report T-159, AI Research Group, Coordinated Science Laboratory.
- [258] Mostow, D., "Transforming declarative advice into effective procedures: a heuristic search example," in *Machine Learning*, Michalski, R., Carbonell, J. and Mitchell, T., eds., Tioga Press, Palo Alto, CA., 1982.
- [259] Mostow, J. "Rutgers Workshop on Knowledge-Based Design." *SIGART Newsletter*. 90 October (1984) 19-32.
- [260] Mostow, J. "Toward better models of the design process." *AI Magazine*. 6:1 Spring (1985) 44-57.
- [261] Mostow, J. "Response to Derek Partridge." *AI Magazine*. 6:3 Fall (1985) 51-52.
- [262] Mostow, J. "Some Requirements for Effective Replay of Derivations." In *Proceeding of the Third International International Machine Learning Workshop*. Rutgers University, Skytop, Pa., June, 1985, 129-132.
- [263] Mozer, M., "An Expectation-Driven Event Interpreter", unpublished.
- [264] Mozetic I. "Compression of the ECG Knowledge-base Using the AQ Inductive Learning Algorithm", Technical report Report No. UIUCDCS-F-85-943, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1985.
- [265] Nagel, D. "Concept Learning by Building and Applying Transformations Between Object Descriptions", Technical report LRP-TR-15, Rutgers University, June 1983.

- [266] Newell, A. and Rosenbloom, P., "Mechanisms of skill acquisition and the law of practice," in *Cognitive Skills and Their Acquisition*, J. Anderson, ed., Erlbaum, Hillsdale, NJ, 1981, 1-55. (Also available as Carnegie-Mellon University Computer Science Tech. Rep. #80-145).
- [267] Newell, A. and Simon, H. *Human Problem Solving*. Englewood Cliffs: Prentice-Hall, 1972.
- [268] Newell, A., Shaw, J. and Simon, H. "Report on a general problem-solving program for a computer." In *Proceedings of the International Conference on Information Processing*. UNESCO, Paris, 1959, 256-264.
- [269] Nilsson, N. *Principles of Artificial Intelligence*. Palo Alto: Tioga, 1980.
- [270] Okabe, H. "Formal Expressions of Infinite Graphs and Their Families." *Information and Control*. 44 (1980) 164-186.
- [271] O'Rorke, P. "Generalization for Explanation-based Schema Acquisition." In *Proceedings of AAAI-84*. Morgan Kaufmann, 1984.
- [272] Pao, Tsyh-Wen Lee "A Solution of the Syntactical Induction-Inference Problem for a Non-Trivial Subset of Context Free Languages". Technical report 70-19, Moore School of Electrical Engineering, University of Pennsylvania, August 1969.
- [273] Piaget, J. *The Child's Conception of Number*. New York: W. Norton and Co., 1952.
- [274] Piaget, J. *Biology and Knowledge*. Chicago: University of Chicago Press, 1971.
- [275] Piaget, J. *The Psychology of Intelligence*. Totowa, NJ: Littlefield, Adams and Co., 1976.
- [276] Pigot, H. and Soldano, H. "Mechanismes d'apprentissage complexes:application à la reconnaissance des voyelles", Technical report, INRIA, Forthcoming.
- [277] Pigot, H.. *Apprentissage binaire de noyaux vocaliques*, PhD dissertation, Paris 6 University, May 1985, 3<sup>e</sup> cycle Thesis.
- [278] Pitrat, J.. "A Program for Learning to Play Chess," in *Pattern Recognition and Artificial Intelligence*, Academic press, 1976.

- [279] Porter, B. and Kibler, D. "Learning Operator Transformations." In *Proceedings of the National Conference on Artificial Intelligence*. 1984.
- [280] Porter, B. "Using and Revising Learned Concept Models: A Research Proposal". Technical report 85-03. University of Texas at Austin, 1985, Artificial Intelligence Lab.
- [281] Powers, D., to appear as Ph.D. Thesis.
- [282] Quinlan, J., "Learning Efficient Classification Procedures and their Application to Chess End Games," in *Machine Learning: An Artificial Intelligence Approach*, Michalski, R., Carbonell, J., and Mitchell, T., ed., Tioga Press, 1983.
- [283] Quinqueton, J. and Sallantin, J. "Algorithms for learning logical formulas." In *Proceedings of IJCAI-83*. Morgan Kaufmann, 1983.
- [284] Quinqueton, J. and Sallantin, J., "CALM (Contester pour Apprendre en Logique Modale)", accepted for AFCET-INRIA congress on Pattern Recognition and Artificial Intelligence, Grenoble, France.
- [285] Quinqueton, J. and Sallantin, J. "Learning techniques in knowledge based systems." In *Proceedings of COGNITIVA85*. Paris, June, 1985.
- [286] Rajamoney, S., DeJong, G., and Faltings, B. "Towards a Model of Conceptual Knowledge Acquisition through Directed Experimentation." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, CA, 1985.
- [287] Rappaport, A. and Chauvet, J. "Symbolic Knowledge Processing for the Acquisition of Expert Behavior: a Study in Medicine", Technical report, The Robotics Institute, Carnegie-Mellon University, CMU-TR-RI-8-84.
- [288] Reinke, R., "Knowledge Acquisition and Refinement Tools for the ADVISE Meta-expert System," Master's thesis, University of Illinois at Urbana-Champaign, 1984, Report no. UIUCDCS-F-84-921, Dept. of Computer Science.
- [289] Rendell, L. "A method for automatic generation of heuristics for state-space problems", Technical report CS-76-10, University of Waterloo, 1976.
- [290] Rendell, L., *An adaptive plan for state-space problems*, PhD dissertation, University of Waterloo, 1981, CS-81-13.

- [291] Rendell, L. "State-space learning systems using regionalized penetrance." In *Proceedings of the Fourth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*. 1982, 150-157.
- [292] Rendell, L. "A new basis for state-space learning systems and a successful implementation." *Artificial Intelligence*. 20 (1983) 369-392.
- [293] Rendell, L. "A learning system which accommodates feature interations." In *Proceedings of IJCAI-83*. Morgan Kaufmann, 1983, 469-472.
- [294] Rendell, L. "A doubly layered, genetic penetrance learning system." In *Proceedings of the Third National Conference on Artificial Intelligence*. AAAI, 1983, 343-347.
- [295] Rendell, L. "Towards a unified approach for conceptual knowledge acquisition." *AI Magazine*. 4:4 Winter (1983) 19-27.
- [296] Rendell, L. "Conceptual knowledge acquisition in search", Technical report CIS-83-15, University of Guelph, November 1983, to appear in Bolc, L. (ed.), *Knowledge Based Learning Systems*, Springer-Verlag.
- [297] Rendell, L. "Utility patterns as criteria for efficient generalization learning." In *Proceedings 1985 Conference on Intelligent Systems and Machines*. , 1985, to appear.
- [298] Rendell, L. "Substantial constructive induction using layered information compression: Tractable feature formation in search." In *Proceedings of IJCAI-85*. Morgan Kaufmann, 1985.
- [299] Rendell, L. "Genetic plans and the probabilistic learning system: Synthesis and results." In *Proceedings of the International Conference on Genetic Algorithms and their Applications*. Carnegie-Mellon University, 1985, to appear.
- [300] Rescorla, R. "Probability of shock in the presence and absence of CS in fear conditioning." *Journal of Comparative and Physiological Psychology*. 66 (1968) 1-5.
- [301] Rich, C. and Waters, R. "Abstraction, Inspection and Debugging in Programming", AI Memo 634, Massachusetts Institute of Technology, June 1981.
- [302] Riddle, P., "A Proposal for Automating Shifts of Representation", Forthcoming Rutgers Technical Report.

- [303] Rosenbloom, P., *The Chunking of Goal Hierarchies: A Model of Practice and Stimulus-Response Compatibility*, PhD dissertation, Carnegie-Mellon University, 1983. (Available as Carnegie-Mellon University Computer Science Tech. Rep. #83-148).
- [304] Rosenbloom, P., Laird, J., McDermott, J., Newell, A. and Orciuch, E. "R1-Soar: An Experiment in Knowledge-Intensive Programming in a Problem-Solving Architecture." In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*. IEEE, October, 1984.
- [305] Rosenbloom, P. and Newell, A. "Learning by chunking: A production-system model of practice". Technical report #82-135, Carnegie-Mellon University Computer Science Department, September 1982.
- [306] Rosenbloom, P. and Newell, A. "Learning by chunking: Summary of a task and a model." In *Proceedings of AAAI-82*. American Association for Artificial Intelligence, Pittsburgh, 1982.
- [307] Rosenbloom, P., and Newell, A., "The chunking of goal hierarchies: A generalized model of practice," in *Machine Learning: An Artificial Intelligence Approach, Volume II*, R. Michalski, J. Carbonell, and T. Mitchell, eds., Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1985, In press (Also available in *Proceedings of the Second International Machine Learning Workshop*, Urbana: 1983).
- [308] Sacerdoti, E. "Planning in a hierarchy of abstraction spaces." *Artificial Intelligence*. 5 (1974) 115-135.
- [309] Sacerdoti, E. *A Structure for Plans and Behavior*. New York: Elsevier, 1977. Also appears as SRI technical report 109, 1975.
- [310] Sammut, C., *Learning Concepts by Performing Experiments*, PhD dissertation, University of New South Wales, 1981.
- [311] Sammut, C. and Banerji, R., "Hierarchical Memories: An Aid to Concept Learning," in *Machine Learning II*, Morgan Kaufmann, 1986, 74-80.
- [312] Samuel, A., "Some studies in machine learning using the game of checkers," in *Computers and Thought*, McGraw-Hill, N.Y., 1963.
- [313] Samuel, A. "Some studies in machine learning using the game of checkers II - recent progress." *IBM Journal of Research and Development*. 11 (1967) 601-617.

- [314] Sanderson, A., Segen, J. and Richey, E. "Hierarchical Modeling of EEG Signals." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. PAMI-2 (1980) 405-415.
- [315] Satinoff, N. "Neural Organization and the Evolution of Thermal Regulation in Mammals." *Science*. 201 July 7 (1978) 15-22.
- [316] Saul, R., "A SOAR2 Implementation of Version-Space Inductive Learning". Unpublished, 1984.
- [317] Schmidt, C., Sridharan, N. and Goodson, J. "The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence." *Artificial Intelligence*. 11 (1978) .
- [318] Schank, R. and Abelson, R. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1977.
- [319] Schank, R. "Looking at Learning." In *Proceedings of the Fifth European Conference on AI*. Paris, France, July, 1982, 11-18.
- [320] Schank, R. *Dynamic Memory- A theory of reminding and learning in computers and people*. Cambridge, Massachusetts: Academic Press, 1982, also published by Cambridge Press, New York, 1983.
- [321] Schank, R. and Collins, G., "Transcending Inductive Category Formation in Learning", unpublished.
- [322] Scherlis, W. and Scott, D. "First steps towards inferential programming." In *IFIP Congress 83*. North-Holland, 1983, Invited paper.
- [323] Segen, J.. *Pattern-Directed Signal Analysis*, PhD dissertation, Carneqie-Mellon University, 1980.
- [324] Segen, J. "Learning Structural Descriptions of Shape." In *Proceedings of CVPR'85*. San Francisco, June, 1985, 96-99.
- [325] Segen, J. "Learning Concept Descriptions from Examples with Errors." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, Calif., August, 1985.

- [326] Serge, A.. and DeJong, G. "Explanation Based Manipulator learning: Acquisition of Planning Ability Through Observation". Technical report AI Research Group Working Paper 62, University of Illinois at Urbana-Champaign, 1985, also appears in the Proceedings of the IEEE International Conference on Robotics and Automation, March 1985.
- [327] Sejnowski, T., "personal correspondence to L. Valiant".
- [328] Shannon, C. "Programming a computer for playing chess." *Philosophical Magazine (Series 7)*. 41 (1950) 256-275.
- [329] Shapiro, E. *Algorithmic Program Debugging*. Cambridge, Mass.: Massachusetts Institute Technology Press, 1982.
- [330] Shapiro, E. "Inductive Inference of Theories From Facts". Research Report 192, Yale University, February 1981.
- [331] Shavlik, J. "Learning About Momentum Conservation." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, August, 1985.
- [332] Shavlik, J. and DeJong, G. "Building a Computer Model of Learning Classical Mechanics." In *Proceedings of 7th Annual Conference of the Cognitive Science Society*. Cognitive Science Society, Irvine, Ca., 1985.
- [333] Shortliffe, E., Scott, A., Bischoff, M., Campbell, A., Van Melle, W., and Jacobs, C. "ONCOCIN: An Expert System for Oncology Protocol Management." In *Proceedings of IJCAI-81*. Morgan Kaufmann, Vancouver, B C., August, 1981, 876-881.
- [334] Shrager, J. "Acquisition of Device Models in Instructionless Learning." In *First Annual Workshop on Theoretical Issues in Conceptual Information Processing*. Kolodner, J. and Riesbeck, C., ed., Georgia Inst. of Technology, Georgia, March, 1984, 10-18.
- [335] Shrager, J. *Instructionless Learning: Discovery of the Mental Model of a Complex Device*. PhD dissertation, Carnegie-Mellon University, August 1985.
- [336] Shrager, J. and Klahr, D. "A Model of Learning in the Instructionless Environment." In *Proceedings of the Conference on Human Factors in Computing Systems*. Association for Computing Machinery, December, 1983, 226-229.

- [337] Shrager, J. and Klahr, D., "Instructionless Learning: Discovery of the Mental Model of a Complex Device", In preparation.
- [338] Silver, B. "Learning Equation Solving Methods from Examples." In *Proceedings of IJCAI-83*. Bundy, A., ed.. Morgan Kaufmann, 1983, 429-431, Also available from Edinburgh as Research Paper 184.
- [339] Silver, B., *Using Meta-Level Inference To Constrain Search And To Learn Strategies In Equation Solving*, PhD dissertation, Dept. of Artificial Intelligence, Edinburgh, 1984.
- [340] Silver, B., "Precondition Analysis: Learning Control Information," in *Machine Learning 2*, Michalski, R., Carbonell, J. and Mitchell, T., eds., Morgan-Kaufmann, 1985.
- [341] Silver, B. *Meta-Level Inference*. North Holland, 1985.
- [342] Silver, B. "Learning Equation Solving Methods from Worked Examples." In *Proceedings of the 1983 International Machine Learning Workshop*. Michalski, R., Carbonell, J. and Mitchell, T., eds., University of Illinois, June 22-24, 1983, 99-104.
- [343] Simon, H. and Lea, G., "Problem solving and rule induction: A unified view," in *Knowledge and cognition*, Lawrence Erlbaum, 1974, 105-127.
- [344] Simpson, R., *A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation*, PhD dissertation, Georgia Institute of Technology, 1985, also available as Technical Report #GIT-ICS-85/18, School of ICS.
- [345] Singer, D., "Mechanismes d'apprentissage," pub. n° 41 CNRS-GR22, Universite Paris 6,These de 3<sup>e</sup> cycle.
- [346] Sleeman, D., Langley, P. and Mitchell, T. "Learning from solution paths: An approach to the credit assignment problem." *AI Magazine*. Spring (1982) 8-52.
- [347] Sleeman, D., and Smith, M. "Modeling student's problem solving." *Artificial Intelligence*. 16 (1981) 171-187.
- [348] Sleeman, D. "Inferring (mal) rules from pupils' protocols." In *Proceedings of the Second International Machine Learning Workshop*. June, 1983, 221-227.

- [349] Sleeman, D. "An attempt to understand student's understanding of basic algebra." *Cognitive Science*. 8 (1984) 387-412.
- [350] Smith, D. "Focuser: A strategic interaction paradigm for language acquisition". Technical report LCSR-TR-36. Rutgers University, 1982.
- [351] Smith, E. and Medin, D. *Categories and Concepts*. Cambridge, Massachusetts: Harvard University Press, 1981.
- [352] Smith, R., Mitchell, T., Chesek, R., and Buchanan, B. "A Model For Learning Systems." In *Proceedings of IJCAI-77*. Morgan Kaufmann, 1977, 338-343.
- [353] Smith, R., Winston, H., Mitchell, H. and Buchanan, B. "Representation and Use of Explicit Justification for Knowledge Base Refinement." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, Calif., 18-23 August, 1985, 673-680.
- [354] Soloway, E. "Knowledge-Directed Learning." *SIGART Newsletter*. 63 (1977) .
- [355] Soldano, H. and Moisy, J. "Statistico-syntactic learning techniques." *Biochemie*. 67 (1985) .
- [356] Soroka, B. "Some simple math for robot programs." In *ROBOTS-8 Conference*. Society of Manufacturing Engineers, Detroit, MI, 1984, Preprint from Robotics Institute, University of Southern California, Los Angeles, CA. 90089-0781.
- [357] Sridharan, N., Lantz, B., Bresina, J. and Goodson, J. "AIMDS Users Manual, Version 3", Technical report, Rutgers University, November 1981.
- [358] Sridharan, N. "A Flexible Structure for Knowledge." In *Proceedings of the International Conference on Cybernetics and Society*. Seattle, Washington, October, 1982.
- [359] Steels, L., "Second Generation Expert Systems," in *Future Generation Computer Systems*, North-Holland, Amsterdam, 1985.
- [360] Steels, L. and Van de Velde, W., "Learning in Second Generation Expert Systems," in *Knowledge Based Problem Solving*, Prentice-Hall, New Jersey, 1985.
- [361] Stefik, M. "Planning with constraints (MOLGEN: Part 1)." *Artificial Intelligence*. 16 (1981) 111:139.

- [362] Stepp, R., *Conjunctive Conceptual Clustering: A Methodology and Experimentation*, PhD dissertation, University of Illinois, at Urbana, 1984.
- [363] Stevens, A. and Collins, A., "Multiple conceptual models of a complex system," in *Aptitude, Learning and Instruction*, Erlbaum, 1980.
- [364] Sulston, J., Schierenberg, E., White, J., And Thomson, J. "The Embryonic Cell Lineage of the Nematode *Caenorhabditis elegans*." *Developmental Biology*. 100 (1983) 64-119.
- [365] Sussman, G. *A Computer Model of Skill Acquisition*. New York: American Elsevier, Artificial Intelligence, Vol. 1, 1975.
- [366] Suwa, M., Scott, A., and Shortliffe, E. "An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System." *AI Magazine*. 3:4 fall (1982) .
- [367] Tong, C., *Knowledge-based circuit design*, PhD dissertation, Stanford University Computer Science Department, 1985.
- [368] Tversky, A. and Gati, I. "Similarity, Separability, and the Triangle Inequality." *Psychological Review*. 89 (1982) 123-154.
- [369] Wilensky, R., Arens, Y. and Chin, D. "Talking to UNIX in English: an Overview of UC." *Communications of the ACM*. 27:6 June (1984) 574-593.
- [370] Utgoff, P. and Mitchell, T. "Acquisition of Appropriate Bias for Inductive Concept Learning." In *AAAI-82*. August, 1982, 414-417.
- [371] Utgoff, P. "Adjusting Bias in Concept Learning." In *Proceedings of IJCAI-83*. Morgan Kaufmann, 1983, 447-449, also appears in Proceedings International Machine Learning Workshop, 1983.
- [372] Utgoff, P., *Shift of Bias for Inductive Concept Learning*, PhD dissertation, Rutgers University, October 1984.
- [373] Utgoff, P. "Acquisition of Appropriate Bias for Inductive Concept Learning", Technical report LCSR-TM-2, Computer Science Dept., Rutgers University, 1982.
- [374] Valiant, L.. Communications Of The ACM 279:11, pages 1134-1142.
- [375] Valiant L.. Phil. Trans. Royal Society London 312:441-446..

- [376] Valiant, L. "Learning Disjunctions of Conjunctions." In *Proceedings of IJCAI-85*. Morgan Kaufmann, Los Angeles, Calif., August, 1985, 560-566.
- [377] VanLehn, K. "Human Skill Procedural Skill Acquisition: Theory, model and psychological validation." In *Proceedings of AAAI-83*. AAAI, 1983.
- [378] VanLehn, K. "Felicity conditions for skill acquisition: Validating an AI-based theory", Technical report CIS-21, Xerox Palo Alto Research Centers, 1983.
- [379] VanLehn, K., "Learning one subprocedure per lesson", submitted for publication.
- [380] Vere, S. "Induction of concepts in the predicate calculus." In *Proceedings of IJCAI-75*. Morgan Kaufmann, 1975, 281-287.
- [381] Vere, S. "Induction of Relational Productions in the Presence of Background Information." In *Proceedings of IJCAI-77*. Morgan Kaufmann, M.I.T., 1977, 349-355.
- [382] Vere, S., "Constrained N-to-1 Generalizations", unpublished.
- [383] Vygotsky, L. *Mind in Society*. Cambridge, Mass.: Harvard University Press, 1978.
- [384] Waldinger, R., "Achieving several goals simultaneously," in *Machine intelligence 8*, Halstead/Wiley, 1977.
- [385] Wasserman, K., *Unifying representation and generalization: Understanding hierarchically structured objects*, PhD dissertation, Columbia University Department of Computer Science, 1985.
- [386] Watanabe, S. *Knowing and Guessing: A Formal and Quantitative Study*. Wiley, 1969.
- [387] Waterman, D.. "Serial Pattern Acquisition: A Production System Approach," in *Pattern Recognition and Artificial Intelligence Approach*, Chen, ed., , 1977.
- [388] Wenger, E., "AI and the communication of knowledge: an overview of intelligent teaching systems", unpublished, University of California, Irvine.

- [389] Wharton, R. "Grammar Enumeration and Inference." *Information and Control.* 33 (1977) 253-272.
- [390] Wile, D. "Program developments: formal explanations of implementations." *CACM.* 26:11 (1983) . Available from USC Information Sciences Institute as RR-82-99.
- [391] Wilensky, R., "Points: A Theory of Structures of Stories in Memory," in *Strategies for Natural Language Processing*, Lehnert, W. and H. Ringle, ed., Laurence Erlbaum, 1982.
- [392] Wilensky, R. *Planning and Understanding: A Conceptual Approach to Human Reasoning*. Reading, MA: Addison-Wesley, 1983.
- [393] Wilkins, D., Buchanan, B., Clancey, W. "Inferring an Expert's Reasoning by Watching." *Proceedings of the 1984 Conference on Intelligent Systems and Machines.* (1984) , also appears as HPP Report HPP-84-29.
- [394] Wilkins, D., Buchanan, B. and Clancey, W. "Inferring an expert's model by watching." In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*. April, 1984, 51-59.
- [395] Wilkins, D. "Patterns and Plans in Chess." *Artificial Intelligence.* 14 (1980) .
- [396] Winston, P., "Learning Structural Descriptions from Examples," in *The Psychology of Computer Vision*, Winston, P., ed., McGraw Hill, New York, 1975, also available as :Technical Report AI-TR-231, Massachusetts Institute of Technology.
- [397] Winston, P. "Learning by Understanding Analogies". Artificial Intelligence Laboratory Memo 520, MIT, January 1980.
- [398] Winston, P. "Learning New Principles From Precedents and Exercises: The Details", Artificial Intelligence Laboratory Memo 632, MIT, November 1981.
- [399] Winston, P. "Learning New Principles from Precedents and Exercises." *Artificial Intelligence.* 19:3 November (1982) 321-350.
- [400] Winston, P. "Learning by augmenting rules and accumulating Censors." In *International Machine Learning Workshop*. Michalski, R., ed., 1983, 2-10.

- [401] Winston, P., Binford, T., Katz, B., and Lowry, M. "Learning Physical Descriptions From Functional Definitions, Examples, and Precedents." In *AAAI-83*. AAAI, 1983, 433-439.
- [402] Younger, M. *A Handbook for Linear Regression*. North Scituate, Massachussets: Duxbury press, 1979.
- [403] Zernik, U. and Dyer, M. "Failure-Driven Acquisition of Figurative Phrases by Second Language Speakers." In *Proceedings of the 7th Annual Conference of the Cognitive Science Society*. August, 1985.
- [404] Zernik, U. and Dyer, M. "Towards a Self-Extending Phrasal Lexicon." In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*. July, 1985.

# INDEX

- A\*, 161  
Abduction, 379  
Abstraction  
  abstraction planning, 281  
  in analogical learning, 25, 81  
  levels of, 25  
  of rules, 353  
Acquired problem-solving knowledge  
  abstract rules, 353  
  conceptual vocabulary, 85, 137, 255  
  control knowledge, 199, 237, 317  
  design histories, 213  
  design plans, 343  
  diagnostic domain knowledge, 119, 227  
  differentially stored cases, 1  
  domain rules, 93, 203, 237, 379  
  episodic memory, 155  
  evaluation functions, 39, 269, 295, 343  
  explanatory schemata, 207  
  heuristics, 167  
  incorrect domain rules, 93  
  macro-operators, 115, 237, 275, 281  
  physics formulae, 307  
  problem schemata, 85, 103, 137, 237, 255  
  procedures, 35, 167, 359  
  prototypes, 241  
  strategies, 173  
Acquiring problem-solving knowledge  
  from experts, 241  
  to meet performance objectives, 127  
  via analogy, 137, 155  
  via experience, 1, 137, 237, 343  
  via instruction, 137, 241, 359  
Aggregation (prior to concept learning), 59, 67  
Analogical learning, 85, 137, 233  
  and generalization, 281  
  during problem-solving, 155  
  purpose-directed, 123  
  using abstraction, 25, 81  
Analytical learning, combined with  
  empirical learning, 55, 321, 375  
Architecture, learning system  
  based on an explicit model of  
    performance, 127  
  of an organism in a simple physical  
    world, 219  
SOAR, 281  
  using cooperating knowledge sources, 137  
  
Bias  
  automatic shift of, 67  
  dynamic, produced by task utility, 269  
  explicitness of, 183  
  simplicity of, 71  
  *see also* Hierarchies, generalization  
Case-based reasoning, 1, 155, 233  
Causal reasoning, 55, 119, 123, 207  
Characterization, *see* Concept learning  
Chunking, 281  
Cluster analysis, 269  
Cognitive map construction, 167  
Cognitive modeling of  
  concreteness, 173  
  contingency, 75  
  egocentricity, 173  
  human skill acquisition, 359  
  interactivity, 173  
  natural problem-solving, 155  
Compilation, 71, 199, 343, 353  
  *see also* Operationalization  
Compiled foresight, 343  
Completing explanation, learning by, 359  
Composition, 75, 115  
Concept hierarchy, *see* Hierarchies,  
  generalization  
Concept learning, 35, 59, 67, 99, 167, 179,  
  233, 299  
  assessment of learning systems for, 269  
  complexity of, 349  
  data driven, 11  
  efficiency and effectiveness of, 269  
  instance-to-class, 141  
  model-driven, 141  
  part-to-whole, 141  
  using concept models, 241  
Conceptual clustering, 67, 269, 333  
  higher dimensional, 269  
  utility-based, 269  
Conceptual framework, 99  
Connectionist model, 349  
Consistency-driven problem reduction, 275  
Constraint propagation, 51, 295  
Constructive induction (generation of new  
  terms), 71, 269  
Contextual meta-knowledge, 127  
Data interpretation, 51, 241  
Deep reasoning vs shallow reasoning, 353  
Degree of fit, 299  
Derivational analogy, 213  
Design derivations, 213  
Design plans, 343  
Development of learning systems, 29, 193  
Difference characterizations, learning, 295

- Discovery, learning by, 115, 151, 183, 321  
 Discrimination, repeated, 333  
 Disjunctive spanning, 11  
 Domain, task  
   algebra, 93, 317, 359  
   algebra story problems, 85, 137  
   archeological axes, 145  
   arithmetic, 359  
   blocks world, 75, 199  
   cardiac arrhythmias, 227  
   checkers, 71, 81  
   chemistry, 255  
   chess, 39, 227, 337  
   circuit design, 63, 179, 343  
   classical physics, 307  
   cups world, 123  
   data bases, 375  
   diagnosis, 119  
   dispute mediation, 155  
   dynamical environments, 183  
   elastic object motion, 141  
   Eleusis, 35  
   football, 43  
   furniture, 233  
   games, 161  
   hardware redesign, 213  
   integral calculus, 127  
   language acquisition 385  
   law, 1, 123, 233  
   learning BASIC, 25  
   letter sequence extrapolation, 141  
   lung tumor pathology, 109  
   mathematics, 237, 321, 359  
   metamorphosis of insects, 35  
   naïve physics, 35  
   narrative understanding, 207  
   natural language, 207  
   oil exploration, 379  
   operating a cassette recorder, 35  
   organism simulation, 103  
   patent abstracts, 179  
   physical environment, simple  
     and reactive, 29, 59, 167, 219  
   physical systems, 55  
   physics, 151  
   plant pathology, 145  
   puzzle solving, 115  
   questionnaire design, 145  
   recipe design, 89  
   robotics, 141, 291, 303  
   rocket control, 145  
   software redesign, 213  
   speech recognition, 11  
   tictactoe, 173  
   vision, 145, 299  
   VLSI chip design, 99, 203, 363  
   VLSI process scheduler design, 99
- Domains, properties of task, 269  
   intractable domains, 337  
   noisy domains, 75  
   novel planning domains, 43  
   weakly explainable domains, 179  
 Dynamic memory, 109  
  
 Empirical learning, combined with  
   analytical learning, 55, 321, 375  
 Envisionment, 71  
 Error propagation information, 379  
 Evaluation functions  
   in single-agent problems, 161  
   in two-person problems, 161  
   learning, 39, 269, 295, 343  
 Exception-based knowledge refinement, 193  
 Exceptions to constraints, 375  
 Experiential learning, 219  
 Experiment design, 51, 255  
 Expert reasoning, 241  
 Explainable compiled foresight, 343  
 Explanation-based learning, 47, 63, 71, 123,  
   179, 199, 203, 207, 237, 255, 303, 307,  
   343, 359  
 Explanatory schema acquisition, 47, 207,  
   237, 303, 307  
  
 Failure-driven learning, 359  
 Feature learning, 183  
 First principles, reasoning from, 379  
 Focus rules, 353  
 Frame selection problem, 303  
 Full memory rule refinement, 193  
 Functional dependencies, 55  
 Functional descriptions, 151  
 Functional properties, derivation of, 59  
 Functional specifications, 363  
 Functional vocabularies, 71  
  
 Game-playing, 161  
 Generalization, 71  
   analytic, 375  
   and analogy, 281  
   empirical, 375  
   explicit theory of, 145  
   of action sets, 207  
   *see also* Concept learning  
 Goal interactions, 89  
 Goal regression, 337  
 Goal structure, 337  
  
 Heuristic search, 161, 199, 269  
   design as, 343  
 Heuristics  
   acquiring problem-solving, 167  
   as invariants, 161  
   for concept learning, 11

- Hierarchies, generalization, 11, 333  
 extending, 145, 179  
 multiple, 145
- Imitation, 219
- Indexing rules, memory, 109
- Induction, *see* Concept learning
- Institution
- Al-Lab., Vrije Universiteit Brussels, Belgium, 353
  - Aiken Computation Laboratory, Harvard University, 349
  - Bolt Beranek and Newman, 25
  - Brunel University, 99
  - Carnegie-Mellon University, 29, 59, 103, 119, 199, 219, 261
  - Columbia University, 39, 63, 161, 179
  - Computer System Laboratory, Kawasaki, 213 Japan, 363
  - Faculdade de Economia, Porto, Portugal, 15
  - Georgia Institute of Technology, 155
  - GTE Laboratories, 115, 173, 317
  - Imperial College of Science and Technology, London, 183
  - LRI ORSAY, University de Paris-Sud, 145
  - Massachusetts Institute of Technology, 55, 193
  - Northeastern University, 151
  - Oregon State University, 51, 71
  - Rutgers University, 123, 127, 203, 213, 233, 275, 295, 321, 337, 343, 375
  - Schlumberger-Doll Research, 379
  - Stanford University, 81, 93, 369
  - University of California, Irvine, 59, 67, 75, 85, 137, 167
  - University of California, Los Angeles, 385
  - University of Colorado at Boulder, 11
  - University of Illinois at Urbana-Champaign, 35, 47, 141, 207, 227, 237, 255, 269, 299, 303, 307, 333
  - University of New South Wales, 291
  - University of Texas at Austin, 241
  - Xerox PARC, 359
  - Yale University, 1, 43, 89, 109
- Integration of rules, 353
- Integrity constraints, semantic, 375
- Interesting situation generator, 379
- Interpretation
- of data, 51, 241
  - rules, 99, 379
- Intractable theory problem, 337
- Invariance, 151
- Invariant problem reduction, 275
- Justifiable generalizations, 203
- Justification structure, 379
- Justifications of rules, 369
- Justifications, ranking, 369
- Knowledge acquisition, *see* Acquired problem-solving knowledge *and* Acquiring problem-solving knowledge
- Knowledge acquisition bottleneck, 379
- Knowledge refinement, 379
- Knowledge repair (evolutionary vs revolutionary), 193
- Knowledge transfer between systems, 15
- Knowledge-rich learning, 241
- Learning apprentice systems, 199, 203, 237, 369, 379
- Learning by experimentation, 51
- Learning task formulation, 127
- Linear regression (for learning evaluation functions), 39
- Macro-operators, 275
- Macro-operators, learning, 115, 199, 237, 281
- Macromoves, 237, 275
- Meaning from context, 385
- Memory modification, 109
- Mini-max, 161
- Model-driven experimentation, 255
- Multi-task learning, 29
- Multiple agents, 291
- Mutual data support, 269, 275
- Narrows of the graph, 275
- Neural modeling
- classical conditioning, 103
  - habituation, 103
  - operant conditioning, 103
  - sensitization, 103
- Noisy data, 269, 299
- Numeric and symbolic learning, combining, 75
- Observation, learning by, 35, 207, 237, 261, 307
- Operating conditions, learning, 363
- Operationalization, concept, 127
- see also* Compilation
- Optimal solution path, 161
- Pattern recognition, 11
- Performance, explicit model of problem-solving, 127
- Personal Construct Theory, 183
- Phrasal lexicon, 385
- Phrase-meaning migration, 385

- Plan  
 interactions, 89  
 recognition, 337, 359  
 repair strategies, 89
- Planning, 219, 291  
 domain-independent strategies for, 43
- Precondition analysis, 317
- Preconditions, missing, 213
- Prediction, 219  
 mechanisms for, 183  
 of symbolic processes, 141
- Premise-based knowledge refinement, 193
- Probabilistic approach for errorful data, 299
- Problem reduction, critical, 275
- Problem space creation, 281
- Procedure acquisition, 35, 167, 359
- Program name  
 ABLE, 307  
 AQ11, 141  
 CHEF, 89  
 CLUSTER/2, 333  
 CLUSTER/S, 333  
 Dipmeter Advisor, 379  
 DISCON, 333  
 DONTE, 343  
 EG, 51  
 GENESIS, 207  
 INFER, 93  
 JEN, 173  
 JUDGE, 1  
 LAS, 379  
 LEAP, 203, 363  
 LEX, 127  
 LEX2, 317  
 LP, 317  
 MA, 237  
 Magrathea, 291  
 MALGEN, 93  
 MEDIATOR, 155  
 METALEX, 127  
 MORE, 119  
 NEXUS, 11, 15  
 NLAG, 81  
 Odysseus, 369  
 Physics 101, 307  
 PLS, 269  
 POPART, 213  
 PRE, 51  
 PRESS, 317  
 PRODIGY, 199  
 REO, 173  
 RESEARCHER, 179  
 RINA, 385  
 RUMMAGE, 333  
 Sierra, 359  
 SLIM, 173  
 SPARC/E, 141
- SPARC/G, 141  
 TAXMAN, 233  
 UNIMEM, 179  
 VEXED, 203, 363  
 WYL, 71
- Progressive refinement, 353
- Pseudo-problem, 275
- Purpose of problem-solving steps, 213, 343
- Purpose-directed analogical reasoning, 123
- Qualitative design space, 343
- Rationalizing discrepancies, 369
- Reactive environments, 167
- Recall strategies (for memories), 109
- Recognition  
 of events, 219  
 of objects, 219
- Reformulation, problem, 275
- Repair theory, 359
- Replay, design derivation, 213
- Representation language  
 boolean function, 75  
 censored production rules, 193  
 conceptual dependency, 207  
 deep plans, 51  
 frames, 67, 137, 237  
 predicate calculus, 145  
 PROLOG, 227  
 schemas, 237, 317  
 semantic memory, 155  
 STRIPS-like operators, 141, 199
- Representation of  
 candidate space, 99  
 causality, 123, 207  
 design derivations, 213  
 episodes, 141  
 event, 219  
 functionality, 71, 123  
 implementation rules, 203  
 physical structure, 71, 123  
 plans, 155, 343  
 prototype-based models of concepts, 233  
 purpose, 123
- Representational issues  
 compositionality of representation, 75, 115  
 representation and learning, 99, 275  
 shift of representation, 275, 343  
 use of two vocabularies, 71, 123
- Retraining problem, 299, 303
- Rule combination, 353
- Salience assignment, 75
- Schemata, acquiring problem-solving, 47, 237, 303
- Search reduction, 67, 343
- Simulation, mental, 219

- Skill acquisition, 359  
State variables, learning, 51  
Strategic integration, 353  
Strategy learning, 173  
Structural vocabularies, 71  
Structured objects, 291, 333  
Student modeling, 359
- Task domain, *see* Domain, task  
Taxonomies of learning tasks and algorithms, 99  
Teaching systems, 15  
Temporal adjacency, 141
- Temporal process, 35  
Theory formation, 51  
predictive, 183  
Theory of learning, 99  
Theory revision, 255  
Tractability, 269, 337, 349  
Utilization of concepts in performance, 59
- World  
information-rich, 261  
simulation of, 29  
World Modelers Project, 29, 59, 103, 167, 219