

**EXPERIENCE
FOR MASTERY
KNOWLEDGE
AND
EXPERIENCE IN
EXPERIENCED
LEARNING**

Edited by

Douglas H. Fisher, Jr.

Michael J. Titzelius, and

Pat Lengyel

C ONCEPT FORMATION: KNOWLEDGE AND EXPERIENCE IN UNSUPERVISED LEARNING

The Morgan Kaufmann Series in Machine Learning

Edited by Pat Langley

Machine learning studies the mechanisms through which intelligent systems improve their performance over time. Research on this topic explores learning in many different domains, employs a variety of methods, and aims for quite different goals, but the field is held together by its concern with computational mechanisms for learning. The Morgan Kaufmann series in machine learning includes monographs and edited volumes that report progress in this area from a variety of perspectives. The series is produced in cooperation with the Institute for the Study of Learning and Expertise, a nonprofit corporation devoted to research on machine learning.

Readings in Machine Learning

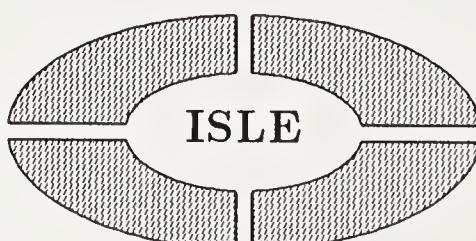
Edited by Jude W. Shavlik (University of Wisconsin-Madison) and Thomas G. Dietterich (Oregon State University).

Computational Models of Scientific Discovery and Theory Formation

Edited by Jeff Shrager (Xerox Palo Alto Research Center) and Pat Langley (NASA Ames Research Center).

Concept Formation: Knowledge and Experience in Unsupervised Learning

Edited by Douglas H. Fisher, Jr. (Vanderbilt University), Michael J. Pazzani (University of California-Irvine), and Pat Langley (NASA Ames Research Center).



CONCEPT FORMATION: KNOWLEDGE AND EXPERIENCE IN UNSUPERVISED LEARNING

Edited by
Douglas H. Fisher, Jr.,
Michael J. Pazzani, and
Pat Langley

Morgan Kaufmann Publishers, Inc.
San Mateo, California

Q 325.5 .C6 1991
Sponsoring Editor *Michael B. Morgan*
Production Editor *Sharon E. Montooth*
Cover Designer *Jo Jackson*
Cover Mechanical Artist *Vicki Philip*
Copyeditor *Bob Klinginsmith*

Cover art is from *The Celtic Art Source Book* by Courtney Davis, copyright 1988; and is reproduced with permission from Cassell Publishers, London, England.

Library of Congress data is available for this book.

Morgan Kaufmann Publishers, Inc.

Editorial Office:

2929 Campus Drive, Suite 260
San Mateo, CA 94403
(415) 578-9911

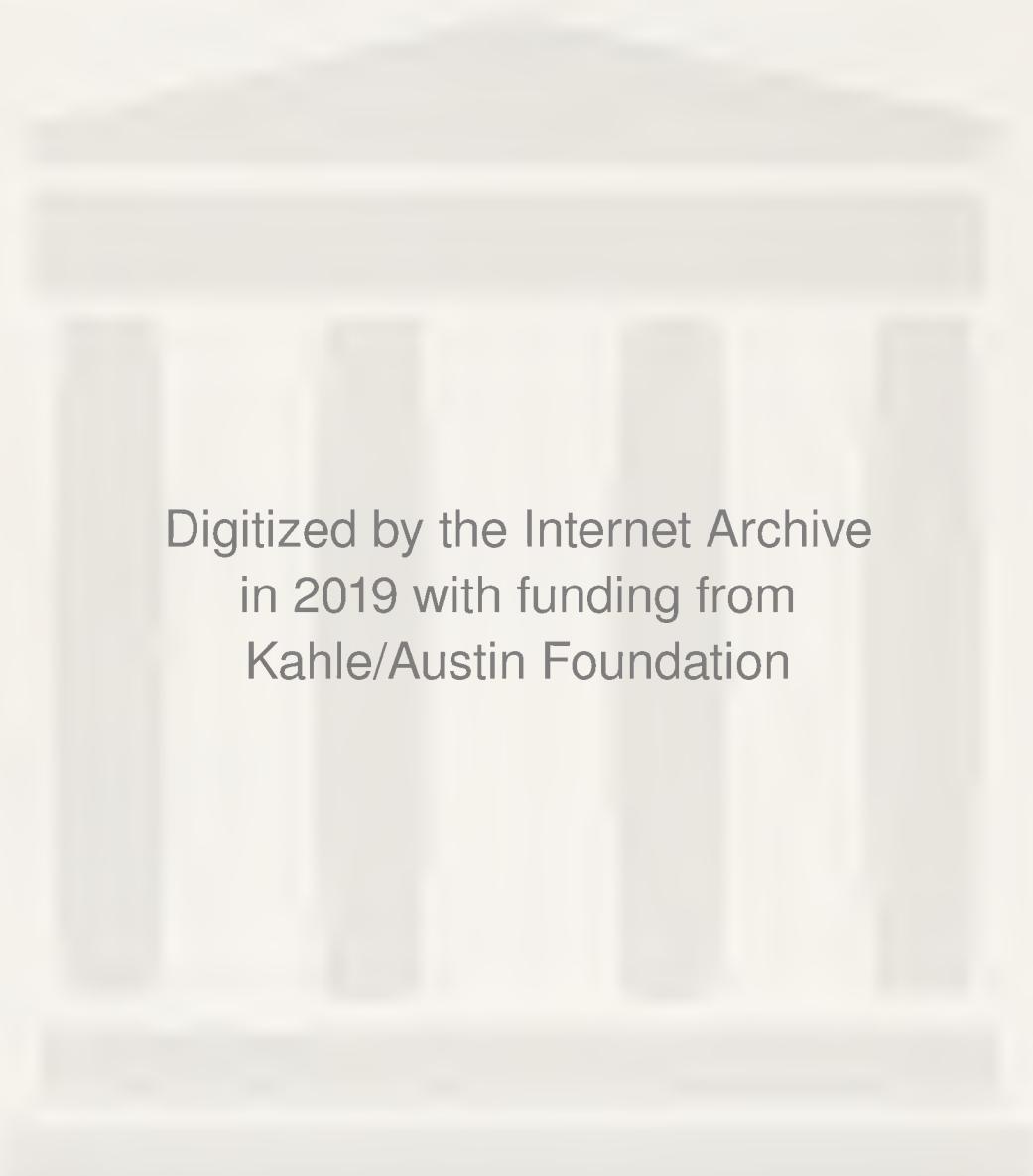
©1991 by Morgan Kaufmann Publishers, Inc.

All rights reserved.

Printed in the United States.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means-electronic, mechanical, photocopying, recording, or otherwise-without the prior written permission of the publisher.

We dedicate this book to the authors and other participants of the Symposium on Computational Approaches to Concept Formation, who have helped us realize success in the completion of this volume.



Digitized by the Internet Archive
in 2019 with funding from
Kahle/Austin Foundation

<https://archive.org/details/conceptformation0000unse>

Contents

I Inductive Approaches to Concept Formation

1	Computational Models of Concept Learning DOUG FISHER AND MICHAEL PAZZANI	3
2	An Incremental Bayesian Algorithm for Categorization JOHN R. ANDERSON AND MICHAEL MATESSA	45
3	Representational Specificity and Concept Learning JOEL D. MARTIN AND DORRIT BILLMAN	71
4	Discrimination Net Models of Concept Formation HOWARD B. RICHMAN	103
5	Concept Formation in Structured Domains KEVIN THOMPSON AND PAT LANGLEY	127

II Knowledge and Experience in Concept Formation

6	Theory-Guided Concept Formation DOUG FISHER AND MICHAEL PAZZANI	165
7	Explanation-Based Learning as Concept Formation RAYMOND J. MOONEY	179
8	Some Influences of Instance Comparisons on Concept Formation BRIAN H. ROSS AND THOMAS L. SPALDING	207

- 9 Harpoons and Long Sticks: The Interaction of
Theory and Similarity in Rule Induction 237
EDWARD J. WISNIEWSKI AND DOUGLAS L. MEDIN

- 10 Concept Formation over Problem-Solving
Experience 279
JUNGSOON YOO AND DOUG FISHER

III The Utility of Concept Formation in Intelligent Behavior

- 11 Concept Formation in Context 307
DOUG FISHER AND MICHAEL PAZZANI

- 12 The Formation and Use of Abstract
Concepts in Design 323
YORAM REICH AND STEVEN J. FENVES

- 13 Learning to Recognize Movements 355
WAYNE IBA AND JOHN H. GENNARI

- 14 Representation Generation in an
Exploratory Learning System 387
PAUL D. SCOTT AND SHAUL MARKOVITCH

- 15 A Computational Account of Children's
Learning About Number Conservation 423
TONY SIMON, ALLEN NEWELL, AND DAVID KLAHR

Preface

DOUG FISHER

MICHAEL PAZZANI

PAT LANGLEY

The problems addressed by artificial intelligence research vary widely, from designing autonomous vehicles for space exploration to developing ‘intelligent’ tools for engineering and design to modeling aspects of human behavior. Artificial intelligence approaches to these problems are characterized by the ability to explore alternative actions and, in many cases, to acquire new information during this exploration. This latter capability is called *machine learning*, and it enables an ‘intelligent’ system to improve its own performance, to concisely communicate its experience to a human analyst, and to model the adaptive nature of human intelligence.

Interest in machine learning has become widespread, but with growing interest comes greater impetus to question traditional assumptions that limit the power of the technology. For example, much of the early research on machine and natural concept learning was *supervised* in that the learner was told the category membership of environmental observations; the sole learning task was to summarize the commonality among members of the same categories and differences among competing ones. In contrast, *unsupervised* methods are not provided with this guidance; rather, they must discover ‘useful’ categories in the data using internalized heuristics. A second feature of many traditional concept learning models is that they assume that all environmental observations are available from the outset of learning. This *nonincremental* or ‘batch’ assumption can be contrasted with an *incremental* strategy that learns over a stream of observations.

We believe that unsupervised and incremental assumptions reflect many real-world situations in which humans find themselves, and that they are increasingly important in the construction of machine learn-

ing systems. Together, these two assumptions comprise the task that we call *concept formation*. Our interest in this area emanated from a common base of research by Mike Lebowitz, Janet Kolodner, and Roger Schank on the dynamic nature of memory. However, we took separate slants on a number of critical dimensions. Fisher's and Langley's work – independently, together, and with the help of others, notably Dennis Kibler – was concerned with inductive learning and took further inspiration from computational work by Robert Stepp and Ryszard Michalski on conceptual clustering, the framework for concept representation put forward by Edward Smith and Doug Medin, and psychological research on basic-level effects, most notably the work of Jim Corter and Mark Gluck. In contrast, Pazzani was more directly influenced by the psychological work of Tom Shultz and others on the development of causal reasoning, the knowledge-intensive approaches to machine learning developed by Gerald DeJong and Tom Mitchell, and the psychological findings of Greg Murphy and Doug Medin on the role of background knowledge in learning. These influences led to computational investigations of how knowledge acquired in one area can facilitate learning in related areas. Despite the differences in our work, it seemed clear that our research programs were not part of competing paradigms. Rather, we were addressing similar issues using similar mechanisms, though with somewhat different motivations and terminological conventions.

To promote interdisciplinary interaction between machine learning and cognitive psychology on unsupervised incremental methods, Langley suggested a symposium that would bring together researchers in both fields. The Symposium on Computational Approaches to Concept Formation was held at Stanford University in January, 1990. It was the second in a series of meetings in the area of machine learning administered through the Institute for the Study of Learning and Expertise (ISLE). The symposium was organized around 16 talks representing research in both machine learning and cognitive psychology. Our goal was to downplay surface distinctions between efforts in these communities, and to stress commonality in the research agendas. For example, both fields showed a recent interest in analytic and knowledge-intensive methods, as evident in explanation-based and case-based research, which can be contrasted with more traditional inductive, data-intensive learning. This dichotomy between knowledge-intensive and data-intensive methods has had a profound effect on how researchers in machine learning

and cognitive psychology view issues of similarity, memory, and problem solving, both in general and with respect to concept formation.

Our desire for cross-disciplinary interaction has also guided our organization of this book, which largely grew out of the presentations at the symposium. The chapters included in this volume are divided into three sections, the first being concerned with inductive, data-intensive methods for concept formation. In particular, Chapters 1 through 5 focus on measures of similarity, strategies for robust incremental learning, the representation and organization of discovered categories, and the psychological consistency of various approaches. In Chapter 1, Fisher and Pazzani give an overview of inductive concept learning in machine learning and psychology, with special emphasis on issues that distinguish concept formation from more prevalent supervised methods and from numeric and conceptual clustering. Chapter 2, by Anderson and Matessa, describes the cognitive consistency of two concept formation systems that are motivated by a *rational analysis* of human behavior relative to a variety of psychological phenomena. Martin and Billman's discussion in Chapter 3 focuses on the merits of various schemes for representing and acquiring knowledge during concept formation. In Chapter 4, Richman reviews some of the earliest work in concept formation and offers some novel accounts of certain psychological data using these methods. In Chapter 5, Thompson and Langley describe a system that forms concepts with both complex componential structure and relations among their components.

In Chapters 6 through 10 we turn our attention from data-intensive approaches to those that exploit domain knowledge to bias the concept formation process. Fisher and Pazzani open that section with an overview of some psychological and computational motivations for bringing domain knowledge to bear in unsupervised models. However, the knowledge-intensive approach has the greatest number of adherents in an area of supervised learning, notably that of *explanation-based learning*. In Chapter 7, Mooney argues that, although explanation-based learning has traditionally been viewed as a supervised task, a number of systems in this paradigm are best cast as unsupervised.

Of course, the knowledge-intensive versus data-intensive dichotomy is somewhat misleading; it is more natural to think of these strategies as lying at far ends of a continuum. In many (if not most) situations, both background knowledge and data play a role in concept learning.

Chapter 8 by Ross and Spalding, and Chapter 9 by Wisniewski and Medin, address the relative role of knowledge and data in biasing human concept learning processes. Most of the experimental data that they describe were obtained in supervised settings, but their findings are nonetheless highly relevant to issues of similarity in concept formation. In fact, Yoo and Fisher describe an unsupervised system in Chapter 10 that embodies some of the principles introduced in these earlier chapters; their system exploits data and background knowledge to cluster and reuse past problem-solving experiences.

Finally, Chapters 11 through 15 extend the theoretical contributions of the first two sections, but they also focus on the utility of concept formation methods in particular contexts. Chapter 11 surveys the role of concept formation in scientific discovery, engineering, problem solving, natural language processing, and information retrieval. In Chapter 12, Reich and Fenves report on an application in engineering design, as well as addressing fundamental issues, such as dealing with numeric data. Iba and Gennari describe a system that learns to recognize physical movements in Chapter 13, using ideas from work on concept formation not only to acquire movement categories but also to determine their componential structure. In Chapter 14, Scott and Markovitch describe a learning system that is not passive, but that actively explores its environment through experimentation. In Chapter 15, Simon, Newell, and Klahr describe their Q-SoAR model of the development of number conservation in young children, which relies on a concept formation strategy to organize developmental experiences.

Collectively, these chapters represent the culmination of considerable effort. We are indebted to the many individuals and organizations who have contributed to this book. Most importantly, we thank participants of the Symposium on Computational Approaches to Concept Formation for their stimulating presentations and discussions, and to the authors of the chapters herein for their patience and dedication to quality. Funding for the symposium was provided by Grant No. 8921582 from the National Science Foundation, Grant No. N00014-90-J-1394 from the Office of Naval Research, a gift from the American Association for Artificial Intelligence, and Vanderbilt University. We thank the officials of these agencies and institutions for their support, which contributed to the smooth operation of the symposium. Mildred Tyler and Pam Wilson of Vanderbilt University, Caroline Ehrlich of the University of California, Irvine, and particularly Helen Stewart and Martha Del Alto of NASA

Ames Research Center provided support in the administration of grants and the preparation of book chapters. Wayne Iba and Kevin Thompson helped with details of the symposium, and Mark Gluck and Gordon Bower secured facilities on the Stanford campus. Kate McKusick deserves special mention in regard to local arrangements — her efforts were indispensable. Michael Morgan and Sharon Montooth of Morgan Kaufmann Publishers were encouraging and helpful — everything that one expects and hopes for in a publisher. Ross Quinlan, Edward Smith, and a third anonymous reviewer provided excellent suggestions on the organization of the book, and we have tried to follow their recommendations.

We believe that the chapters collected here provide a representative cross-section of the work currently under way in the area of concept formation. In addition to cognitive scientists and AI researchers, the book should interest data analysts involved in clustering, philosophers concerned with the nature and origin of concepts, and researchers dealing with issues of similarity, memory organization, and problem solving. We hope that the book promotes interaction, and that researchers in related fields will help extend our understanding of the incremental and unsupervised acquisition of conceptual knowledge.

Contributors

John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
(ANDERSON@PSY.CMU.EDU)

Dorrit Billman
School of Psychology
Georgia Institute of Technology
Atlanta, GA 30332
(BILLMAN@PRAVDA.GATECH.EDU)

Steven J. Fenves
Engineering Design Research Center
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
(FENVES@CE.CMU.EDU)

Douglas H. Fisher, Jr.
Department of Computer Science
Vanderbilt University
Nashville, TN 37235
(DFISHER@VUSE.VANDERBILT.EDU)

John H. Gennari
Department of Computer Science
Keio University
3-14-1 Hiyoshi
Kohoku-ku, Yokohama, Japan 223
(GENNARI@CS.KEIO.AC.JP)

Wayne Iba
AI Research Branch (MS 244-17)
NASA Ames Research Center
Moffett Field, CA 94035
(IBA@PTOLEMY.ARC.NASA.GOV)

David Klahr
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
(KLAHR@PSY.CMU.EDU)

Pat Langley
AI Research Branch (MS 244-17)
NASA Ames Research Center
Moffett Field, CA 94035
(LANGLEY@PTOLEMY.ARC.NASA.GOV)

Shaul Markovitch
Computer Science Department
Technion — Israel Institute
of Technology
Haifa 32000 Israel
(SHAUL@CS.TECHNION.AC.IL)

Joel D. Martin
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
(JOEL@PRAVDA.GATECH.EDU)

Michael Matessa
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
(MM4B+@ANDREW.CMU.EDU)

Douglas L. Medin
Department of Psychology
The University of Michigan
Ann Arbor, MI 48104-2994
(DOUG_MEDIN@UM.CC.UMICH.EDU)

- Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712-1188
(MOONEY@CS.UTEXAS.EDU)
- Allen Newell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
(AN@CS.CMU.EDU)
- Michael J. Pazzani
Department of Information &
Computer Science
University of California
Irvine, CA 92717
(PAZZANI@ICS.uci.edu)
- Yoram Reich
Engineering Design Research Center
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
(YORAM.REICH@CS.CMU.EDU)
- Howard B. Richman
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
- Brian H. Ross
Department of Psychology
University of Illinois
Champaign, IL 61820
(BRoss@PSYCH.UIUC.EDU)
- Paul D. Scott
Department of Computer Science
University of Essex
Wivenhoe Park, Colchester CO4 3SQ
England
(SCOTP@ESSEX.AC.UK)
- Tony Simon
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
(TONY.SIMON@CS.CMU.EDU)
- Thomas L. Spalding
Department of Psychology
University of Illinois
Champaign, IL 61820
(TSPALDING@PSYCH.UIUC.EDU)
- Kevin Thompson
AI Research Branch (MS 244-17)
NASA Ames Research Center
Moffett Field, CA 94035
(KTHOMPSO@PTOLEMY.ARC.NASA.GOV)
- Edward J. Wisniewski
Department of Psychology
The University of Michigan
Ann Arbor, MI 48104-2994
(EDWARD_WISNIEWSKI@UM.CC.UMICH.EDU)
- Jungsoon Yoo
Department of Computer Science
Vanderbilt University
Nashville, TN 37235
(YOOJP@VUSE.VANDERBILT.EDU)

Part I

Inductive Approaches to Concept Formation

CHAPTER 1

Computational Models of Concept Learning

DOUG FISHER

MICHAEL PAZZANI

1. Introduction

The success of an intelligent agent, whether human or machine, depends critically on an ability to adapt to the environment through learning. Figure 1 (Dietterich, Clarkson, Dromey, & London, 1981) illustrates that learning organizes experiences in a manner that ideally improves performance on some task(s) (Simon, 1983). However, learning methods vary widely in terms of the environmental inputs that they assume, the manner in which they process and store this information, and the performance tasks along which the costs and benefits of learning are evaluated. This chapter delineates the task of *concept formation* along environmental, knowledge base, and performance dimensions. A concept formation system accepts a *stream* of observations (i.e., events, objects, instances), and discovers a classification scheme over the data stream. Thus, concept formation is an ‘unsupervised’ task in the traditional sense — observations are not classified *a priori* by an external source (e.g., a teacher). Second, learning is *incremental* — a classification scheme evolves and changes as new observations are processed — observations are not processed *en masse*.

Incremental and unsupervised constraints characterize many real-world situations in which humans find themselves, and adaptability under these same conditions is increasingly important in artificial intelligence and machine learning. For example, from news stories we can extract classes of politicians that inform our voting, or a person with diabetes may develop categories over behavioral features (e.g., meals,

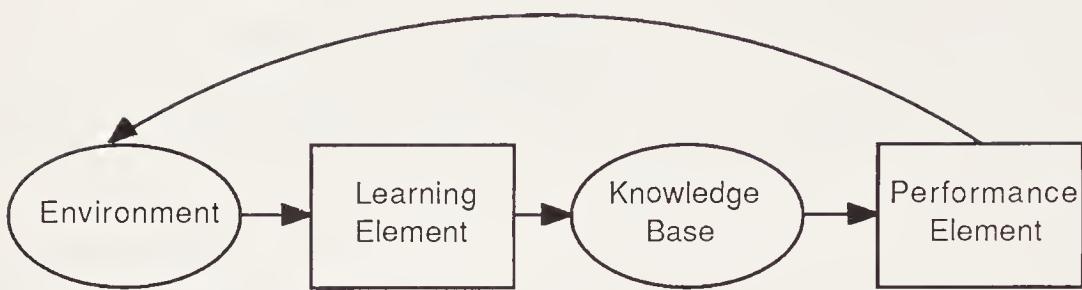


Figure 1. A framework for learning and performance.

exercise, insulin dosage, stress) that facilitate flexible decision making (e.g., how much and what kind of breakfast will minimize required insulin given a morning swim of 20 laps). This book explores a variety of AI and psychological models that address incremental, unsupervised tasks. This chapter prepares for later discussions by motivating the dimensions implicit in Figure 1, describing how they are managed during concept formation, and placing concept formation in context relative to better known tasks of supervised learning and other unsupervised tasks found in statistics and machine learning.

2. Supervised Learning

Psychology and AI have traditionally emphasized tasks that do not require the learner to discover categories. Rather, a ‘teacher’ delimits the classes to be learned. Nonetheless, these traditional *supervised* learning models introduce some important issues of representation and processing that are requisite background for our later introduction to concept formation.

2.1 Psychological Models of Supervised Learning

Early psychological studies (Bruner, Goodnow, & Austin, 1956) required subjects to identify a rule or concept that an experimenter was using to classify a stream of observations. After each observation, the subject predicted the observation’s category membership and was then told the correct answer. Subjects were assumed to have discovered the appropriate concept after correctly predicting class membership for a sufficient number of consecutive trials. Many studies assumed that the

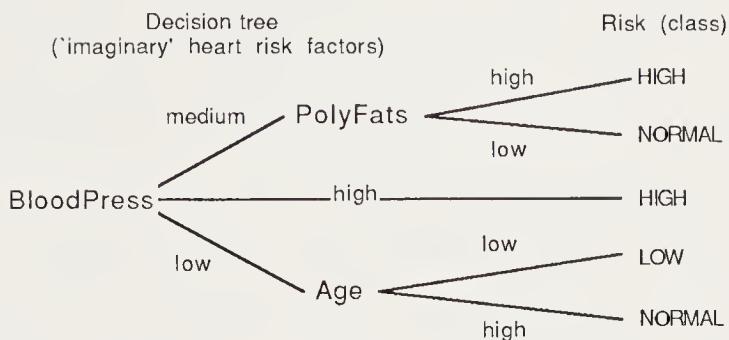


Figure 2. An ‘imaginary’ decision tree for diagnosing heart-attack risk.

experimenter classified observations into a ‘positive’ or ‘negative’ category based on the presence or absence of a conjunction of features (Bruner et al., 1956). The strategy followed by some human learners was modeled by the WHOLIST procedure for acquiring conjunctive concepts: the learner initialized the candidate concept description to the first positive instance that the subject observed, and removed features from this set if they failed to appear in subsequent positive instances.

Conjunctive concepts have been well studied in psychology, but they are limited in important ways. For example, humans are able to learn disjunctive concepts as well, in which no one set of features is necessary and sufficient for concept membership. More generally, an implication of conjunctive representations is that all concept members are ‘equal’, since the requisite conjunction is either satisfied or not. However, human subjects do not treat concept instances equally, but regard certain members as more ‘typical’ than others. In this regard, several studies (e.g., Rosch & Mervis, 1975) indicate that subjects more quickly affirm that a *robin* is a *bird* than they will affirm that a *chicken* is a *bird*. In particular, response time is proportional to a typicality gradation of category members. These and other experimental findings (Mervis & Rosch, 1981; Smith & Medin, 1981) have motivated researchers to propose alternatives to conjunctive representations.

One alternative is to organize a system of many simple concepts into a single classifier. This idea is behind Hunt, Marin, and Stone’s (1966) Concept Learning System, CLS, which was intended to model aspects of human disjunctive concept learning (also see Richman, this volume). In

particular, CLS constructed a *decision tree*, like that shown in Figure 2. CLS partitioned the known observations into subsets, each corresponding to the value of a ‘best’ divisive attribute (e.g., `color = red blue or green`).¹ The system used simple ‘rules of thumb’ to determine the attribute with values that best discriminated the known positive and negative observations. Attribute values label arcs from the root of the decision tree and segregate objects into disjoint subsets; the same procedure is recursively applied to each subset until one reaches nodes that only cover members of one (positive or negative) category. Each path from root to leaf corresponds to a conjunction of features, where each conjunction may reside in different parts of feature space.² CLS was not designed with typicality effects in mind, but we might expect that typical instances would tend to be more quickly classified since shorter paths would be required to distinguish them from contrasting categories.

Another alternative to conjunctive descriptions are *probabilistic concepts* (Smith & Medin, 1981), which associate a probability or weight with properties of a concept definition. For example, an *independent cue* model stores the conditional probability, $P(f|C_k)$, or other weight of each feature f ’s presence with respect to each category C_k . Classification involves summing the weights of features that are present in a new observation until the sum passes a specified threshold (Smith & Medin, 1981). More typical instances will tend to have features with higher weights, and typicality phenomena are explained by the differences in the time required to reach some criterial sum. However, despite their rather natural account of typicality, independent cue models are limited to recognizing linearly separable categories. They have greater expressive power than conjunctive concepts, but they are nonetheless restricted. In this regard though, several researchers (Fisher & Langley, 1990; Utgoff & Brodley, 1990; Anderson & Matessa, this volume) point out that probabilistic concepts can be used as nodes in structures like decision trees, thereby expanding the representational power of each.

-
1. Actually, CLS was limited to binary-valued attributes (e.g., present or absent), though it spawned a line of systems that have overcome this limitation (Quinlan, 1986). We show an example tree formed by one of these extensions.
 2. *Exemplar* or instance-based approaches (Smith & Medin, 1981; Medin & Schaffer, 1982; Aha & McNulty, 1989; Aha, Kibler, & Albert, 1991) store a subset of the training instances which, like the alternate paths of a decision tree, may appear very dissimilar in the feature description space — even instances that are members of the same category.

2.2 Inductive Approaches to Machine Learning

In many respects machine learning techniques are similar to strategies that have been hypothesized to underlie human concept learning. For example, the well-known ID3 system for learning decision trees (Quinlan, 1986) descends directly from CLS. In contrast, many systems mimic WHOLIST's strategy for learning conjunctive concepts by initializing a concept to contain the properties of the first positive instance, and gradually dropping properties as they fail to be observed in subsequent positive instances (Mitchell, 1982; Dietterich & Michalski, 1983; F. Hayes-Roth & McDermott, 1978; Vere, 1980; Pazzani & Sarrett, 1990). In contrast, several systems follow the inverse strategy; they begin with a maximally general concept (e.g., the 'empty' concept of no conditions) that is gradually specialized by adding conditions in a manner that excludes negative examples (Langley, 1987). Thus, machine learning systems differ in the *direction* in which they search for meaningful concepts (Simon & Lea, 1974; Mitchell, 1982). There are also some systems that combine specific-to-general and general-to-specific abilities by carrying out a *bidirectional* search (Anderson & Kline, 1979; Mitchell, 1982), though they vary widely in their details.

Machine learning systems also differ in terms of the *search control* strategies that they employ. For example, consider a general-to-specific learning system that conducts a *breadth-first* search for a correct concept (e.g., Mitchell, 1982): there may be many ways to 'minimally' specialize a concept in order to exclude a negative case, and a breadth-first learner will remember each possibility. More generally, at any particular point in time, there will be a frontier of concepts that exclude all of the previous negative observations. Control strategies range from exhaustive techniques like breadth-first search, which retain *all* concepts that are consistent with the known observations, to heuristic methods like *beam search* (Michalski, 1983), which retain only the 'best' hypotheses that are consistent with the observations.

Unlike experimental human subjects, many AI learning systems do not actively predict class membership for each new observation. Rather, they process all available observations *en masse*. However, there are many situations in which one wants to assimilate and respond to observations incrementally (Schlimmer & Fisher, 1986). There are at least two motivations for such a bias. Some systems are motivated by a need to rapidly and frequently exploit conceptual information during learning

(e.g., a child learning to play a competitive game). In contrast, other systems may have little concern with very accurate concepts (e.g., learning to distinguish the views of candidates for political office). In either case it may be desirable to reduce assimilation cost, even if this means a decrease in absolute quality of acquired knowledge (Simon, 1969).

For example, Hunt et al. adapted CLS to perform incrementally by rebuilding a decision tree from known observations whenever a new one was misclassified by the existing tree. This process was made more efficient by having the system remember only a bounded number of past observations. Several efforts (Larson & Michalski, 1978; Reinke & Michalski, 1988) have refined this strategy of reusing remembered instances in various ways. Other approaches do not save examples at all, but maintain a summary description of past instances that can be used to rebuild a concept (Schlimmer & Granger, 1986). In particular, consider Schlimmer and Fisher's (1986) ID4, which descends from ID3 and CLS. This system maintains statistics at each node of a decision tree which reflect the distribution of attribute values over observations that have been classified under the node. Information-theoretic scores for selecting the 'best' divisive attribute are computed from these summary descriptions; if at any point during learning the score of a node's divisive attribute drops below the score of an alternate attribute, then the subtree that is rooted at this node is deleted and the values of the new most-informative attribute are used to initialize a new subtree. As more observations are observed, this subtree is gradually regrown under the constraints of the new organization.

ID4 does not assume that perfect consistency with the data is necessarily desirable (or even possible). Thus, it does not rebuild a tree following each misclassification. Rather, recomputation is triggered only after some degree of inconsistency with the data indicates that an attribute is a poor basis for dividing a node. Probabilistic summaries of attribute-value distributions support this conservative revision strategy. Schlimmer and Fisher's (1986) experiments with ID4 indicated that it took many more observations to obtain a classifier of the same accuracy as a version of ID3 that was rerun after each misclassification. However, it nonetheless quickly approximated the accuracy of the alternative classifier. Utgoff (1989) has developed ID5, which reorganizes subtrees rather than deleting them, thereby saving considerably on the number of observations required for learning, retaining most of ID4's efficiency, and obtaining trees identical to those of ID3 on the same data.

2.3 Issues of Supervised Learning

We have surveyed a variety of supervised learning methods. We now highlight relevant aspects of these systems that will be important in our discussion of unsupervised learning and concept formation.

2.3.1 THE ENVIRONMENT AND PERFORMANCE

Our review of individual systems ignores several factors that confound learning. One factor is the possibility of noise or variation in data that blurs the boundaries between categories. As we will see, noise can complicate learning and degrade prediction accuracy. A second factor is the representation of instances. Most systems assume an attribute-value representation, but others deal with more complicated *structured* representations (Dietterich & Michalski, 1983; Medin, Wattenmaker, & Michalski, 1986; Langley, 1987; Quinlan, 1990). For example, suppose the learner sees two card hands (positive instances), where one contains three Jacks and two Kings and the second contains two Jacks and three Kings. One summary of these observations is that they each contain at least two Jacks and at least two Kings, but an alternative is that they contain two cards of one face and three of another (i.e., a full house). These alternatives stem from the different ways that components (i.e., cards) in the two hands can be matched. Alternative matchings can complicate the task of predicting category membership. For example, a concept representation for a full house may also cover a card hand for a pair (i.e., two cards of one face and two cards of another), unless special precautions are taken to disambiguate between these alternatives at playing time (i.e., the full house match takes priority).

We have also alluded to issues that are relevant when observations are presented in a *stream*, and when responses to these observations are required in a timely manner. The design of an incremental learning system depends on constraints along dimensions of assimilation cost and concept quality, such as the amount of time one has to respond to environmental stimuli and the required accuracy of these responses. Very often the need for accurate prediction and timely response act in opposition, and a tradeoff analogous to *bounded rationality* comes into play (Simon, 1969); an actor performs as well as possible (or necessary) under resource (e.g., time) constraints. Specifically, there are four di-

mensions that characterize incremental learning systems (Schlimmer & Fisher, 1986):

1. the cost of assimilating a single observation;
2. the asymptotic level of prediction accuracy that is achieved;
3. the learning rate or the number of observations that are needed to reach specified accuracy levels; and
4. the total cost required to achieve a specified level of accuracy (i.e., the number of observations times the cost per observation).

We now turn to the techniques that different models use to manage these tradeoffs.

2.3.2 THE LEARNING COMPONENT

To operate efficiently in incremental settings, several systems conduct a form of constrained search called *hill climbing*, which maintains a single ‘active’ concept description that may be modified after each training instance. Systems like ID4 keep no explicit memory of alternative hypotheses, though they may create ‘alternatives’ by application of their learning mechanisms. For example, Figure 3 illustrates the abstract behavior of ID4 and ID5. If an internal node’s divisive attribute becomes less informative than an alternate, then ID4 will delete this subtree and grow a new one from subsequent instances. Thus, ID4 has *bidirectional* capabilities: specialization operators that expand the tree and generalization operators (i.e., dropping a subtree) that simulate backtracking. In contrast, ID5 can move ‘laterally’ along an imaginary search frontier by reorganizing the subtree that is deemed nonoptimal. Memory limits make many of these systems sensitive to the order of training instances, but bidirectional mobility mitigates these effects.

Although hill climbing is effective in many settings (Utgoff, 1989; Langley, Gennari, & Iba, 1987), most of these systems make an assumption that objects have an attribute-value representation. In contrast, the nondeterminism of matching structured descriptions suggests the utility of more extensive search (Dietterich & Michalski, 1983). To compromise between efficient response and accurate relational concepts, some systems (e.g., Winston, 1975) retain true backtracking ability, along with mechanisms for bidirectional mobility like those of ID4 and ID5.

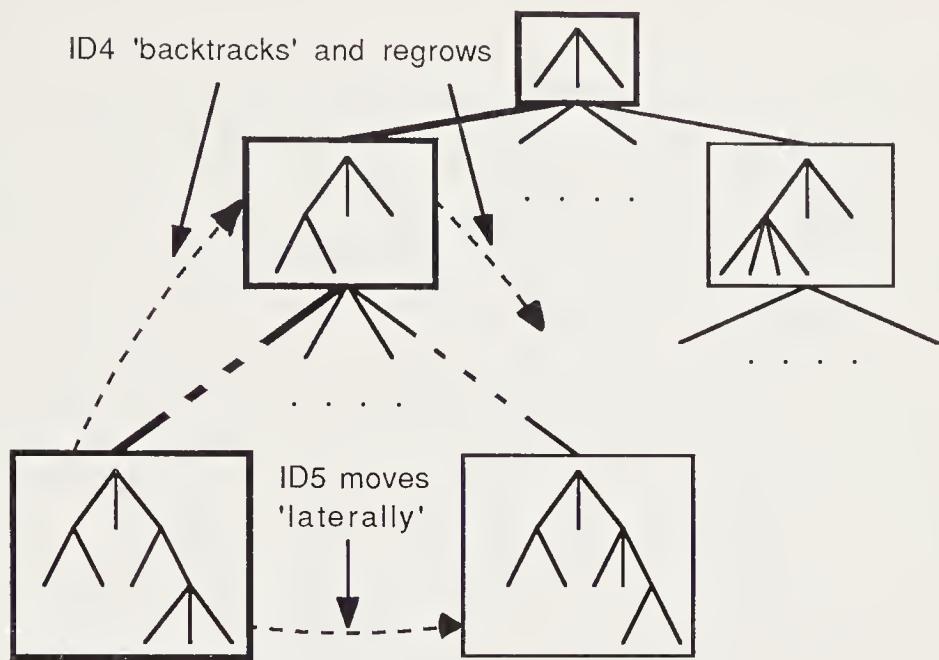


Figure 3. Bidirectional and lateral mobility in ID4 and ID5.

In addition to the complications introduced by relational descriptions, problems of noise can also confound the search process, since noise often negates the possibility of concepts that are perfectly consistent with the data. Some systems like ID3 mitigate noise by pruning disjunctive extensions to a decision tree if the average distribution of classes (e.g., positive and negative) at the children of a node is not significantly different (e.g., by a χ^2 test) from the distribution at the node itself (Breiman, Freidman, Olshen, & Stone, 1984; Michalski, 1987; Clark & Niblett, 1989; Mingers, 1990). Another option maintains a weighted set of competing descriptions (Anderson & Kline, 1979; Langley, 1987; Schlimmer & Granger, 1986; Salzberg, in press). These systems can be viewed as doing a conservative beam search for the best hypotheses: concepts are retained as long as they cover a ‘significant’ portion of the observations seen so far, and are removed only if they drop below a certain threshold of coverage. Several of these systems can also be viewed as relaxing the strict hill-climbing approach because of the greater search required by relational descriptions (Langley, 1987; Anderson & Kline, 1979), though this may not have been the original motivation.

2.3.3 THE KNOWLEDGE BASE

The structure of the knowledge base can facilitate the search performed by learning systems, especially in incremental settings. In particular, we have distinguished conjunctive, instance-based, probabilistic, and tree-structured representations. There is some evidence that in noise-free, nonincremental settings, humans prefer conjunctive concepts (Medin, Hampson, & Michalski, 1986; Bruner, Goodnow, & Austin, 1956). However, a strict conjunctive representation will only be robust in the face of incremental processing if it is coupled with search control strategies (e.g., backtracking, breadth-first processing) that recover from the frequent inconsistencies that arise over a stream of observations. To improve the efficiency of assimilation, several systems use a probabilistic summary of instances to trigger repairs to (logical) concepts in the face of inconsistent data: statistics over the observations need not be recomputed as they are with CLS (and many instance-based approaches), but they are incrementally maintained over the duration of learning, as in ID4. We speculate that the utility of probabilistic representations to efficiently and conservatively deal with inconsistencies that unfold incrementally is related to their account of typicality findings: incremental learning may insist on representations that facilitate graded inconsistency during learning — a characteristic that remains with the post-learning concept.

3. Unsupervised Learning

The ubiquity of supervised learning tasks motivates considerable study, but there are many scenarios in which a learner cannot rely on externally defined categories. Instead, the learner must invoke internalized heuristics to organize its observations into categories. This section describes work of this type from several disciplines.

3.1 Clustering Methods of Numerical Taxonomy

Perhaps the oldest area of research on unsupervised processing is *numerical taxonomy* (Everitt, 1980). Algorithms from this paradigm group or cluster similar observations into the same category and segregate dissimilar observations into distinct categories. For example, the popular *agglomerative* strategy builds (typically binary) trees called *dendograms*. Each node of the tree represents a class of observations. The tree is con-

structed in a bottom-up manner by computing the ‘similarity’ between all pairs of observations (for N observations there will be $(N - 1)N/2$ pairs), and merging the two most similar into a single category. This process is repeated using the new set of $N - 1$ categories (i.e., $N - 2$ singletons and the newly created category of two observations). Each merging operation adds an internal node to the developing tree. The merging process is repeated $N - 1$ times, at which point all observations have been placed in a single category that represents the root of the dendrogram.

Agglomerative methods depend on some way of computing similarity between two categories so that the ‘most similar’ can be merged. A *nearest-neighbor* strategy reduces the problem of determining similarity between categories to one of determining similarity between observations: the most similar pair of observations from contrasting categories represents the similarity of the categories. There are a plethora of measures for determining similarity between observations, including the inverse of *Euclidean distance*, $[\sum_{k=1}^K (x_{ik} - x_{jk})^2]^{1/2}$, where x_{ik} and x_{jk} are the values of attribute k for observations i and j , respectively; the closer the values of individual attributes, the smaller the denominator of the inverted distance, and the greater the similarity.³

Most agglomerative approaches use the basic strategy outlined above, but there are many variations, particularly in the measures used to determine similarity between categories (Everitt, 1980; Jain & Dubes, 1988). In addition, there are a wide variety of *divisive* methods that also build a tree, but do so in a top-down manner beginning with a single category containing all observations and repeatedly subdividing this category until singleton categories are obtained. Collectively, agglomerative and divisive methods are known as *hierarchical* approaches. Once determined, a tree formed by a hierarchical method can be examined at arbitrary levels by a data analyst in search of useful clusters. Hierarchical methods are relatively efficient, and trees allow the analyst flexibility in determining the best number of clusters in the data. However, sometimes the assumption that useful clusters will arise from a series of local, pairwise comparisons is invalid (Everitt, 1980).

3. In general, this approach is best suited to numerically valued attributes, but a simple (though somewhat unsatisfying) measure of similarity over nominally valued attributes is the proportion of attributes with the same value in the two observations.

Optimization or partitioning techniques more thoroughly search for an optimal partitioning of the observations. In general, these methods do not form a tree-structured classification, but simply partition the observations at a single level. They rely on an explicit measure of cluster quality to guide search, rather than a local, pairwise measure of similarity from which good clusterings might emerge. For example, a common strategy is to partition the data in a way that minimizes the squared error within clusters, which is equivalent to maximizing within-category similarity. This approach has intuitive appeal, but in reference to psychological models, Medin (1983) points out that the set of singleton categories optimizes intra-category similarity, since each observation is maximally similar to itself. Thus, attention to intra-category similarity alone does not provide a sufficient basis for deciding upon the appropriate number of clusters. Inversely, there are measures that explicitly seek to minimize similarity between contrasting categories, but inter-category dissimilarity favors a single all-inclusive category, since there are no contrasting categories to share properties with it (Medin, 1983). As a result, inter-category measures are not sufficient either, and clustering methods often require that the user specify the number of categories, n , to be formed.

A second concern with optimization strategies is that the number of ways to partition a data set can be huge. Thus, *iterative optimization* strategies like *k-means clustering* use heuristics to direct the search. If n clusters are to be formed, then an initial set of n observations are drawn from the population. These act as 'seeds' around which the n categories are grown. The remaining, nonseed observations are placed in categories with the most similar seed observations. After all observations have been classified, the centroid of each category is computed and these are used as new seeds. All observations are classified with respect to these seeds as before. The process iterates until some measure of convergence or stability is achieved (e.g., observations are identically categorized over two iterations).

Finally, the basic optimization strategy can be modified to form overlapping clusters, where observations may be members of more than one cluster. Everitt (1980) describes one method that simply partitions the data beginning at different starting or seed points. Rather than treating these as distinct alternatives, the partitions are 'merged' into a single set of overlapping clusters. It is also easy to imagine how a hierarchical (e.g., agglomerative) approach could be modified to form

overlapping classifications: rather than merging an observation with a best matching category, one places it with some number of best matching categories. Collectively, methods that form overlapping clusters are known as *clumping* techniques.

Before turning to other methods of unsupervised ‘learning’, it is important to note that clustering does not occur in a vacuum. Rather, it is used as one tool in exploratory data analysis (Tukey, 1977). Typically, a data analyst must be concerned with modifying the data as appropriate, such as scaling numerical data, selecting ‘relevant’ features, or weighting the importance of features. In addition, methods must be used to devise functions that classify new data points into clusters after they have been constructed. Of course, many of the measures used to form clusters (e.g., distance from the centroid) can be adapted to classification, but it is important to keep the distinction between clustering and classification in mind. In general, the use of clustering in exploratory data analysis requires the analyst to iterate between data engineering, clustering, and classifier design until he or she finds clusterings (and classifiers) that suit the particular needs of the study.

3.2 Clustering Methods in Machine Learning

Despite the widespread popularity of clustering techniques of numerical taxonomy, there are some important limitations. First, most similarity measures are best suited to numerical data. Techniques for dealing with nominal data (e.g., proportion of identically valued attributes in two observations) are problematic in many contexts. Second, the application of clustering is an iterative, search-based process, with much of the responsibility of search control left to a human user. Methods of clustering in machine learning have been developed with two primary goals in mind: to facilitate application in AI domains where nominal data is frequently found, and to incorporate some of the human analyst’s search into the ‘clustering’ program itself.

Recently, machine learning researchers have developed methods for *conceptual clustering* (Stepp, 1987). An important aspect of these systems is that the development of a clustering and a classifier are not independent – a partitioning of data is viewed as ‘good’ if and only if each cluster has a ‘good’ conceptual interpretation. Fisher and Langley (1985, 1986) view conceptual clustering as composed of two subtasks:

- *clustering*, which determines useful subsets of an object set; and
- *characterization*, which determines a concept for each extensionally defined subset discovered by clustering.

The clustering subtask is the focus of work in numerical taxonomy, whereas characterization is the focus of supervised learning systems.

CLUSTER/2 (Michalski, 1980; Michalski & Stepp, 1983a, 1983b) is the original and best known conceptual clustering system. It forms categories that have ‘good’ conjunctive expressions of features that are common to all or most category members. One criterion of goodness in **CLUSTER/2** is *simplicity*, which prefers short conjunctive expressions for the sake of comprehensibility. A second criterion, *fit*, prefers detailed (specific) conjunctive descriptions because these descriptions convey more characteristics about category members. These criteria are used to guide an iterative optimization procedure. A good overview of the method can be found in Fisher and Langley (1986).

The interested reader can see Michalski and Stepp (1983a) and Dale (1985) for a discussion of the relative merits of **CLUSTER/2** and numerical taxonomy methods for data analysis. However, the real value of conceptual clustering, as opposed to its original realization in the form of **CLUSTER/2**, is that it explicitly couples characterization and clustering. This is not to say that numerical taxonomy is not concerned with characterization, but the interface between clustering and classifier development is largely manual. In conceptual clustering, these two aspects are not independent, and in most cases they are tightly coupled routines. **CLUSTER/2** takes a first step towards automating this coupling, thus suggesting that the interpretation tasks of data analysis can be automated along with clustering.

CLUSTER/2 builds classification schemes for which ‘good’ conjunctive descriptions exist, but we have seen that our view of concepts need not be restricted to these types of descriptions: probabilistic concepts are of increasing interest. **WITT** (Hanson & Bauer, 1989) is one system that maintains probabilistic concepts. It forms clusters using a strategy similar to the k -means algorithm. Clusters are formed that maximize pairwise feature correlations within clusters and minimize the average correlations between clusters. Because it optimizes a *tradeoff* of within and between category quality, **WITT** need not be told the number of

classes to form, as must many optimization methods of numerical taxonomy. Rather, a ‘suitable’ number optimizes its quality measure.

AUTOCLASS (Cheeseman, Kelly, Self, Stutz, Taylor, & Freeman, 1988) also employs a probabilistic representation, but uses a Bayesian method to calculate the ‘most probable’ categories present in the observations. Intuitively, the most probable classes are those with attribute-value distributions that differ most from the population as a whole — those that were least likely to have arisen by a ‘random’ partitioning of the data. The most probable number of classes also emerges from the data — smaller classes must exhibit greater skew in their attribute-value distributions to be accepted as clusters, thus biasing AUTOCLASS against numerous classes unless the data warrant them. Conversely, there is a natural bias against overly large classes because value distributions will have a greater *a priori* tendency to match the value distributions of the population as a whole. As with WITT, AUTOCLASS need not be told the number of clusters to form since a number naturally arises in the optimization of a suitable tradeoff function. Cheeseman et al.’s system is also unusual in that the clusters formed by the system are interpreted as probabilistic: an observation is assigned to each cluster with a probability, rather than being assigned to one cluster with probability 1.0, which is the implicit assumption with other (supervised and unsupervised) systems that we have examined. This interpretation should not be confused with the interpretation of a clustering assumed by fuzzy systems (Bezdek, 1987), which assume that observations vary in their *degree of membership*, not their probability of membership.

3.3 Psychological Perspectives on Clustering

In psychology, a task that corresponds to clustering is *sorting*: items are shown to a subject with instructions to partition them into two or more categories (Medin, Wattenmaker, & Hampson, 1987; Ahn & Medin, 1989). An initial supposition was that subjects would sort based on principles of maximizing intra-cluster similarity and minimizing inter-cluster similarity found in numerical taxonomy and machine learning (e.g., WITT). In psychology, these aspects of similarity correspond to *family resemblance* principles, which Rosch and Mervis (1975) hypothesized were responsible for typicality effects (see Section 2.1). However, experiments showed that subjects tended to sort based on a single dimension (Medin et al., 1987); e.g., they created categories that con-

tained items of the same color. Nonetheless, some data from Medin et al. (1987) seemed to indicate that if a unidimensional rule could not strictly partition the data, then subjects would still use a single dimension to approximate a partition, and follow this by a secondary process that converged on a perfect partition.

In response, Ahn and Medin (1989) developed a two-stage categorization model that initially divided a data set into categories based on the most extreme values (e.g., large and small) of the most salient dimension (e.g., size), and then placed the remaining objects without extreme values (e.g., medium) into categories based on overall similarity (i.e., in much the same way as a numerical method would do). Depending on domain, this second stage leads to family resemblance sorting in cases where this is also observed in humans, and to unidimensional sorting when the absence of exceptional cases obviates the need for a second stage. Thus, this model and humans appear to use considerably different rules than the computational methods that we described, since the latter have no analog to a unidimensional first stage. However, the following section suggests extensions to these studies that might reconcile these psychological findings and the behavior of most computational models. Ahn (1990) also extends the results in some important ways, as described by Fisher and Pazzani (Chapter 6, this volume).

3.4 Issues of Unsupervised Learning

Unsupervised systems have not been as widely studied as supervised ones, but they are still quite varied in technique and methodology. Here we summarize some important principles taken from across the perspectives of data analysis, machine learning, and psychology.

3.4.1 THE ENVIRONMENT AND PERFORMANCE

Unsupervised systems are subject to many of the same complexities as supervised systems. For example, most of them assume featural or attribute-value representations. However, there are a few systems that operate on relational data (Stepp, 1984; Holder, 1987). Vere's (1978) THOTH system uses an agglomerative approach over relational descriptions.⁴ Initially, observation pairs are considered for merging.

4. THOTH has sometimes been interpreted as a supervised system that learns disjunctive descriptions from positive examples only (Dietterich & Michalski, 1983).

We have noted in discussing supervised systems that there may be many ways to generalize over relational objects, and THOTH considers all possible maximally specific concepts for each pair. These concepts (and their constituent observations) then become candidates for further merging. THOTH does not exploit heuristics like CLUSTER/2's fit and simplicity, but instead it looks for a minimal set of maximally specific concepts that cover all of the observations. Vere's work illustrates that relational descriptions can considerably increase the search required for unsupervised, as well as supervised, systems.

There are also some important similarities between supervised and unsupervised systems in terms of performance. As we have noted, supervised systems are typically used to predict membership with respect to *a priori* known classes. In fact, clustering is often used to create classifications that can be used for precisely the same performance task (Romesburg, 1984). For example, Cheng and Fu (1985) used the output of a conceptual clustering system to diagnose disease in the domain of traditional Chinese medicine. The discovered concept descriptions did a good job of diagnosis and corresponded closely to the classification scheme typically used by experts. In numerical taxonomy this is related to 'relative' validation methods (Jain & Dubes, 1988), which compare discovered classes with those that were expected to exist in the data (Stepp, 1984). However, 'rediscovery' and/or the performance task of supervised systems, where class labels are known, need not be the sole evaluation methods for unsupervised systems.

Rather, we can abstract out an alternative performance task for unsupervised learning from work in machine learning (Lebowitz, 1982; Kolodner, 1983; Fisher, 1987c; Cheeseman et al., 1988) and pattern completion (Duda & Hart, 1973): unsupervised learning supports prediction, not simply of a single class label, but of arbitrary information that is missing from observations. To illustrate we turn to some empirical data from Fisher's (1987b) unsupervised COBWEB system. The graph of Figure 4 compares prediction accuracy between an unsupervised system called COBWEB and a reimplementation of ID3 on the 36 attributes of a soybean disease domain (Stepp, 1984). In the case of ID3, one decision tree was generated to predict the values of each attribute (i.e., 36 decision trees). In contrast, one classification tree was

In general, such systems (Stepp, 1979) can be interpreted as unsupervised, since they try to partition uniformly labeled data in the best possible way. The partition elements correspond to different conjuncts of a DNF expression.

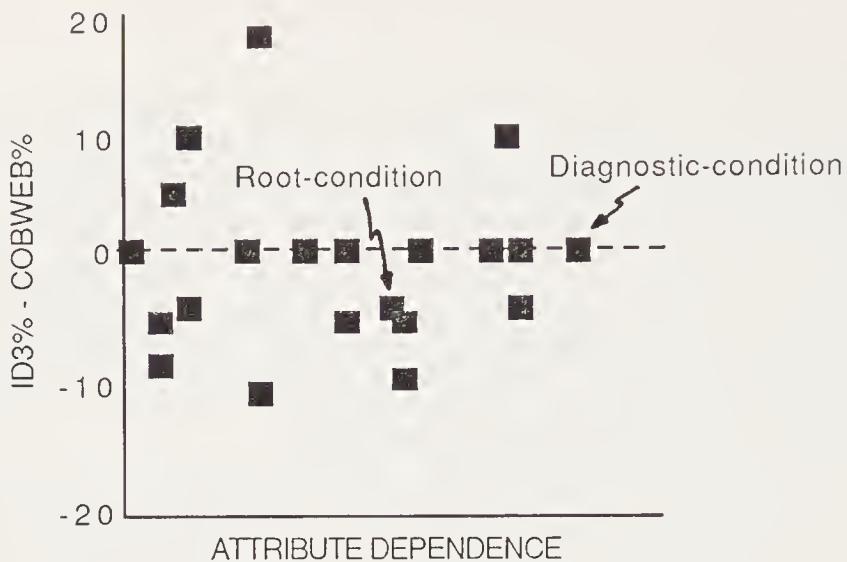


Figure 4. Attribute prediction in unsupervised versus supervised contexts from Fisher (1987b).

produced by the unsupervised method and used to make predictions for all attributes. The vertical axis represents the *difference* between the accuracy levels of the two methods. For example, COBWEB predicted root condition of diseased plants correctly 100% of the time and ID3 predicted it correctly 96% of the time, thus giving a difference of -4%. On average, the approaches performed comparably, with some advantage for the multiple ID3 trees. Fisher (1987a, 1987b) qualifies the comparison in some important ways and explains the importance of the horizontal axis, but our main objective now is to illustrate a performance task that several researchers have associated, implicitly or otherwise, with unsupervised systems.

In sum, evaluation is an important activity in unsupervised learning, but it is often not strongly linked to an overt performance task as are supervised systems. In general, we believe that missing-attribute prediction can be better exploited as an evaluation strategy for purposes of comparing clusterings and comparing clustering systems. Novel work along these latter lines has been performed by Mahoney and Mooney (1989), who compared the performance of Fisher's COBWEB and an unsupervised, connectionist strategy known as *competitive learning* (Rumelhart & Zipser, 1985). We also believe that concerns with prediction accuracy may qualify certain psychological results: recall that subjects were found to cluster based on a unidimensional rule. If the perceived 'performance task' was to explain their classifications to the

experimenter, then the simple logical rules that subjects chose may be optimal. An interesting extension to these experiments would impose a prediction-based performance task. A working hypothesis is that an attribute prediction task will induce family resemblance sorting, even at the top division of data.⁵ In general, assumptions about performance are vital in the design of clustering and sorting models, though they are often left implicit or not considered at all.

3.4.2 THE LEARNING COMPONENT

Our discussion of numerical taxonomy included hierarchical, optimization, and clumping strategies. Hierarchical techniques were further decomposed into agglomerative and divisive techniques. This taxonomy can also be used to characterize machine learning strategies (Fisher & Langley, 1985, 1986), but an alternative scheme is to view them along the same search dimensions that were used to describe their supervised counterparts. All methods conduct a search of the space of data clusterings (partitions). This search depends on a subordinate search through the space of characterizations for each candidate partition. In addition, hierarchical methods can be viewed as conducting a search through the space of hierarchies. This search depends on a subordinate search through the space of partitions to determine how the hierarchy is to be partitioned at each level. Thus, some systems can conduct a three-tiered search (hierarchies, clusterings, characterizations).

We will not detail most of these search processes here, since they are of limited relevance to future discussion, but two points are worthy of mention. First, search at the lowest level — through characterizations — demonstrates all of the variance in search control and search direction that we discussed earlier relative to supervised systems. Second, the search for clusterings include operators that *merge* existing clusters into larger clusters (e.g., as in agglomerative methods) and *split* clusters into groups of finer granularity (e.g., as in divisive methods). Together, these operators define the direction that search can proceed: from finer to coarser granularity (Vere, 1978) or vice versa (Fisher, 1984; Langley & Sage, 1984). This latter point will become relevant in our discussion of concept formation systems, which — like their incremental, supervised counterparts — depend on bidirectional mobility.

5. Fisher and Pazzani (Chapter 6, this volume) discuss related ways that Medin, Ahn, and colleagues were able to induce family resemblance sorting.

3.4.3 THE KNOWLEDGE BASE

Different organizations of the knowledge base facilitate different types of search during learning. For example, data analysts are often interested in strict partitions of mutually exclusive clusters. Even so, hierarchies supply a degree of flexibility in finding a suitable partitioning of data. That is, the data analyst can sever the hierarchy at ‘appropriate’ points to obtain a flat clustering. In this case, the search for a flat partitioning is split between a hierarchical clustering method and the data analyst. Duda and Hart (1973) suggest the full automation of this bipartite procedure, and WITT provides a partial realization of this idea. However, it is best crystalized in a more recent system by Biswas, Weinberg, Yang, and Koller (1991). In each case, a hierarchical method is used to initialize the starting points of an optimization technique.

Finally, overlapping clusters are generally motivated in domains where there are *a priori* beliefs that observations can be members of multiple classes. This is true in the case of language (Everitt, 1980), where words might have multiple meanings and/or satisfy different parts of speech. This is precisely the motivation for Wolff’s (1978) clumping system, SNPR, which induces grammatical classes from sample sentences. However, we might view overlapping clusters as supplying a type of flexibility that is similar to that attributed to trees. Multiple partitions are implicit in a knowledge base of overlapping clusters, and the data analyst can extract those partitions that are deemed most useful. In fact, this motivation seems implicit in reports of GLAUBER (Langley, Zytkow, Simon, & Bradshaw, 1985), a ‘scientific discovery’ system and descendent of SNPR, which extracts some clusterings (e.g., salts, alkalis, and acids) as scientifically more interesting than numerous other options. The importance of overlapping classifications is even more explicit in probabilistic (Cheeseman et al., 1988) and fuzzy (Bezdek, 1987) clustering, which assign an observation a probability or degree of membership to each cluster.

4. Concept Formation

The unsupervised systems that we have described so far are nonincremental, requiring all training instances at the outset. However, in many cases human learners appear to assimilate instances as they become available. We refer to this process — the incremental unsupervised

acquisition of categories and their intensional descriptions — as *concept formation*. This section deviates somewhat from the organization of the past two sections by specifically summarizing four influential concept formation systems.⁶ This treatment paves the way for their discussion in future chapters.

4.1 EPAM

EPAM (Feigenbaum, 1961) was originally designed as a model of rote learning behavior (e.g., of nonsense syllables), though it has since been used to model a significant and varied set of phenomena, including word recognition and chess patterns (Feigenbaum & Simon, 1984; Richman, this volume). Like ID4 and ID5, EPAM builds a decision tree over a stream of observations, and classification at each node is directed by an observation's value along a single attribute. However, there are several important differences. First, EPAM does not assume that all of an attribute's possible values are known *a priori*. Rather, arcs emanating from a node may include special 'else' values that are matched by an observation with a previously unobserved value. Second, EPAM's learning task is unsupervised; observations are not preclassified and thus the leaves of an EPAM tree do not include category labels. Instead, a leaf contains a pattern or concept that matches a subset of the observations; concepts are not explicitly associated with internal nodes of the tree.

A new observation is classified down a path of matching arcs to a leaf. If the observation matches the concept stored at the leaf, then a process of *familiarization* occurs: the leaf is specialized by adding a feature that is present in the observation and not present in the current leaf. If there is a mismatch of one or more features, then one of two actions is taken. The observation and pattern may disagree on a dimension corresponding to an 'else' link that was taken during classification. If so, then this point of disagreement is found and two new arcs are created, one with the value of the new observation and one with the value of the leaf. The latter arc leads to the same subtree as before. The former arc leads to a new leaf that is constructed to fit the new observation, thus increasing the breadth of the tree. This leaf contains all of the features of the observation that were used in tests during classification. If no disagreement along the classification path is found, then the difference

6. Our discussion of individual systems owes much to the treatment given by Genzani, Langley, and Fisher (1989).

that triggered the original search is used to differentiate the leaf's pattern and a new leaf corresponding to the observation, thus increasing the depth of the tree. Again, this pattern contains the features on the path from root to leaf. In sum, leaves begin as general patterns that are gradually specialized by familiarization as subsequent observations are classified.

EPAM is quite simple on a number of dimensions. However, it accounts for a significant amount of experimental data, notably modeling the memorization of stimuli as a general-to-specific search at leaves, which reflects the gradual process of memorization (familiarization) that presumably occurs. In addition, it is undoubtedly the earliest computational model of concept formation and has had a significant influence on the systems that follow, as well as on supervised systems like ID4.

4.2 CYRUS and UNIMEM

CYRUS (Kolodner, 1983) and UNIMEM (Lebowitz, 1982) were designed as general-purpose methods for organizing memory for efficient and intelligent information retrieval. Each was intended to model certain high-level aspects of human learning (e.g., information retrieval is more efficient as humans acquire more knowledge). As with EPAM, each system builds a discrimination network over a sequence of observations and classification is top down, but there are some important differences.

In particular, CYRUS and UNIMEM may index a node from its parent by more than one attribute value. For example, in a hierarchy of animal species, *mammals* might be indexed by *has-hair*, *warm-blooded*, and *births-live-young*. A node is investigated as a candidate host for an observation if the observation possesses at least one of the values that index the node. Moreover, each value may index more than one child of a node; thus, *warm-blooded* might index nodes corresponding to *mammals* and *birds*. An observation is compared with all indexed nodes. UNIMEM and CYRUS may form overlapping categories, but there are two ways that these systems guard against an explosion of comparisons and categorizations.

First, to actually be classified at a node, the observation must match the concept description stored at the node. In initial versions of UNIMEM and its precursor IPP (Lebowitz, 1983), a node's concept was a perfect conjunction of features. A candidate observation had to satisfy this conjunction if it was to be classified under the node. In contrast, CYRUS

and more recent versions of UNIMEM (Lebowitz, 1987) allow exceptions to the conjunction. A feature in a node's description is regarded as *normative* (Kolodner, 1983) of a node's observations. Intuitively, it need not be true of all observations, but it should be *predictable* (Lebowitz, 1982) with high probability. In both systems, an observation is classified under a node if it shares a specified proportion of the node's predictable attribute values. During the top-down classification process, if no child of the current node sufficiently matches the observation, then the observation is made a new child of the node.

To enforce the predictable status of attribute values, each value is weighted. In UNIMEM this is an integer weight; whenever an observation is compared against a candidate concept, an attribute value's weight is incremented if the observation and node agree along this attribute, and the value's weight is decremented in the case of disagreement. If the weight falls below a user-defined threshold (e.g., -2), then the attribute value is dropped from the concept description. When a new node is created (i.e., when an observation does not match any existing node sufficiently well), the predictable values of the new node are those values of the observation that are not predictable values of the new node's parent. Thus, values that are shared by the observation and parent are inherited from the node's parent.

A second way of controlling classification in UNIMEM and CYRUS is to select indices in a relatively conservative manner. For example, in UNIMEM when a new node is created, all values that distinguish it from its parent are used as indices to the node (i.e., values that serve as indices and predictable values are initially the same). However, if this addition causes a value to index more than a user-specified number of categories (e.g., three), then it is removed as an index to any node. Intuitively, indices are used to guide classification and to predict which nodes are likely candidates for an observation. A value that predicts too many candidates has little predictive or diagnostic value. CYRUS employs heuristics with similar intent.

In summary, CYRUS and UNIMEM employ the general top-down classification strategy of EPAM, but they advance the basic scheme in a number of ways. UNIMEM and CYRUS are *polythetic* classifiers (Everitt, 1980), in that classification at each step in the hierarchy is guided by many attribute values, not by a single value as in the *monothetic* approach of EPAM. Classification in these latter systems is guided by the

predictiveness and *predictability* of several of an observation's features: predictive, indexing features are used to suggest candidate categories for classification, but an observation must satisfy a partial matching criterion over a node's predictable or normative features. Both systems may form overlapping categories, since observations are classified into each node that they match sufficiently well.

4.3 COBWEB

In many respects, Fisher's (1987) COBWEB was intended to synthesize principles of earlier machine learning systems, notably UNIMEM and CYRUS, and certain results from cognitive psychology. In particular, COBWEB was initially inspired by research on *basic level* effects. In classification hierarchies humans tend to prefer one 'basic' level of abstraction over others. For example, humans can typically verify that an item (e.g., a picture of a particular robin) is a bird more quickly than they can verify the same item is an animal, vertebrate, or robin. Evidence of this type (Rosch, Mervis, Gray, Johnson, & Boyes-Braem, 1976) and from a variety of other experimental settings (Mervis & Rosch, 1981; Lassaline, Wisniewski, & Medin, in press) indicate that within hierarchies containing animals, vertebrates, birds, and robins, the concept of birds resides at the preferred or basic level. COBWEB's design assumes that principles which dictate preferred concepts in humans are good heuristics for machine concept formation as well.

Several measures have been proposed to account for basic level effects (Rosch et al., 1976; Jones, 1983). COBWEB uses Gluck and Corter's (1985) *category utility* measure, which assumes that basic level concepts maximize

$$CU(C_k) = P(C_k) \left[\sum_j P(V_j|C_k)^2 - \sum_j P(V_j)^2 \right] \quad ,$$

where $P(C_k)$ is the proportion of a population covered by a category, C_k ; $P(V_j|C_k)$ is the category validity or predictability of value V_j relative to C_k ; and $P(V_j)$ is the base rate probability of V_j . Category utility is a tradeoff between the degree that C_k increases the *expected number* of correct predictions about the presence of attribute values over the uninformed case (i.e., the bracketed term) and the proportion of the population to which this increase applies given by the $P(C_k)$ term. Thus, the function rewards categories from which much can be

predicted (i.e., highly specific categories), but where predictions apply to a large proportion of the environment (i.e., highly general categories). A partition of intermediate generality, roughly corresponding to a basic level, will tend to optimize this tradeoff. Thus, like most of the unsupervised machine learning systems that we have examined, COBWEB need not be told the number of classes to form.

COBWEB assumes that each node in the classification hierarchy is a probabilistic concept, with a probability $P(V_j|C_k)$ for each feature at a node. Base rate probabilities over an entire population of observations are stored at the root of the hierarchy. A new observation is added by first updating the base rate probabilities at the root to reflect the observation's attribute values. Each child is then evaluated as a possible host. The conditional probabilities are tentatively updated to reflect the observation's values; the children are ordered by the average category utility of the partition of n categories (i.e., $[\sum_{k=1}^n CU(C_k)]/n$) that they yield after tentative update. This ordering reflects the merit of each existing category to include the new observation. In most cases the observation is placed in the best existing host: its probabilities are permanently updated, and it (and its probability distributions) serve as the root (and base rates) in a recursive application of the classification procedure. However, in some cases an observation is distinct enough from the existing categories that it is used to initialize a new category. This is determined by evaluating the average category utility of the $(n + 1)$ element partition created by adding a new singleton category. COBWEB also has three additional operators — *merging*, *splitting*, and *promotion* — which we will discuss in Section 4.4.2.

The system incorporates an observation along a path of best-matching nodes, whereas UNIMEM and CYRUS classify along all nodes that 'adequately' match. Thus, COBWEB partitions the observations at each level and forms a strict tree. Also, the system stores all attribute values and their distribution at every node to guide classification. However, note that if an attribute value is independent of membership in a category, then $P(V_j|C_k) \approx P(V_j)$, and $P(V_j|C_k)^2 - P(V_j)^2 \approx 0$. That is, V_j will be 'irrelevant' to a category's score and presumably to an observation's membership in C_k . Thus, the representation scheme does not explicitly encode inheritance, but the subtraction of base rate probabilities (at the parent) from the node's distribution has the same impact from the standpoint of classification. Moreover, COBWEB does not explicitly distinguish predictable or predictive values, but by moving $P(C_k)$ into the

summations and applying Bayes' rule, we can reexpress category utility as $\sum_j P(V_j)P(C_k|V_j)P(V_j|C_k)^2 - \sum_j P(V_j)P(C_k)P(V_j)$. The first term is a function of both the cue validity, $P(C|V)$, and category validity, $P(V|C)$ of features, which provides a formal interpretation of predictiveness and predictability, respectively. Thus, the function can be viewed as a family resemblance measure that is responsible for COBWEB's account of typicality and fan effects (Anderson, 1974), as well as basic level phenomena (Fisher & Langley, 1990).

4.4 Issues of Concept Formation

The relevant dimensions that we use to compare and discuss concept formation models are inherited from our discussion of unsupervised and incremental learning. This section summarizes these issues relative to the concept formation systems surveyed above and highlights other systems in the literature.

4.4.1 THE ENVIRONMENT AND PERFORMANCE

Our analysis of incremental, supervised systems considered four performance dimensions: (1) the cost of assimilating individual observations, (2) the asymptotic accuracy levels that are obtained, (3) the number of observations before reaching asymptotic or some other specified accuracy level (i.e., the learning rate), and (4) the total cost (i.e., cost per observation times the number of observations) required to obtain a desired level of accuracy. Concept formation systems can be evaluated along the same dimensions; most seek to minimally trade accuracy for as great a savings as possible in terms of assimilation cost. For example, Fisher (1987) and Lebowitz (1987) show by analytic and empirical means that average assimilation cost in COBWEB and UNIMEM is logarithmic due to the hierarchical structures produced by these systems. Assimilation cost has been a primary focus of Gennari's (1989) CLASSIT, Decaestecker's (1989) ADECLU, and Hadzikadic and Yun's (1989) INC, which are concerned with how humans and resource-limited machines learn to focus *attention* on informative features like the predictive and predictable ones found in UNIMEM and CYRUS. Recall that the subtraction of base rates in category utility flags irrelevant features, but comparisons are still made. CLASSIT, ADECLU, and INC exploit similar observations to selectively focus attention.

Like unsupervised tasks more generally, the prediction of unknown attributes is an important performance dimension, but incremental learning is concerned with learning rate over time as well as asymptotic performance. As we have seen, though, noise can confound feature intercorrelations and degrade prediction. Gennari, Langley, and Fisher (1990) experimented with a simple method of pruning classification trees produced by CLASSIT, a descendent of COBWEB that mitigates data overfitting. In particular, they pruned the children of a node if they failed to bring about a sufficient gain in information (i.e., their average category utility score fell below a specified threshold). Anderson and Matessa (this volume) describe a Bayesian hierarchical technique that is an incremental analogue to AUTOCLASS. Tree decomposition terminates prior to instance-level descriptions if such a decomposition is viewed as improbable. This ‘pruning’ strategy is similar to that used by CLASSIT. However, pruning at a particular level may not be optimal for all attributes.⁷ In this regard, Fisher (1989) describes two methods for determining attribute-specific optimal prediction points that are based on pruning heuristics used by decision tree learners (Mingers, 1989).

Finally, there has been considerable work using structural representations. RESEARCHER (Lebowitz, 1986) and MERGE (Wasserman, 1985) descend from UNIMEM. They have been applied to concept formation over complex physical devices (e.g., disk drives and other electromechanical devices) that are best represented structurally. Nevin’s (1990) HIERARCH system builds on the UNIMEM approach, but adds information-theoretic guidance in a manner similar to COBWEB. Levinson (1984) did early work on clustering general graph structures, which can be viewed as structured descriptions of binary predicates of the same name (i.e., each represents an unlabeled arc between nodes). His early studies were in the domain of molecular chemistry, and later work has focused on clustering patterns in chess games. Levinson’s system also influenced recent work by Wogulis and Langley (1989). Segen (1990) has also developed a general graph-clustering algorithm that exploits information-theoretic (minimum description length) principles, whereas Thompson and Langley (this volume) use a measure similar to category utility.

7. Pruning in CLASSIT has also been motivated for efficiency reasons — as a way of mitigating logarithmic growth in the number of comparisons during assimilation and as a way of mitigating linear growth in the number of classes stored. These are legitimate reasons for pruning beyond the issue of prediction accuracy.

4.4.2 THE LEARNING COMPONENT

Concept formation systems can be characterized along the dimensions of search that were used to describe earlier paradigms. Notably, several of the systems that we surveyed can be viewed as specific-to-general learners. COBWEB, UNIMEM, CYRUS, and related systems by Decaestecker (1989) and Hadzikadic and Yun (1989) all begin with very shallow hierarchies in which specific instances appear near the top. As more observations are assimilated, they are generalized with existing nodes during classification, thus creating more general, intermediate nodes. In contrast, EPAM is best viewed as a general-to-specific learner; it initializes leaves to be very general patterns that are gradually specialized. Another general-to-specific learner is Martin's (1989) CORA system, which incrementally conjoins features that are highly correlated. The system does not form an abstraction hierarchy, but simply creates a series of conjunctive concepts that describe (possibly) nondisjoint categories.

Fisher (1989) and Fisher and Chan (1990) have argued that incremental, specific-to-general learners can more quickly exploit relevant information for purposes of prediction than can their general-to-specific counterparts. For example, a system can ideally distinguish mice from men with high, but not perfect, accuracy after a single example of each. Martin and Billman (this volume) are also intimately concerned with search direction, and they present experimental evidence in support of a specific-to-general preference. Despite this, it is important to keep in mind that some important assumptions are being made.

We will touch upon two caveats to the specific-to-general bias, but there are others. First, we are assuming that the model allows partial matching. In cases where 'perfect matching' (e.g., of conjunctive conditions) is required, a specific-to-general strategy may overly constrain the learner's behavior to fit the observed instances (Langley, 1985). Second, most specific-to-general systems assume that one can remember a single observation in its entirety. However, research with EPAM (Richman, this volume) has focused on modeling human visual and auditory perception, including limitations on simultaneous processing. Martin and Billman's work, Gennari's (1989) CLASSIT, Decaestecker's (1989) ADECLU, and Hadzikadic and Yun's (1989) INC also address these issues, particularly (as we have noted) how humans and resource-limited machines learn to focus attention on informative features. In cases where attention is limited, a general-to-specific learner may be the most plausible model.

Although approaches to concept formation differ in search direction, many share the hill-climbing search control of their incremental counterparts for supervised learning. Mechanisms for concept formation are designed to be rational but resource-bounded learners (Simon, 1969). Each observation triggers small changes to the current categorical structure. As such, concept formation systems suffer from ordering effects, in that they may discover different categories depending on the order in which they process observations. Fortunately, random orderings tend to yield trees that still facilitate prediction and approximate the form of trees that would be obtained by a more extensive search (Anderson & Matessa, this volume; Fisher, 1987c; Knuth, 1973). Ironically, however, the random ordering assumption is not valid in many situations that are designed to facilitate 'learning'. For example, teachers usually present facts and problems in order of 'similarity'. Zoos cluster animals based on similarity so that patrons are likely to experience a very skewed ordering. The problem with such orderings relates to an observation by Medin (1983): many functions used for categorization and clustering are strongly biased to place new observations into larger categories (e.g., in category utility this bias stems from the $P(C_k)$ term). A highly skewed ordering may establish several large categories before observations in all parts of the description space are experienced, thus biasing their placement towards the existing categories.

The likelihood of biased orderings has led many researchers to adopt more robust search mechanisms. For example, UNIMEM and CYRUS delete a node and its associated subtree if the node's corresponding concept becomes poor by a criterion similar to CLUSTER/2's fit measure. This lets a new subtree be grown to reflect the characteristics of future data. COBWEB includes a *merging* operator that combines existing categories, and an 'inverse' *splitting* operator that breaks existing categories into smaller ones. Collectively, these operators provide bidirectional search capabilities that can be used to approximate backtracking, as in supervised systems like ID4 and ID5. An alternative approach used by Lebowitz (1988) and Hanson and Bauer (1989) is to buffer observations until a significant number become available. At this point the buffered node is expanded and a set of children is created.

Undoubtedly, all of these approaches mitigate ordering effects to some extent, but the application of these operators lacks heuristic foresight (Ahn, personal communication); they are only triggered by and impact very local conditions in a hierarchy. One research area is the develop-

ment of more global reorganization strategies for hierarchical methods. For example, early versions of COBWEB (Fisher, 1987a) did not delete ‘poor’ subtrees, but looked for better placement for them within the hierarchy. This strategy has been considerably expanded and improved by McKusick and Langley (in press), and a similar approach is also employed by Nevins’ HIERARCH. In particular, McKusick and Langley’s ARACHNE system looks for a class that is ‘less similar’ to its parent than its grandparent. Such a class is likely to be misplaced; it is promoted and made a child of the better matching ancestor. Presumably, further processing (e.g., merging) will then integrate the transplanted category into its surroundings and/or move it still farther up the hierarchy. This procedure can be viewed as an incremental variant on the hierarchical/optimization hybrids that we discussed in Section 3.4.3 (e.g., Biswas et al., 1991): objects and classes are moved (perhaps over a stream of many observations) to better-matching categories in a partition. Thus, this work illustrates a type of more ‘global’ processing intended to approximate the quality of nonincremental systems, while limiting the cost of assimilating individual observations.⁸

4.4.3 THE KNOWLEDGE BASE

Appropriate structures in a knowledge base can significantly improve the efficiency of both supervised and unsupervised incremental systems. Concept formation systems almost universally form abstraction hierarchies. They even do so when the primary objective is to extract a flat partition of the observations. Cheeseman (1990) has objected that many of these systems (e.g., COBWEB) impose a tree structure, even when none is warranted. However, we have seen that trees flexibly allow partitions to be extracted during data analysis. In this regard, Anderson and Matessa’s system can extract the ‘basic’ level following the construction of a classification tree (see Section 3.4.3). This flexibility becomes considerably more important in an incremental setting. Consider Fisher’s COBWEB, which can be alternately viewed as building a classification tree, or as discovering a partition that corresponds to the basic level (i.e., the top level). In the latter interpretation, the tree is used to constrain search: if a node at the topmost level appears ill-suited to the basic level, then one may want to split it up in the best possible

8. This observation grew out of discussions with Kathleen McKusick, Pat Langley, Jerry Weinberg, and Gautam Biswas.

way and combine it with other members of the top level. Rather than examining many ways of breaking up such a node, COBWEB assumes that the node's children are a good approximation of the best partition, which is promoted and is subject to further processing. In general, the tree structure supplies a restricted memory of alternative partitions from which one can choose during incremental processing. However, as we have noted, the primary difficulty is insufficient heuristic foresight. We have seen how ARACHNE extends foresight in one way, but others are possible (Reich & Fenves, this volume).

In addition to hierarchical structuring, many systems form overlapping categories, much like clumping in numerical taxonomy (Martin & Billman, this volume; Scott & Markovitch, this volume). For example, both UNIMEM and CYRUS incorporate observations into all 'satisfactory' categories. As we noted in Section 3.4.3, clumps offer an additional way to flexibly constrain search. Even if one is interested in a partition of data, clumps compactly represent many alternatives. In effect these systems carry out a 'beam' search of the best partitions. In fact, systems by Levinson (1983) and by Wogulis and Langley (1989) form overlapping categories, in part to better manage the increased search that is required by structural descriptions. After training is complete, there may be multiple partitions that are justified by the data, and it may be advantageous for purposes of prediction to retain them all. For example, having observed a stream of animals, one may conclude that {mammals, reptiles, fish, birds, amphibians} and {herbivores, omnivores, carnivores} are both useful partitions – they capture orthogonal correlations of features that will contribute towards attribute prediction accuracy. Finally, clumps may be psychologically more plausible ways of structuring data. For example, studies summarized in Smith and Medin (1981) indicate that chicken is treated as less similar to bird than to animal. The apparent contradiction stems from an assumption that chicken, bird, and animal constitute a single path within a strict tree. However, if we admit multiple paths between categories (e.g., as in a directed acyclic graph), then paths between ancestrally related nodes need not be 'equilength'. Rather, chicken may be 'closer' to animal via one path than it is to bird via another.

In addition to similarities in hierarchical structures of the knowledge base, concept formation systems also exhibit considerable uniformity in the representation of individual concepts. Most systems use probabilistic summaries of one form or another because they grade inconsis-

tency (or typicality) and because they are more flexible in the face of inconsistencies that are liable to arise during incremental processing. Hierarchically organized probabilistic concepts also allow all portions of a feature description space to be represented — individual concepts are limited to linearly separable categories, but systems of such concepts are not. We have also noted the importance of features that focus attention (Gennari, 1989; Decaestecker, 1989; Hadzikadic & Yun, 1989; Lebowitz, 1982; Kolodner, 1983) to improve efficiency in probabilistic schemes. For example, UNIMEM and CYRUS exploit predictive values to index a constrained set of candidate nodes, which are then evaluated using predictable values. There are several ways that UNIMEM-like indexing schemes can be improved. With respect to supervised systems, Bareiss (1989) suggests *opportunistic* indexing that can skip levels in a classification hierarchy. For example, if an observation has hair, then one can bypass the vertebrate node on the path to mammal. Fisher and Langley (1990) have formalized this idea in the context of a cognitive model of fan, basic level, and typicality effects based on COBWEB.

Of course, there are exceptions to the probabilistic bias. Most notably, humans appear to prefer unidimensional or conjunctive category formation (Medin et al., 1987; Ahn & Medin, 1989). We have previously suggested that an appropriate performance task may elicit family resemblance clustering. In addition, we should remember that sorting studies presented observations to subjects *en masse* so that sorting was performed in a ‘nonincremental’ manner. An informative extension to these studies would require that subjects form categories incrementally. Experience with machine learning systems (see Section 2.3.3) suggests that subjects might be more inclined to form probabilistic categories given the greater elasticity of these representations in the face of (1) inconsistencies that arise over a stream of data, and (2) memory limitations that restrict the set of conjunctive descriptions they can remember.

5. Concluding Remarks

This chapter has reviewed supervised and unsupervised models of inductive learning. The chapters that follow elaborate aspects of our discussion as it relates to concept formation. However, it is worth highlighting some of the most important issues here. We have noted that concept formation, and unsupervised learning generally, are often not viewed as methods of improving performance. Rather, an implicit assumption is

that the primary performance task of interest for unsupervised methods is communicability or perhaps ‘rediscovery’. However, we have suggested that an important evaluation task for these systems is attribute prediction. An explicit consideration of suitable performance tasks can have significant implications on the design of both psychological and computational models of unsupervised learning, but the importance of this observation is sometimes overlooked.

In addition, research on concept formation continues to progress in several directions that are shared by other learning paradigms. For example, important areas concern more complete representation languages for objects and concepts, notably structured descriptions that place an added burden on search. Complications caused by noise in the environment is a traditional research topic in supervised scenarios, and it is receiving increased attention in concept formation and unsupervised learning. Finally, there are issues that apply particularly to concept formation and incremental supervised systems. The most important of these is the development of robust control and flexible representations that can mitigate ordering effects, without overly sacrificing assimilation efficiency. Some promising research is under way, but considerable work remains on such issues as attention, overlapping categories, improved bidirectional abilities, and concept representations.

Acknowledgements

We thank Pat Langley, Woo-Kyoung Ahn, Gautam Biswas, and Kathleen McKusick for comments on style and correctness. The first author was supported by Grant NCC 2-645 from NASA Ames Research Center, and the second author was supported by Grant IRI-8908260 from the National Science Foundation.

References

- Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6, 37–66.
- Aha, D., & McNulty, D. (1989). Learning relative attribute weights for instance-based concept descriptions. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 530–537). Ann Arbor, MI: Lawrence Erlbaum.

- Ahn, W., & Medin, D. L. (1989). A two-stage categorization model of family resemblance sorting. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 315–322). Ann Arbor, MI: Lawrence Erlbaum.
- Anderson, J. R. (1974). Retrieval of propositional information from long term memory. *Cognitive Psychology*, 6, 451–474.
- Anderson, J. R., & Kline, P. J. (1979). A learning system and its psychological implications. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (pp. 16–21). Tokyo, Japan: Morgan Kaufmann.
- Bareiss, R. (1989). *Exemplar-based knowledge acquisition*. San Diego, CA: Academic Press.
- Bezdek, J. (1987). Some non-standard clustering algorithms. In P. Legendre & L. Legendre (Eds.), *Developments in numerical ecology*. Berlin: Springer-Verlag.
- Biswas, G., Weinberg, J. B., Yang, Q., & Koller, G. R. (1991). *Conceptual clustering and exploratory data analysis* (Tech. Rep. No. CS-91-03). Nashville, TN: Vanderbilt University, Department of Computer Science.
- Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.
- Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking*. New York: John Wiley and Sons.
- Cheeseman, P. (1990). On finding the most probable model. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation*. San Mateo, CA: Morgan Kaufmann.
- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54–64). Ann Arbor, MI: Morgan Kaufmann.
- Cheng, Y., & Fu, K. (1985). Conceptual clustering in knowledge organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7, 592–598.
- Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3, 261–284.

- Dale, M. B. (1985). On the comparison of conceptual clustering and numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7, 241–244.
- Decaestecker, C. (1989). Incremental concept formation with attribute selection. *Proceedings of the Fourth European Working Session on Learning* (pp. 49–58). Montpellier, France.
- Dietterich, T. G., Clarkson, K., Dromey, G., & London, R. (1982). Learning and inductive inference. In P. R. Cohen & E. A. Feigenbaum (Eds.), *The handbook of artificial intelligence*. San Mateo, CA: Morgan Kaufmann.
- Dietterich, T. G., & Michalski, R. S. (1983). A comparative review of selected methods of learning from examples. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York, NY: John Wiley & Sons.
- Everitt, B. (1981). *Cluster analysis*. London: Heinemann.
- Feigenbaum, E. (1961). The simulation of verbal learning behavior. *Proceedings of the Western Joint Computer Conference* (pp. 121–132). Reprinted in J. W. Shavlik & T. G. Dietterich (Eds.). (1990). *Readings in machine learning*. San Mateo, CA: Morgan Kaufmann.
- Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, 8, 305–336.
- Fisher, D. H. (1985). *A hierarchical conceptual clustering algorithm* (Tech. Rep. No. 85-21). Irvine: University of California, Department of Information and Computer Science.
- Fisher, D. H. (1987a). *Knowledge acquisition via incremental conceptual clustering*. Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.
- Fisher, D. H. (1987b). Conceptual clustering, learning from examples, and inference. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 38–49). Irvine, CA: Morgan Kaufmann.
- Fisher, D. H. (1987c). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139–172.
- Fisher, D. H. (1989). Noise-tolerant conceptual clustering. *Proceedings of the Eleventh International Joint Conference Artificial Intelligence* (pp. 825–830). Detroit, MI: Morgan Kaufmann.

- Fisher, D. H., & Chan, P. K. (1990). Statistical guidance in symbolic learning. *Annals of Mathematics and Artificial Intelligence*, 2, 135–148.
- Fisher, D. H., & Langley, P. (1985). Approaches to conceptual clustering. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 691–697), Los Angeles, CA: Morgan Kaufmann.
- Fisher, D. H., & Langley, P. (1986). Conceptual clustering and its relation to numerical taxonomy. In W. A. Gale (Ed.), *Artificial intelligence and statistics*. Reading, MA: Addison-Wesley.
- Gennari, J. (1989). Focused concept formation. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 379–382). Ithaca, NY: Morgan Kaufmann.
- Gennari, J., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11–62.
- Gluck, M. A., & Corter, J. E. (1985). Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum.
- Hadzikadic, M., & Yun, D. (1989). Concept formation by incremental conceptual clustering. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 831–836). Detroit, MI: Morgan Kaufmann.
- Hanson, S. J., & Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3, 343–372.
- Hayes-Roth, F., & McDermott, J. (1978). An interference matching technique for inducing abstractions. *Communications of the ACM*, 21, 401–410.
- Holder, L. B. (1988). *Discovering substructure in examples*. Master's thesis, Department of Computer Science, University of Illinois, Urbana, IL.
- Hunt, E., Marin, J., & Stone, P. (1966). *Experiments in induction*. New York: Academic Press.
- Jain, A. K., & Dubes, R. C. (1988). *Algorithms for cluster analysis*. Englewood Cliffs, NJ: Prentice Hall.
- Jones, G. (1983). Identifying basic categories. *Psychological Bulletin*, 94, 423–428.

- Kline, P. J. (1983). *Computing the similarity of structured objects by means of heuristic search for correspondences*. Doctoral dissertation, Department of Psychology, University of Michigan, Ann Arbor, MI.
- Knuth, D. E. (1973). *The art of computer programming: Searching and sorting* (Vol. 3). Reading, MA: Addison-Wesley.
- Kolodner, J. L. (1983). Reconstructive memory: A computer model. *Cognitive Science*, 7, 281-328.
- Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & D. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.
- Langley, P., Gennari, J., & Iba, W. (1987). Hill-climbing theories of learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 312-323). Irvine, CA: Morgan Kaufmann.
- Langley, P., & Sage, S. (1984). Conceptual clustering as discrimination learning. *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence* (pp. 95-98). London, Canada.
- Larson, J. B., & Michalski, R. S. (1978). *Selection of most representative training examples and incremental generation of VL₁ hypotheses: The underlying methodology and the description of the programs ESEL and AQ11* (Tech. Rep. No. UIUCDCS-R-78-867). Urbana: University of Illinois, Department of Computer Science.
- Lassaline, M. E., Wisniewski, E. J., & Medin, D. L. (in press). Basic levels in artificial and natural categories: Are all basic levels created equal? In B. Burns (Ed.), *Percepts, concepts, and categories: The representation and processing of information*. Amsterdam: North Holland Press.
- Lebowitz, M. (1982). Correcting erroneous generalizations. *Cognition and Brain Theory*, 5, 367-381.
- Lebowitz, M. (1983). Generalization from natural language text. *Cognitive Science*, 7, 1-40.
- Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2, 103-138.
- Lebowitz, M. (1988). Deferred commitment in UNIMEM: Waiting to learn. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 80-86). Ann Arbor, MI: Morgan Kaufmann.

- Levinson, R. (1984). A self-organizing retrieval system for graphs. *Proceedings of the Fourth National Conference on Artificial Intelligence* (pp. 203–206). Austin, TX: Morgan Kaufmann.
- Mahoney, J. J., & Mooney, R. J. (1989). *Can competitive learning compete? Comparing a connectionist clustering technique to symbolic approaches* (Tech. Rep. No. AI89-115). Austin: The University of Texas, Department of Computer Science, Artificial Intelligence Laboratory.
- Martin, J. D. (1989). Reducing redundant learning. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 396–399). Ithaca, NY: Morgan Kaufmann.
- McKusick, K. B., & Langley, P. (in press). Constraints on tree structure in concept formation. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*. Sydney: Morgan Kaufmann.
- Medin, D. (1983). Structural principles of categorization. In T. Tighe & B. Shepp (Eds.), *Perception, cognition, and development*. Hillsdale, NJ: Lawrence Erlbaum.
- Medin, D. L. (1989). Concepts and conceptual structure. *American Psychologist*, 44, 1469–1481.
- Medin, D. L., & Schaffer, M. (1978). A context theory of classification learning. *Psychological Review*, 85, 207–238.
- Medin, D. L., Wattenmaker, W. D., & Hampson, S. E. (1987). Family resemblance, conceptual cohesiveness, and category construction. *Cognitive Psychology*, 19, 242–279.
- Medin, D. L., Wattenmaker, W. D., & Michalski, R. S. (1986). *Constraints and preferences in inductive learning* (Technical Report). Urbana: Department of Computer Science, University of Illinois.
- Mervis, C., & Rosch, E. (1981). Categorization of natural objects. *Annual Review of Psychology*, 32, 89–115.
- Michalski, R. S. (1980). Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*, 4, 219–243.
- Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.

- Michalski, R. S., & Stepp, R. E. (1983a). Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5, 219-243.
- Michalski, R. S., & Stepp, R. E. (1983b). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Mingers, J. (1989). An empirical comparison of pruning methods for decision-tree induction. *Machine Learning*, 4, 227-243.
- Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18, 203-226.
- Nevins, A. J. (1990). *A branch and bound incremental conceptual clusterer* (Tech. Rep. No. CIS-90-10-01). Atlanta: Georgia State University, Department of Computer Information Systems.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81-106.
- Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5, 239-266.
- Pazzani, M. J., & Sarrett, W. (1990). Average case analysis of conjunctive learning algorithms. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 339-347). Austin, TX: Morgan Kaufmann.
- Reinke, R. E., & Michalski, R. S. (1988). Incremental learning of concept descriptions. In J. E. Hayes, D. Michie, & J. Richards (Eds.), *Machine intelligence* (Vol. 11). Oxford, England: Oxford University Press.
- Romesburg, H. C. (1984). *Cluster analysis for researchers*. Belmont, CA: Lifetime Learning Publications.
- Rosch, E., & Mervis, C. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7, 573-605.
- Rosch, E., Mervis, C., Gray, W., Johnson, D., & Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, 18, 382-439.
- Rumelhart, D. E., & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, 9, 75-112.

- Salzberg, S. (in press). A nearest hyperrectangle learning method. *Machine Learning*.
- Schlimer, J. C., & Fisher, D. (1986). A case study of incremental concept induction. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). Philadelphia, PA: Morgan Kaufmann.
- Schlimer, J. C., & Granger, R. H., Jr. (1986). Incremental learning from noisy data. *Machine Learning*, 1, 317–334.
- Segen, J. (1990). Graph clustering and model learning by data compression. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 93–100). Austin, TX: Morgan Kaufmann.
- Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.
- Simon, H. A. (1983). Why should machines learn? In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Simon, H. A., & Lea, G. (1974). Problem solving and rule induction: A unified view. In L. W. Gregg (Ed.), *Knowledge and cognition*. Hillsdale, NJ: Lawrence Erlbaum.
- Smith, E. E., & Medin, D. L. (1981). *Categories and concepts*. Cambridge, MA: Harvard University Press.
- Stepp, R. E. (1979). *Learning without negative examples via variable valued logic characteristics: The uniclass inductive program Aq7-UNI* (Tech. Rep. No. UIUCDCS-R-79-982). Urbana: University of Illinois, Department of Computer Science.
- Stepp, R. E. (1984). *Conjunctive conceptual clustering: A methodology and experimentation*. Doctoral dissertation, Department of Computer Science, University of Illinois, Urbana.
- Stepp, R. E. (1987). Concepts in conceptual clustering. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 211–213). Milan, Italy: Morgan Kaufmann.
- Tukey, J. W. (1977). *Exploratory data analysis*. Reading, MA: Addison-Wesley.
- Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning*, 4, 161–186.

- Utgoff, P. E., & Brodley, C. E. (1990). An incremental method for finding multivariate splits for decision trees. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 58–65). Austin, TX: Morgan Kaufmann.
- Vere, S. A. (1978). Inductive learning of relational productions. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press.
- Vere, S. A. (1980). Multilevel counterfactuals for generalization of relational concepts and productions. *Artificial Intelligence*, 14, 139–164.
- Wasserman, K. (1985). *Unifying representation and generalization: Understanding hierarchically structured objects*. Doctoral dissertation, Department of Computer Science, Columbia University, New York.
- Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.
- Wogulis, J., & Langley, P. (1989). Improving efficiency by learning intermediate concepts. *Proceedings of the Eleventh International Conference on Artificial Intelligence* (pp. 657–662). Detroit, MI: Morgan Kaufmann.
- Wolff, J. G. (1982). Language acquisition, data compression, and generalization. *Language and Communication*, 2, 57–89.

CHAPTER 2

An Incremental Bayesian Algorithm for Categorization

JOHN R. ANDERSON

MICHAEL MATESSA

1. Introduction

A rational analysis (Anderson, 1990) is an attempt to specify a theory of some cognitive domain by specifying the goal of the domain, the statistical structure of the environment in which that goal is being achieved, and the computational constraints under which the system is operating. The predictions about the behavior of the system can be derived assuming that the system will maximize the goals it expects to achieve while minimizing expected costs, where expectation is defined with respect to the statistical structure of the environment. This approach is different from most approaches in cognitive psychology because it tries to derive a theory from assumptions about the structure of the environment rather than assumptions about the structure of the mind.

We have applied this approach to human categorization and have developed an effective algorithm for categorization. The analysis assumes that the goal of categorization is to maximize the accuracy of predictions about features of new objects. For instance, one might want to predict whether an object will be dangerous or not. This approach to categorization sees nothing special about category labels. The fact that an object might be called a *tiger* is just another feature one might want to predict about the object.

2. The Structure of the Environment

Predictions are driven by assumptions about the structure of the environment. The theory we have developed rests on the structure of natural kind categories produced by the phenomenon of *species*, which form a nearly disjoint partitioning of the natural objects because of the inability to interbreed. Within a species there is a common genetic pool, which means that individual members of the species will display particular feature values with probabilities that reflect the proportion of that phenotype in the population. Another useful feature of species structure is that the display of features within a freely interbreeding species is largely independent. Thus, there is little relationship between size and eye color in species where those two dimensions vary. In summary, the critical aspects of speciation are the disjoint partitioning of the object set and the independent probabilistic display of features within a species.

Other types of objects may display these same properties. Artifacts are another common type of object that approximate a disjoint partitioning, but there are occasional exceptions — for instance, mobile homes are both homes and vehicles. Other classes of objects (stones, geological formations, heavenly bodies, etc.) seem to approximate a disjoint partitioning, but here it is hard to know whether this is just a matter of our perceptions or whether this holds in some objective sense. One can use the understanding of speciation for natural kinds and the understanding of the intended function in manufacture in the case of artifacts to objectively assess the hypothesis of a disjoint partitioning.

2.1 Algorithms for Prediction

We have used this disjoint, probabilistic model of categories to understand the structure of the environment and to make predictions about object features. To maximize the prediction of object features, we must induce a disjoint partitioning of the object set into categories and determine the probability of features for each category. The ideal prediction function would be described by the formula

$$Pred_{ij} = \sum_x P(x|F_n) Prob_i(j|x) , \quad (1)$$

where $Pred_{ij}$ is the probability that an object will display a value j on a dimension i which is not observed for that object, the summation is

across all possible partitionings of the n objects seen into disjoint sets, $P(\mathbf{x}|F_n)$ is the probability of partitioning \mathbf{x} given the objects display observed feature structure F_n , and $Prob_i(j|\mathbf{x})$ is the probability that the object in question would display value j in dimension i if \mathbf{x} were the partition. The problem with this approach is that the number of partitions of n objects grows exponentially as the Bell exponential number (Berge, 1971). Assuming that humans cannot consider an exponentially exploding number of hypotheses, we were motivated to explore incremental algorithms such as those developed by Fisher (1987) and Lebowitz (1987).

The following steps give a formal specification of our incremental algorithm's learning and performance on each object it encounters:

1. If no previous object has been seen, initialize the category partitioning of the objects to be the empty set (i.e., no categories).
2. Given a partitioning for the first m objects, calculate for each category k the probability P_k that the $m + 1$ st object comes from category k . Let P_0 be the probability that the object comes from a completely new category.
3. Create a partitioning of the $m + 1$ objects in which the $m + 1$ st object is assigned to the category with maximum probability.
4. To predict value j on dimension i for the $n + 1$ st object calculate

$$Pred_{ij} = \sum_k P_k P(ij|k) \quad , \quad (2)$$

where P_k is the probability that the $n + 1$ st object comes from category k and $P(ij|k)$ is the probability of displaying value j on dimension i given membership in k .

The basic algorithm is one in which the category structure is grown by assigning each incoming object to its most likely category. Thus, a specific partitioning of the objects is produced. However, note that the prediction for the new $n + 1$ st object is *not* calculated by determining its most likely category and the probability of j given that category. Rather, the calculation is performed over all categories. This gives a much more accurate approximation to the ideal $Pred_{ij}$ because it handles situations in which the new object is ambiguous between multiple categories. It will weight these competing categories approximately equally.

The algorithm is not guaranteed to produce the maximally probable partitioning of the object set since it only considers partitionings that can be incrementally grown. Neither does it weight multiple possible partitionings as the ideal algorithm would. In cases of strong category structure, there will be only one probable partitioning and the incremental algorithm will uncover it. In cases of weak category structure, it will often fail to obtain the ideal partitioning, but still the predictions obtained by Equation 2 closely approximate the ideal quantity because of the weighting of multiple categories. As we will see, the correlations are about 0.95 between the predictions of our algorithm and the ideal quantities in cases of small data sets.

It remains to come up with a formula for calculating P_k and $P(ij|k)$. Since $P(ij|k)$ proves to be involved in the definition of P_k , we will focus on this latter term. In Bayesian terminology P_k is a posterior probability $P(k|F)$ that the object belongs to category k given that it has feature structure F . Bayes' formula can be used to express this in terms of a prior probability $P(k)$ of coming from category k before the feature structure is inspected and a conditional probability $P(F|k)$ of displaying the feature structure F given that it comes from category k :

$$P_k = P(k|F) = \frac{P(k)P(F|k)}{\sum_i P(i)P(F|i)} , \quad (3)$$

where the summation in the denominator is over all categories i currently in the partitioning, including the potential new one. This then focuses our analysis on the derivation of a prior probability $P(k)$ and a conditional probability $P(F|k)$.

2.2 Prior Probability

With respect to prior probabilities, the critical assumption is that there is a fixed probability c that any two objects come from the same category and that this probability does not depend on the number of objects seen so far. This is called the *coupling probability*. If one takes this assumption about the coupling probability between two objects being independent of the other objects and generalizes it, one can derive (Anderson, 1990) a simple form for the prior probability

$$P(k) = \frac{cn_k}{(1 - c) + cn} , \quad (4)$$

where c is the coupling probability, n_k is the number of objects assigned to category k so far, and n is the total number of objects seen so far. Note for large n this closely approximates n_k/n , which means that we have a strong base rate effect in these calculations with a bias to put new objects into large categories. The rational basis for this bias should be apparent.

We also need a formula for $P(0)$, which is the probability that the new object comes from an entirely new category. This is

$$P(0) = \frac{(1 - c)}{(1 - c) + cn} . \quad (5)$$

For large n this closely approximates $(1 - c)/cn$, which is again a reasonable form; i.e., the probability of a new category depends on the coupling probability and number of objects seen. The greater the coupling probability and the more objects, the less likely that the new object comes from an entirely new category.

The impact of the coupling parameter c will be to influence the number and size of categories formed. The larger the value, the fewer and larger the categories that will be produced. Since computation costs are linearly related to number of categories and not to size of categories, there might be some pressure to set c larger than its true value in the environment.

One consequence noted of Equations 4 and 5 is that there is a bias to put objects into large categories. Some have questioned the rationality of this strategy. However, we should stress that Equation 4 just sets the priors and must be combined with conditional probabilities for Equation 3. If an instance much better matches a smaller category, the conditional probabilities for the smaller category will be much higher and the instance will be assigned to that category. Thus, the bias in Equations 4 and 5 does not mean that such evidence will be ignored. However, if such feature-matching evidence is equivocal, the system will assign the instance to the larger category, which is the sensible action.

This base rate effect contributes to the order sensitivity of our algorithm. Suppose we have an instance that is ambiguous between two categories. If by chance we have seen more instances of one category before the instance, we will be biased to assign it to that category. This will make that category larger and increase our tendency to assign instances to the category. In some cases of ambiguous stimuli, this process can snowball.

2.3 Conditional Probability

We can consider the probability of displaying features on various dimensions given category membership to be independent of the probabilities on other dimensions. Then we can write

$$P(F|k) = \prod_i P(ij|k) , \quad (6)$$

where $P(ij|k)$ is the probability of displaying value j on dimension i given that one comes from category k .

This independence assumption does not prevent one from recognizing categories with correlated features. Thus, one may know that being black and retrieving sticks are features found together in labradors. This would be represented by high probabilities of the stick-retrieving and the black features in the labrador category. The independence assumption does prevent one from representing categories in which values on two dimensions are either both one way or both the opposite. For instance, it would prevent one from recognizing a single category of animals that were either large and fierce or small and gentle. However, this is not a very serious limitation. In such cases, our algorithm spawns a different category to capture each two-feature combination; it would create a category of large and fierce creatures and another category of small and gentle creatures.

The effect of Equation 6 is to focus our attention on an analysis of the individual $P(ij|k)$. Derivation of this quantity is itself an exercise in Bayesian analysis. We will treat separately discrete and continuous dimensions.

2.4 Discrete Dimensions

The basic Bayesian strategy for making inferences along a dimension is to assume a prior distribution of values along the dimension, determine the conditional probability of the data under various possible values of the priors, and then calculate a posterior distribution of possible values. The common practice is to start with a rather weak distribution of possible priors and, as more and more data accumulate, come up with a tighter and tighter posterior distribution.

In the case of a discrete dimension, the typical Bayesian analysis (Berger, 1985) assumes that the prior distribution is a Dirichlet density.

For a dimension with m values, a Dirichlet distribution is characterized by m parameters α_j . We can define $\alpha_o = \sum_j \alpha_j$. The mean probability of the j th value is $p_j = \alpha_j/\alpha_o$, and the value α_o reflects the strength of belief in these prior probabilities, p_j . The data after n observations will consist of a set of C_j counts of observations of value j on dimension i . The posterior distribution of probabilities is also a Dirichlet distribution but with parameters $\alpha_j + C_j$. This implies that the mean expected value of displaying value j in dimension i is $(\alpha_j + C_j)/\sum(\alpha_j + C_j)$. This is $P(ij|k)$ for Equation 6:

$$P(ij|k) = \frac{C_j + \alpha_j}{n_k + \alpha_0} , \quad (7)$$

where n_k is the number of objects in category k that have a value on dimension i and C_j is the number of objects in category k with the same value as the object to be classified. For large n_k this approximates C_j/n_k , which one frequently sees promoted as the rational probability. However, it must have this more complicated form to deal with problems of small samples. For instance, if one has just seen a single object in a category and it had the color red, one would not want to guess that all objects are red. If we assume there are seven colors and all the α_j were 1, the above formula would give 1/4 as the posterior probability of red and 1/8 for the other six colors unseen as yet.

2.5 Continuous Dimensions

Application of Bayesian inference schemes to continuous dimensions is more problematic but there is one approach that appears most tractable (Lee, 1989). The natural assumption is that the variable is distributed normally and the induction problem is to infer the mean and variance of that distribution. In standard Bayesian inference methodology, we must begin with some prior assumptions about the mean and variance of this distribution. It is unreasonable to suppose we can know precisely in advance either the mean or the variance. Our prior knowledge must take the form of probability densities over possible means and variances. This is basically the same idea as in the discrete case, where we had a Dirichlet distribution giving priors about probabilities of various values. The major complication is the need to state separately prior distributions for mean and variance.

The tractable suggestion for the prior distributions is that the inverse of the variance Σ^2 is distributed according to a chi-square distribution

and the mean has a normal distribution. Given these priors, the posterior distribution of values, x , on a continuous dimension i for category k , after n observations, has the following t distribution:

$$f_i(x|k) \sim t_{a_i} \left(\mu_i, \sigma_i \sqrt{1 + 1/\lambda_i} \right) . \quad (8)$$

The parameters a_i , μ_i , σ_i , and λ_i are defined as follows:

$$\lambda_i = \lambda_0 + n \quad (9)$$

$$a_i = a_0 + n \quad (10)$$

$$\mu_i = \frac{\lambda_0 \mu_0 + n \bar{x}}{\lambda_0 + n} \quad (11)$$

$$\sigma_i^2 = \frac{a_0 \sigma_0^2 + (n - 1)s^2 + \frac{\lambda_0 n}{\lambda_0 + n}(\bar{x} - \mu_0)^2}{a_0 + n} , \quad (12)$$

where \bar{x} is the mean of the n observations and s^2 is their variance. These equations basically provide us with a formula for merging the prior mean and variance, μ_0 and σ_0^2 , with the empirical mean and variance, \bar{x} and s^2 , in a manner that is weighted by our confidences in these priors, λ_0 and a_0 .

Equation 8 for the continuous case describes a probability density that serves the same role as Equation 7 for the discrete case, which describes a probability. The product of conditional probabilities in Equation 6 will then be a product of probabilities and density values. Basically, Equations 6, 7, and 8 give us a basis for judging the similarity of an object to the category's central tendency.

2.6 Conclusion

This completes our specification of the theory of categorization. Before looking at its application to various empirical phenomena, a word of caution is in order. The claim is not that the human mind performs any of the Bayesian mathematics that fills the preceding pages. Rather the claim of the rational analysis is that, whatever the mind does, its output must be optimal. The mathematical analyses of the preceding pages serve the function of allowing us, as theorists, to determine what is optimal.

A second comment is in order concerning the output of the rational analysis. It delivers a probability that an object will display a particular feature. There remains the issue of how this relates to behavior. Our basic assumption will only be that there is a monotonic relationship between these probabilities and behavioral measures such as response probability, response latency, and confidence of response. The exact mapping will depend on such things as the subject's utilities for various possible outcomes, the degree to which individual subjects share the same priors and experiences, and the computational costs of achieving various possible mappings from rational probability to behavior. These are all issues for future exploration. What is remarkable is how well we can fit the data simply assuming a monotonic relationship.

3. Application of the Algorithm

We have applied the algorithm to a number of examples to illustrate its properties. The predictions of this algorithm are potentially order sensitive in that different partitionings may be uncovered for different orderings of instances. In the presence of a strong categorical structure, the algorithm picks out the obvious categories and, as we will discuss later, there usually is little practical consequence to the different categories it extracts in the case of weak category structure. The incremental algorithm is also extremely fast. A FRANZ LISP implementation categorized the 290 items from Michalski and Chilausky's (1980) data set on soybean disease (each with 36 values) in one CPU minute on a Vax 780 or on a MAC II. This was without any special effort to optimize the code. It also diagnosed the test set of 340 soybean instances with as much accuracy as apparently did the hand-crafted diagnostic system of Michalski and Chilausky (1980).

The first experiment in Medin and Schaffer (1978) is a nice one for illustrating the detailed calculations of the algorithm. They had subjects study the following six instances, each described with binary features:

1 1 1 1 1	0 0 0 0 0
1 0 1 0 1	0 1 0 0 0
0 1 0 1 1	1 0 1 1 0

The first four binary values were choices in visual dimensions of size, shape, color, and number. The fifth dimension reflected the category label. They then presented these six objects without their category

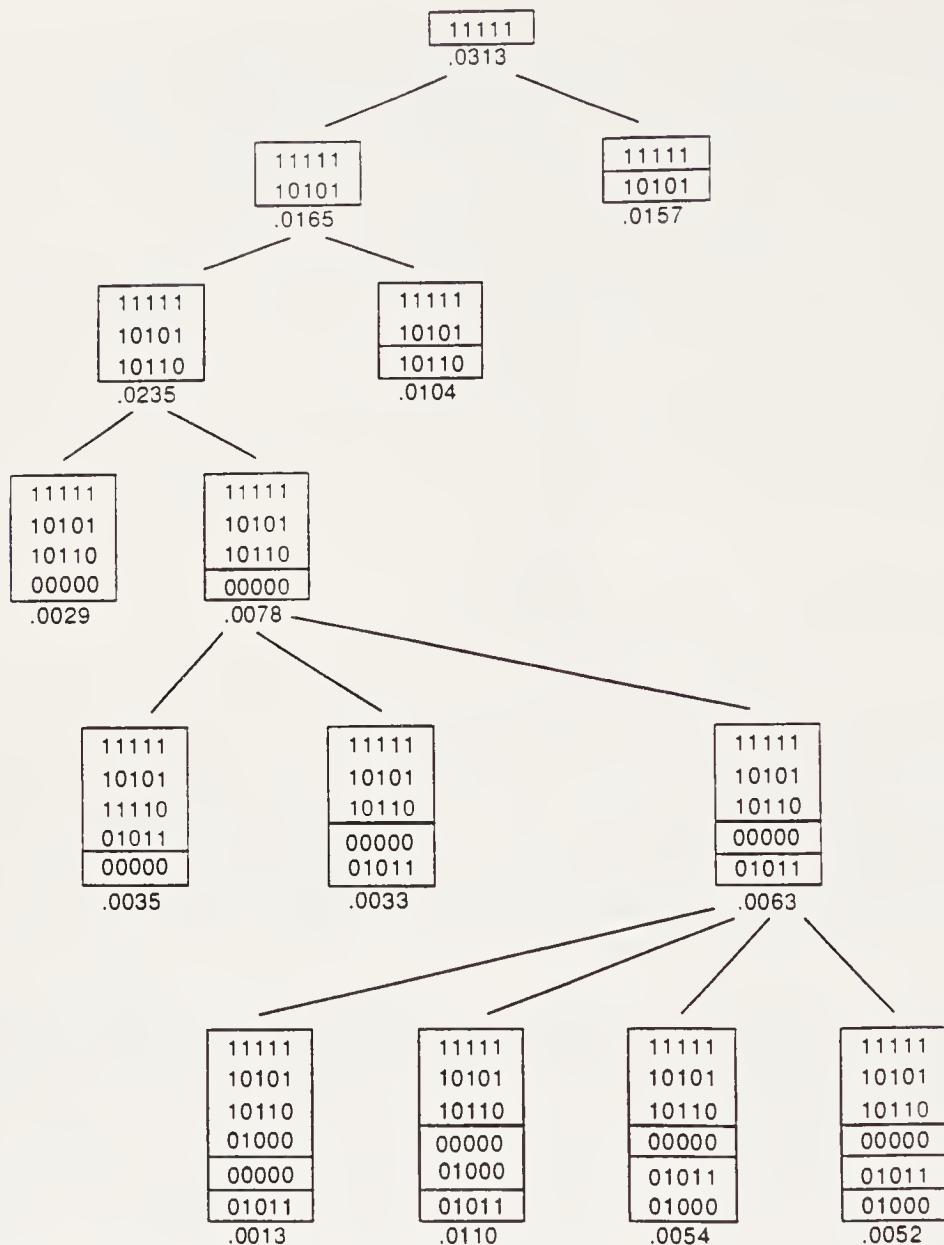


Figure 1. An illustration of the operation of the incremental algorithm in the material from the first experiment of Medin and Schaffer (1978).

label plus six new objects without a label: 0111₋, 1101₋, 1110₋, 1000₋, 0010₋, and 0001₋. Subjects were to predict the missing category label.

We derived simulations of this experiment by running the program across various random orderings of the stimuli and averaging the results. Figure 1 shows one simulation run in which we used the order 11111, 10101, 10110, 00000, 01011, 01000 and had the coupling probability

$c = 0.5$ (see Equations 4 and 5) and set all $\alpha_j = 1$ (see Equation 7). Figure 1 illustrates the search behavior of the algorithm as it considers various possible partitionings. The numbers associated with each partition are measures of how probable the new item is given the category to which it is assigned in that partition. These are the values $P(k)P(F|k)$ calculated by Equations 4 through 11. Thus, we start out with categorizing 11111 in the only possible way — that is, assigning it to its own category. The probability of this is the prior probability of a 1 on each dimension or $(0.5)^5 = 0.0313$. Then we consider the two ways to expand this to include 10101 and choose the categorization that has both objects in the same category because that is more likely. Each new object is incorporated by considering the possible extensions of the best partition so far. We end up choosing the partition {11111, 10101, 10110}, {00000, 01000}, {01011}, which has three categories. Note that the system's categorization does not respect the organization given by Medin and Schaffer.

Having come up with a particular categorization, we then tested the algorithm by presenting it with the 12 test stimuli and assessing the probabilities it would assign to the two possible values for the fifth dimension (the label). Figure 2 relates our algorithm to their data. Plotted along the abscissa are the 12 test stimuli of Medin and Schaffer in their rank order determined by subjects' confidence that the category label was a 1. The ordinate gives the algorithm's probability that the missing value was a 1. Figure 2 illustrates three functions for different ranges of the coupling probability. The best rank order correlation occurred for coupling probabilities in the range 0.2 to 0.3.

This coupling probability gave a rank order correlation of 0.87, and a coupling probability of 0.3 produced correlations of 0.98 and 0.78 for two slightly larger experimental sets used by Medin and Schaffer. These rank order correlations are as good as those obtained by Medin and Schaffer with their many-parameter model. It also does better than the ACT simulation reported in Anderson, Kline, and Beasley (1979). We have set the coupling probability c to 0.3 throughout our applications.

The reader will note that the actual probabilities of category labels estimated by the model in Figure 2 only deviate weakly above and below 0.5. This reflects the very poor category structure of these objects. Better structured material gives much higher prediction probabilities.

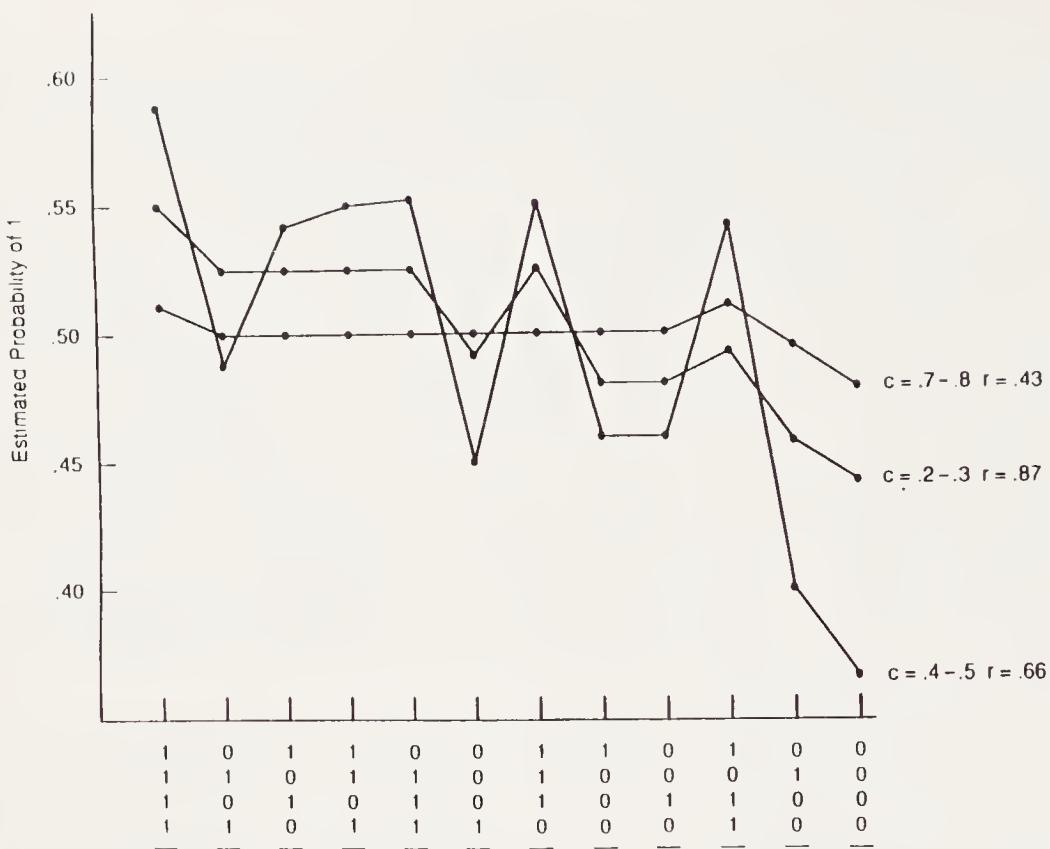


Figure 2. Estimated probability of category 1 for the 12 test stimuli in the first experiment of Medin and Schaffer (1978). Different functions are for different ranges of the coupling probability.

Detailed descriptions of the application of the algorithm to particular experiments can be found in Anderson (1990) and Anderson (in press). However, we will briefly review the applications of the algorithm to a number of empirical phenomena. The following are among the empirical phenomena we have successfully simulated:

1. *Extraction of central tendencies for continuous dimensions.* For continuous dimensions, the Bayesian model implies that categorization should vary with distance from central tendency. This enables the model to simulate the data of Posner and Keele (1968) on categorization of dot patterns and Reed (1972) on categorization of faces.
2. *Extraction of central tendencies for discrete dimensions.* The model implies that stimuli should be better categorized if they display the majority value for a dimension. This enables the model to simulate the data of Hayes-Roth and Hayes-Roth (1977).

3. *Effect of individual instances.* If an instance is sufficiently different from the central tendency for its assigned category, the model will form a distinct category for it. This enables the model to account for the data of Medin and Schaffer (1978) on discrete dimensions and Nosofsky (1988) on continuous dimensions.
4. *Linearly separable vs. non-linearly separable categories.* In contrast to some categorization models, this model is able to learn categories that cannot be separated by a plane in an n -dimensional hyperspace. This is because it can form multiple internal categories to correspond to an experimenter's category. This enables the model to account for the data of Medin (1983) on discrete dimensions and Nosofsky, Clark, and Shin (1989) on continuous dimensions.
5. *Basic-level categories.* The internal categories that the model extracts correspond to what Rosch (1976) meant by basic-level categories. Thus, it can simulate the data of Murphy and Smith (1982) and Hoffman and Ziessler (1983).¹
6. *Probability matching.* Faced with truly probabilistic categories and large samples of instances, the model will estimate probability of features that correspond exactly to the empirical proportion. Thus, it predicts Gluck and Bower's (1988) data on probability matching.
7. *Base-rate effect.* Because of Equation 4, this model predicts that usually there will be a greater tendency to assign items to categories of large size. Thus, it handles Homa and Cultice's (1984) data and reproduces the more subtle interactions of Medin and Edelson (1988).
8. *Correlated features.* As noted earlier the model can handle categories with correlated features by breaking out separate internal categories for each feature combination. Thus, it handles the data of Medin, Altom, Edelson, and Freko (1982).
9. *Effects of feedback.* If the category structure of the stimuli is strong enough, the model can extract the categories without any feedback as to category identity. In the face of weak category structure, one must provide category labels to get successful learning. Thus, this model reproduces the data of Homa and Cultice (1984).

1. For a similar application, see Gluck and Corter (1985).

10. *Effects of input order.* In the presence of weak category structure, the categories that the model forms are sensitive to presentation order. In this way it is able to simulate the data of Anderson and Matessa (see Anderson, 1990) and Elio and Anderson (1984).

All these simulations were done with a constant setting of the parameters: c from Equations 4 and 5 at 0.3, α_j from Equation 7 at 1, λ_0 from Equation 9 at 1, a_0 from Equation 10 at 1, μ_0 from Equation 11 at the mean of the stimuli, and σ_0^2 from Equation 12 at the square of 1/4 the stimulus range. All of these are plausible settings and often correspond to conventions for setting Bayesian non-informative priors.

4. Comparison to AUTOCLASS

The Bayesian character of this classification model raises the issue of its relationship to the AUTOCLASS system of Cheeseman, Kelly, Self, Stutz, Taylor, and Freeman (1988). Although it is hard to know the significance of the differences, there are a number of points of contrast.

Algorithm. Rather than an algorithm that incrementally incorporates instances into an existing category structure, Cheeseman et al. use a parameter-searching program that tries to find the best-fitting set of parameters. The Cheeseman program is quite fast and is not sensitive to the order of the examples. On the other hand, it does not easily generate predictions that can be incrementally updated with each example.

Number of classes. AUTOCLASS has a bias in favor of fewer classes, whereas this bias can be set in the rational model according to the parameter c . The system does not calculate a prior corresponding to the probabilities of various partitionings.

Conditional probabilities. It appears that AUTOCLASS uses the same Bayesian model as we do for discrete dimensions. The treatment of continuous dimensions is somewhat different, although we cannot discern its exact mathematical basis. The posterior distribution is a normal distribution, which will only be slightly different from the t distribution we use. Both AUTOCLASS and the rational model assume the various distributions are independent.

Qualitatively, the most striking difference is that AUTOCLASS derives a probability of an object belonging to a class, whereas the rational model assigns the object to a specific class. However, Cheeseman et al. report that in the case of strong category structure, the probability is very high that the object comes from a single category.

5. Order Sensitivity and Hierarchical Algorithms

The categorization algorithm that we have described is order sensitive and this has been a point of criticism of the model (Ahn & Medin, 1989). If critical examples have accidental similarities, the model will create pseudo-categories around these. If the initial examples have exaggerated differences, the algorithm will fail to identify the true categories but will split them into lower-level categories. The basic problem is that the algorithm is unable to split categories that it has formed into subcategories or to merge existing categories into larger categories, as occurs in Fisher's (1987) approach. In Anderson (1990) we showed that subjects display some sensitivity to order, but much less than our algorithm.

An interesting question concerns the consequences of this order sensitivity from the rational analysis goal of maximizing predictive accuracy, where maximal accuracy is defined with respect to the ideal algorithm (Equation 1). It is usually impossible to calculate the predictions of the ideal algorithm but the first experiment of Medin and Schaffer (1978, Figures 1 and 2) used a sufficiently small stimulus set to make such calculation possible. We calculated the ideal probabilities for the test stimuli in Figure 2 using $c = 0.5$ and $\alpha_j = 1$. At $c = 0.5$, depending on ordering, the incremental algorithm selects one of the following three partitionings:

- A: (01011) (00000, 01000) (10101, 10110, 11111)
- B: (11111, 01011) (00000, 01000) (10101, 10110)
- C: (10101, 10110, 11111) (01011, 00000, 01000)

Figure 1 illustrates the partitioning A, which has log probability -25.77 and occurs 22% of the time.² Similarly, B has log probability -26.52 and occurs 16%, whereas C has log probability -25.64 and occurs 61% of the time. The latter is the most probable

2. We are calculating the product of $P(k|F)$ (from Equation 3) for all the instances. This represents the likelihood of the data given parameters c and α .

Table 1. Correlation among various bases for predicting the stimuli in Figure 2.

	IDEAL	A	B	C
PARTITION A	0.89	×	×	×
PARTITION B	0.98	0.86	×	×
PARTITION C	0.80	0.49	0.81	×
AVERAGE	0.96	0.78	0.96	0.92

partitioning of all. By comparison, a partitioning that merges all into one category has log probability -26.50 , one that breaks them up into single categories has log probability -27.37 ,³ and something awful like (11111, 00000) (01000, 10101), and (10110, 01011) has log probability -32.07 . The median probability of the 203 partitions expressed in log terms is -28.66 , or about 5% the probability of the most probable. The Medin and Schaffer stimulus set has weak category structure and the algorithm does not always find the most probable partition. In the case of strong category structure, the program extracts the most probable interpretations independent of order of presentation. Fisher (1987) reports a similar result for his COBWEB program.

The critical issue is how well the various partitions do at predicting features. Therefore, we looked at various partitions with respect to predicting the fifth dimension of the 12-stimuli illustration in Figure 2. We looked at the correlations among the predictions of various procedures. Table 1 reproduces the correlation matrices among the predictions of the three partitionings A, B, and C, their weighted average (as produced by the incremental algorithm), and the weighted prediction from the ideal algorithm (Equation 1). As can be seen, they all are relatively highly correlated with the ideal and, except for A and C, with each other. The weighted average of A through C is very highly correlated ($r = 0.96$) with the ideal. This suggests that there is relatively little cost associated with using the incremental algorithm.

It is interesting that the predictions made by the most probable partition are not particularly good. This reflects that the most probable

3. A singleton category structure is less likely than a single category because of the high value of c . At $c = 0.3$, the log probability of the single category becomes -28.11 and the log probability of the singleton categories is -22.95 .

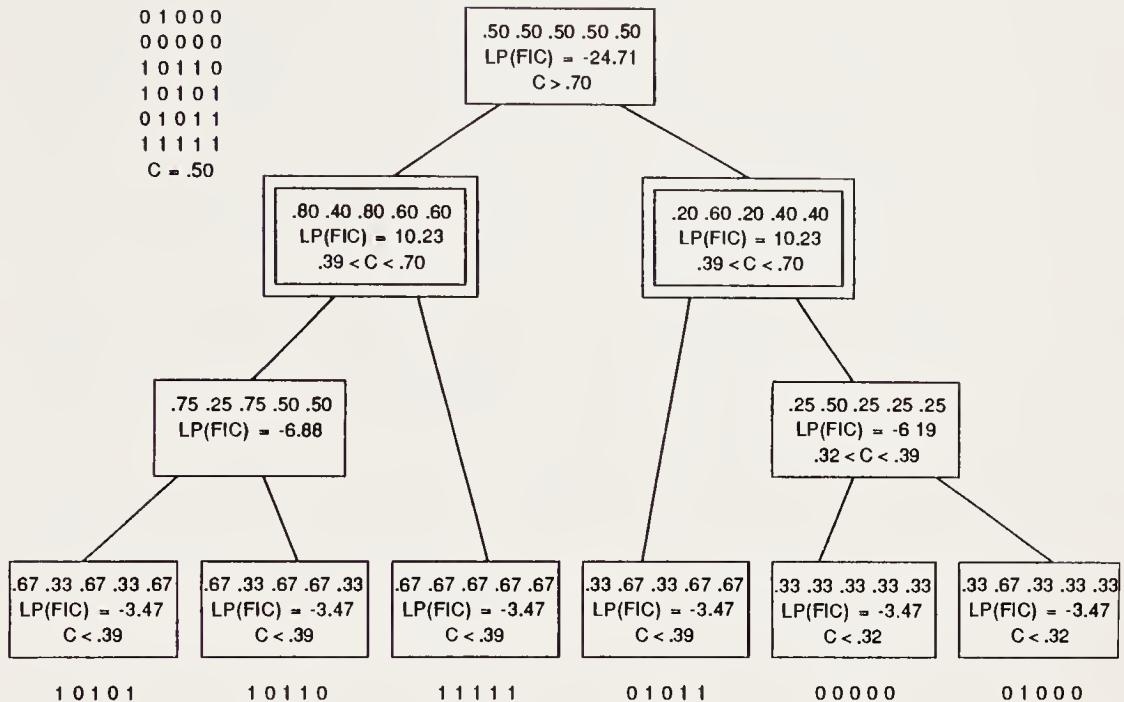


Figure 3. Hierarchical organization of stimuli from first experiment of Medin and Schaffer (1978).

partition has only 5% of the probability of the 203 possible partitions of the six stimuli. As the set size gets larger or as the category structure improves, the most probable partition will tend to dominate the prediction. This suggests that it makes sense for the system to strive for the most probable partition but that prediction from some other highly probable partition may be as good or better.

5.1 A Hierarchical Algorithm

Consideration of the order sensitivity of the algorithm has led us to consider other incremental algorithms that are less order sensitive. We were also interested in producing a hierarchical category structure and exploring the issue of whether other levels in the hierarchy, besides the basic level, might be useful for prediction.

We have developed another algorithm that is somewhat more successful at identifying the maximally probable partition but avoids considering all possible partitions as does the ideal algorithm. This algorithm

organizes the data into a hierarchical structure. Figure 3 illustrates a hierarchical organization for the stimuli from the first experiment of Medin and Schaffer. Having built such a hierarchical representation of the stimulus set, our algorithm tries to determine which partitioning within the hierarchy offers the optimal decomposition of the stimulus set. This will depend on the setting of the coupling parameter c . The higher the value of c , the larger the categories that the algorithm will tend to select. Given the structure in Figure 3, the algorithm will select the top-level node as the single category for values of c greater than 0.7. For values of c in the range 0.39 to 0.70, it will select the two subnodes. For values of c from 0.32 to 0.39, it will select the bottom-level nodes except for 00000 and 01000, which it will merge into a single category. Below 0.32 it selects singleton categories.

At any point in time, the algorithm will have a hierarchical organization for the observed instances and, given a value of c , it will have identified a set of categories. The basic algorithm for growing this network consists of the following steps:

1. As before, given a new instance, determine a category K to associate with this instance.
2. If K is an existing category, sort the new instance to a location below that category node.
3. If K is a new category, sort that category to a location below the top node for the hierarchy, with the constraint that one category cannot be placed under another.
4. Search upward from where the new item was inserted to see whether some change in the category structure is warranted. Note that this does not reorganize the hierarchy, but only changes which nodes in the hierarchy might be considered category nodes.

Figure 4 illustrates the basic logic for sorting and inserting a new instance into the hierarchical structure. We have an existing hierarchical structure consisting of a node a with subnodes b and c . We have a new instance d that we want to incorporate somewhere in the hierarchical structure under a . There are three possibilities: (i) d will be associated with the hierarchical structure dominated by b ; (ii) d will be associated with the hierarchical structure dominated by c ; or (iii) a binary branch will be created with b and c in one and d in the other. The way to choose among these is to identify the branching that will yield the maximally probable pair of categories to cover all the items under b , c , and d . For

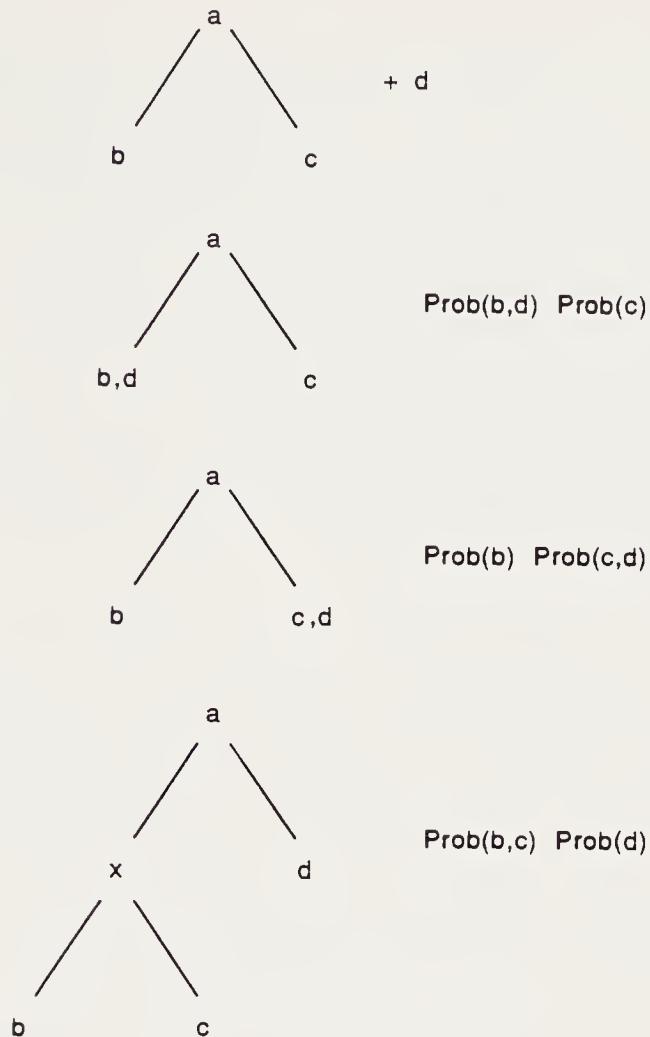


Figure 4. Choose structure to maximize probability. Should it depend on prior probability?

- (i) we consider the product of the probability of the category consisting of {b d} and the category consisting of {c}. For (ii) we consider the {b} category and the {c d} category. For (iii) we consider the {b c} category and the {d} category.

It is worthwhile comparing the performance of this hierarchical algorithm with that of the previous algorithm. We used the material of Medin and Schaffer for this purpose. There are 720 possible orderings of the six stimuli. With $c = 0.5$, the old algorithm identified three different categories and identified the optimal categorical structure 61% of the time, producing categorical structures with average log probability of -25.81 . The hierarchical algorithm identified six differ-

ent categories, but identified the optimal category structure 80% of the time, and produced categorical structures with average log probability of -25.74 . Thus, there is not much difference in average goodness, but the hierarchical algorithm is somewhat more successful at finding the optimal structure. With larger stimulus sets, we have not been able to test the hierarchical algorithm exhaustively, but it does appear more stable and does more often select the ideal structure.

We have also explored hierarchical algorithms that are not order sensitive. Basically, they used classical clustering techniques (Annenberg, 1973) to create hierarchies of stimuli sorted according to similarity where this was measured by Equations 6–8. Such algorithms are more expensive computationally because they must perform all pairwise comparisons. We have not found that they yield notably better results.

A good case for illustrating the problems of these algorithms is the iris data base of Fisher (1936). According to Fisher there are three underlying types of irises. Our algorithms, whether hierarchical or not, and whether incremental or not, fail to identify this category structure. They always identify one of the categories, Iris Setosa, but either fail to separate the other two (Iris Versicolor and Iris Virginica) or split them up inappropriately (as defined by Fisher). Cheeseman et al. claim that their algorithm is successful on these data, but we have observed that it produces an inappropriate splitting into three categories that do not correspond to the original ones. We also have observed that human subjects extract two categories or produce an inappropriate splitting into three. This notwithstanding, one can show that the original categorization produced by Fisher is more probable than the two-category solution or the various inappropriate three-category solutions. However, it is apparently impossible to find this more probable partitioning given the various approaches, artificial or human. However, prediction of features is not enhanced by the more probable partitioning. Thus, it is not clear that we should consider the behavior of these various algorithms as failures.

5.2 Genus-Level Identification

Although it does not seem that the hierarchical approach produces substantially better categorical organization, we think there may be some significance to levels of the hierarchy. There are at least two other lev-

els that are significant for purposes of prediction — the *genus*⁴ (at the higher level) and the *individual* (at the lower level). We will discuss the significance of each for prediction.

The genus level offers a level of aggregation above the species, which corresponds to a group of biologically related species that are more similar to one another than are arbitrary pairs of species. The significance of the genus level does not come in making predictions about known properties of known species. For instance, we are much better off predicting the cat-chasing propensity of Fido knowing that he is a dog than knowing that he is a mammal. The genus level's significance comes in making predictions about unknown properties of a known species (e.g., whether Fido has a spleen) and in making predictions about unknown species.

In Bayesian terms, this means that genus level can be used to set more informed priors for the species under the genus. This helps in making predictions about new species and about unexperienced properties of existing species. The interesting complication is that these priors themselves depend on estimates of the parameters for the existing species, which in turn depend on the priors. Thus, it might seem that we have a difficult joint estimation problem. The typical Bayesian approaches to such estimation problems are known as *hierarchical* methods (Berger, 1985, Section 4.6). The technical development of such methods can be quite complex and is not justified here, since we have not yet gathered data that require such complex quantitative analysis. We will simply note that, for our purposes, they provide a rationale for making estimates of the mean and variance within a species sensitive to estimates of the mean and variances for other species within a genus.

There certainly is evidence that people have this sensitivity. Even young children have expectations about the properties of new animals based on animals that they have seen (Carey, 1985). They also have expectations that certain dimensions are less variable for certain types of categories. Thus, they expect that animals within a category will have the same constitution, whereas artifacts within a category will have

4. We use the term *genus* in its more general sense to refer to a kind and not to imply the precision that is involved in the distinction among genus, family, order, class, and phylum in biology. We suspect that the level useful in prediction might be considerably above the biological genus level and actually closer to the phylum level.

the same function (Gelman, 1988). Moreover, these expectations show developmental trends to more accurate forms as experience accumulates.

The experiment of Nisbett, Krantz, Jepson, and Kunda (1983) also illustrates differential sensitivity to variance in categories of different kinds. They asked subjects to suppose they had a sample of a new mineral, a new bird, or a new tribe of people from a new island. They were given samples of different size and told that all the objects within the sample had some property. Subjects were willing to extrapolate from a single observation for some dimensions, like conductivity of the mineral or color of the tribe of people, whereas they required 20 observations before they extrapolated with any confidence for other dimensions, like the obesity of the people.

This ability to show sensitivity to variance is one thing that distinguishes this hierarchical Bayesian approach to categorization from most others. Many approaches (e.g., instance-based models) would predict that subjects would be biased in their estimate of the mean of a new species by the mean of existing species. However, these other approaches do not have the mechanisms for showing a similar sensitivity to variance.

5.3 Individual-Level Identification

The individual provides a much lower level of aggregation below the category. For purposes of prediction, there is a real advantage to identifying a repetition of an individual and making predictions from the individual rather than the category. This is because the individual may reliably deviate from the mean of the category and because many features are much more certain at the individual level than at the category level.

Retrieving an individual and making a prediction based on it corresponds to a memory retrieval. From this perspective the difference between retrieval and categorization concerns whether prediction is being made at the individual level or the category level. It is basically the same logic of prediction; however, it must be parameterized differently:

1. To reflect the fact that individuals repeat themselves much less often than categories, we use a lower value of c .
2. We need to capture the fact that the features are much less likely to change. This requires lower values of the α_i for discrete dimensions and much lower values of σ_0^2 for continuous ones.

There has been much speculation as to how categorization behavior relates to memory behavior. The instance-based models (Medin & Schaffer, 1978; Nosofsky, 1986) would argue that everything is really memory-based, whereas connectionist models (McClelland, Rumelhart, & Hinton, 1986) would argue that there are no separate representations of instances and everything is merged. Both frameworks try to account for differences between categorization and memory by arguing that a single representation is processed in different ways. The current model offers a representation that distinguishes the two levels but uses the same Bayesian logic at both levels. Of course, the rational representation is only an acknowledgement of the fact that there are individuals and categories in the real world. It does not really make any claims about how they are processed in the head. Anderson (in press) presents experimental evidence that people make different predictions when they operate at the category level from those they make at the individual level.

6. Summary

In summary, we have identified an incremental Bayesian algorithm that is fast, yields near-optimal predictions about stimulus dimensions, and corresponds with uncanny accuracy to the behavior to humans. We have explored the potential of some hierarchical variations of the algorithm, but these produce marginal improvements at best in the prediction behavior. However, there is reason to suppose that human categorization behavior has sensitivities to levels both above and below the basic level category.

Our interest in the original incremental algorithm began as an attempt to approximate the ideal, computationally impossible, prediction specified by Equation 1. It was not intended as a serious model for human cognition or as an artificial intelligence application. However, after more than two years of exploration, we have failed to find a real improvement and continue to be surprised at its success.

Acknowledgements

The research reported in this chapter was supported by Grant N00014-90-1489 from the Cognitive Science Program, Office of Naval Research.

References

- Ahn, W., & Medin, D. L. (1989). A two-stage categorization model of family resemblance sorting. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 315-322). Ann Arbor, MI: Lawrence Erlbaum.
- Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Lawrence Erlbaum.
- Anderson, J. R. (in press). The adaptive nature of human categorization. *Psychological Review*.
- Anderson, J. R., Kline, P. J., & Beasley, C. M. (1979). A general learning theory and its applications to schema abstraction. In G. H. Bower (Ed.), *The psychology of learning and motivation* (Vol. 13). New York: Academic Press.
- Annenberg, M. R. (1973). *Cluster analysis for applications*. New York: Academic Press.
- Berge, C. (1971). *Principles of combinatorics*. New York: Academic Press.
- Berger, J. O. (1985). *Statistical decision theory and Bayesian analyses*. New York: Springer-Verlag.
- Carey, S. (1985). *Conceptual change in childhood*. Cambridge, MA: MIT Press.
- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). A Bayesian classification system. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54-64). Ann Arbor, MI: Morgan Kaufmann.
- Elio, R., & Anderson, J. R. (1984). The effects of information order and learning mode on schema abstraction. *Memory and Cognition*, 12, 20-30.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139-172.
- Fisher, R. A. (1936). Multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, 179-188.
- Gelman, S. A. (1988). The development of induction within natural kind and artifact categories. *Cognitive Psychology*, 20, 65-95.

- Gluck, M. A., & Bower, G. H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General*, 8, 37–50.
- Gluck, M. A., & Corter, J. E. (1985). Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum.
- Hayes-Roth, B., & Hayes-Roth, F. (1977). Concept learning and the recognition and classification of exemplars. *Journal of Verbal Learning and Verbal Behavior*, 16, 321–338.
- Hoffman, J., & Ziessler, C. (1983). Objektidentifikation in künstlichen begriffshierarchien. *Zeitschrift für Psychologie*, 194, 135–167.
- Homa, D., & Cultice, J. (1984). Role of feedback, category size, and stimulus distortion in the acquisition and utilization of ill-defined categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 10, 83–94.
- Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2, 103–138.
- Lee, P. M. (1989). *Bayesian statistics*. New York: Oxford.
- McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The appeal of parallel distributed processing. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing*. Cambridge, MA: MIT Press.
- Medin, D. L. (1983). Structural principles of categorization. In B. Shepp & T. Tighe (Eds.), *Interaction: Perception, development, and cognition*. Hillsdale, NJ: Lawrence Erlbaum.
- Medin, D. L., Altom, M. W., Edelson, S. M., & Freko, D. (1982). Correlated symptoms and simulated medical classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 8, 37–50.
- Medin, D. L., & Edelson, S. M. (1988). Problem structure and the use of base-rate information from experience. *Journal of Experimental Psychology: General*, 117, 68–85.
- Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review*, 85, 207–238.

- Michalski, R. S., & Chilausky, R. L. (1980). 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. *International Journal of Policy Analysis and Information Systems*, 4, 125-161.
- Murphy, G. L., & Smith, E. E. (1982). Basic level superiority in picture categorization. *Journal of Verbal Learning and Verbal Behavior*, 21, 1-20.
- Nisbett, R. E., Krantz, D. H., Jepson, D., & Kunda, Z. (1983). The use of statistical heuristics in everyday inductive reasoning. *Psychological Review*, 90, 339-363.
- Nosofsky, R. M. (1986). Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, 115, 39-57.
- Nosofsky, R. M. (1988). Similarity, frequency, and category representation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11, 54-65.
- Nosofsky, R. M., Clark, S. E., & Shin, H. J. (1989). Rules and exemplars in categorization, identification, and recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15, 282-304.
- Posner, M. I., & Keele, S. W. (1968). On the genesis of abstract ideas. *Journal of Experimental Psychology*, 77, 353-363.
- Reed, S. K. (1972). Pattern recognition and categorization. *Cognitive Psychology*, 3, 382-407.
- Rosch, E., Mervis, C. B., Gray, W., Johnson, D., & Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, 7, 573-605.

CHAPTER 3

Representational Specificity and Concept Learning

JOEL D. MARTIN
DORRIT BILLMAN

1. Introduction

When a young child is fed by an adult, she might at first assume that all generally similar creatures, i.e., all adults, would treat her equally well. With more experience, she might come to expect food from a smaller group of adults, such as all familiar adults, and then from a yet more specific class of creatures, such as mother, father, and grandmother (MFG). On the other hand, the child could initially expect feeding from a very specific set, mother only, and with experience generalize to female relatives (MG) and then to larger classes, such as MFG.

Clearly, both approaches, starting with general classes or starting with specific classes, can produce appropriate results. Both converge toward an optimal level of generality. For this child, MFG might be the most appropriate class for several properties such as feeds. Even though the two approaches attain the same result, one might be more efficient. For instance, if one approach generally required fewer examples before converging on the correct class, we would use that method in machine learning algorithms and would expect to find it used in natural systems.

One may suppose, though, that neither approach should consistently outperform the other. Rather, depending on the learning situation, sometimes one and sometimes the other would learn most quickly or most cheaply. For example, when regularities in the input occur be-

tween small numbers of properties so that the predictive structure is rule-like, a system that begins by considering general classes (i.e., a general-to-specific method) might learn faster or require fewer resources during learning. On the other hand, a system that begins with specific classes (i.e., a specific-to-general method) might learn relatively slowly. When the regularities occur among many features so that the predictive structure is more idiosyncratic, the specific-to-general system should learn more quickly.

This chapter reports our development and evaluation of these hypotheses about search direction for learning. Surprisingly, we find a different pattern of results than expected. Before we present those results, we first define general and specific representational units, as well as general-to-specific and specific-to-general learning methods. We then outline specific algorithms that embody these methods and report the behavior of the algorithms in systematically varied domains. Finally, we consider the implications of our results for the interaction between domain and method.

2. General and Specific Representational Units

Traditionally, general *representational units*¹ that cover many instances have been contrasted with specific representational units that cover few instances. As in our simple example, a young child may expect the property of feeds from a very general class, adults, or from a specific class, mother. With experience, the child may specialize representational units that are too general, or may generalize units that are too specific.

For many years, both psychologists and computer scientists have employed this distinction (e.g., Medin & Schaffer, 1978; Mitchell, 1982). Some researchers believe that learning methods which rely on specific representational units have the advantage because they preserve much valuable, context-specific information. In contrast, others argue for learning methods that rely on general representational units because they yield better applicability and are less sensitive to errors.

1. A representational unit is a cohesive component of a system's representation of its world. For example, an instance is a representational unit in a case-based approach, and a rule is a representational unit in a rule-based approach.

Psychologists have traditionally tried to determine how humans form generalizations. Recently, though, several researchers have proposed that learning methods which acquire only instances provide equally good or better models of human behavior (e.g., Medin & Schaffer, 1978) than do learning methods which form explicit generalizations. However, it is rarely clear when a representational unit is general or specific. For instance, a representational unit can combine several instances but still not lose any information about idiosyncrasies, yielding a unit that is general in one sense but that has specific content (Barsalou, 1989). Even when instances and rules can be distinguished, the corresponding learning methods can produce equivalent results. The performance algorithm that operates on learned knowledge can simply carry out the additional work necessary to dynamically extract rules from cases or to reconstruct cases from appropriately stored rules.

The same issues arise in the field of machine learning, in which some argue for the use of specific cases over general rules, because cases have important contextual richness that rules lack. However, in practice, cases and rules are difficult to distinguish. Some ‘cases’ are abstracted and become far more general than some rules. Also, ‘rules’ are often used to represent individual instances, particularly exceptions. Furthermore, even when the specificity of representational units can be easily distinguished, performance algorithms can carry out the additional processing to produce equivalent behavior. Put simply, the specificity of representational units is ill defined and seems to be a poor determiner of learning behavior.

Moreover, many learning systems represent knowledge at different levels of specificity at different stages, making this a continuum rather than a dichotomy. Nevertheless, these systems typically move in a single direction as they gain experience, working either from specific to general or from general to specific. If one holds the performance element constant, the effects of such a control strategy on behavior depends on one’s definition of specificity. Thus, to study these effects, we first extract some precise definitions from the intuitive meanings of specificity.

2.1 Types of Specificity

Three contrasting senses of specificity can be identified, each of which provides a different formalization of the notion that a specific representational unit is somehow ‘closer’ to the input instances.

2.1.1 AMOUNT OF EVIDENCE

Our first sense defines specificity as the amount of evidence that supports a representational unit, that is, the number of observed instances that have contributed to the unit's description. This measure increases even if the same instance is observed multiple times. By this definition, a representational unit is specific if it combines information from one or very few observed instances, and is general if it combines information from many observed instances. For example, a category of tailless cats may combine information from only one or two observed instances and therefore be quite specific. Alternatively, a category of house cats may combine many observed instances and hence be quite general. Several existing systems adopt one or the other extreme on this dimension. Trace and instance models in cognitive psychology (e.g., Estes, 1986; Hintzman, 1986) assume that each representational unit combines one or few instances. The same is true for low-level or leaf nodes in decision tree methods such as ID3 (Quinlan, 1986) and COBWEB (Fisher, 1987). In contrast, prototype models (e.g., Neumann, 1974) integrate across many instances, as do high-level nodes in ID3 and COBWEB.

In general, this definition of specificity may seem most natural for characterizing probabilistic rather than logical representational units, because the former is more concerned with amount of evidence. However, the distinction is still useful for logical units. Suppose a system forms logical representational units by intersecting the attribute values of multiple instances. Early in learning, the system may not want to intersect many instances for fear of overgeneralizing, but as it becomes more confident, it intersects more and more instances. Hence it begins with low-evidence representational units and moves toward higher-evidence ones. Analogous learning methods exist that move in the opposite direction.

2.1.2 NUMBER OF ATTRIBUTES

Our second sense defines specificity as the number of attributes specified in the representational unit. If an instance of a particular disease has ten attributes, a maximally specific representational unit would maintain and use information about all ten, while a general representational unit would include information about only one or a few attributes. Different numbers of attributes are appropriate for different

situations. For example, if most furry animals give live birth, that information can be stored in a representational unit that specifies only the attributes body-covering and birth-method. On the other hand, the learner sometimes needs to use several attributes. If there is an exception to the previous relationship involving furry animals with webbed feet and a fleshy bill that lay eggs, one must keep information about body-covering, birth-method, foot-type, and mouth-type.

Specificity as amount of evidence (definition 1) is conceptually independent from specificity as number of attributes (definition 2). A representational unit that combines information about many instances can maintain either many or few attributes, and the same is true for units that combine few instances.

Many methods maintain information about all attributes, presumably because the additional contextual information is expected to lead to better performance. Classic examples from psychology include most prototype models (e.g., Neumann, 1974) and the Medin and Schaffer model (1978). From machine learning, Fisher's COBWEB (1987) maintains and uses information about all attributes in all representational units.² At the other extreme, there are also several methods whose representational units maintain information about only a few attributes. Bruner, Goodnow, and Austin (1956) suggested such methods in their classic work on concept attainment. In addition, most early work in machine learning of concepts proposed methods that formed highly general representational units with values of only one or a few attributes. More recently, Schlimmer and Granger (1987) proposed a method that begins with highly general representational units and that can specialize or generalize the units as necessary.

This definition of specificity is most useful for logical rather than probabilistic representational units, because removing attributes is the only way the former can reduce the impact of some attributes on performance. However, this distinction is still useful for probabilistic units when the learner decides to generalize and not represent some of the possible variation. For example, all hamsters have fur, all are small, and many live outside, although some live inside. A learner may decide that where a hamster lives is not sufficiently related to its being a hamster and may drop that variation from the representation, possibly

2. Although the category utility function at the heart of COBWEB always combines information about all attributes, not all are equally relevant during performance.

allowing some other representational unit, such as one about pets, to make predictions about where a particular hamster lives.

2.1.3 EXTENSIONS OF THE UNIT

These two definitions are related to another common interpretation of the general versus specific dichotomy. Under this third interpretation, a general representational unit ‘matches’ many distinct instances chosen from the theoretical space of all possible instances (see Mitchell, 1982, for an example of this definition of specificity). By ‘matches’, we mean that the representational unit is somehow activated by an instance. On the other hand, specific representational units match relatively few distinct instances. For example, in the set of possible instances, there may be millions of distinguishable objects that bark, but by logical necessity only a proper subset of these possible objects bark and are black. A representational unit that matches all members of the larger set of barking creatures is more general, by this definition, than one that only matches the members of the smaller set.

This third definition refers to the extension — the instances in the domain. In contrast, the first two definitions focus on what the learner represents. Because we wanted to separate characteristics of the learning method from characteristics of the domain, we chose to use only the first two definitions to guide algorithm development.

2.2 Distinct Definitions

Although related, our three definitions of specificity do differ. They are related both because all three are variants of the intuitive meaning and because in some situations all three definitions agree as to which units are specific. For instance, if we choose a representational unit that matches a large number of instances (general by definition 3), that representational unit would likely summarize a great deal of evidence (general by definition 1). Moreover, with a large extension, the system can maintain fewer attributes (general by definition 2), because larger classes, like mammals, often have fewer distinguishing attributes than do smaller classes like platypuses.

Nevertheless, in some circumstances, the three definitions diverge. To begin with, a representational unit that matches many instances would not be based on much evidence if the members of that class were rarely

encountered. For example, there may be more distinct insects than there are humans in the space of possible instances. However, as humans, we are more likely to encounter and notice other humans. Therefore, we would collect more evidence about the smaller of the two classes simply because we more frequently encounter members of that class. In other words, a representational unit that matches many instances (general by definition 3) does not always produce a corresponding representational unit based on a lot of evidence (general by definition 1).

Furthermore, a representational unit that maintains few attributes does not necessarily match more possible instances than does a unit with many attributes. In particular, these definitions of specificity will differ when the matcher permits partial or best matches. In this case, a representational unit can match an instance even when some attribute values in the instance contradict the representational unit. For example, suppose we have three specific representational units (many attributes), one each for the categories of *poodle*, *siamese*, and *finch*. Of these, the *siamese* category would be the best match for all cat instances, because it shares more attribute values with them than do the other two descriptions. Moreover, when we encounter our first lion, our matcher would again choose the *siamese* category as the best match, even though the lion is bigger and meaner than we expect. Alternatively, we might only have general representational units (few attributes) for the more general categories of *dog*, *cat*, and *lion*. Of these, the *cat* category will be the best match for all cat instances, but will not be the best match for lions. The more specific category, *siamese*, matches more instances than the more general one, *cat*, because the former does not have any strong competitors. Therefore, a representational unit like *cat*, with few attributes (general by definition 2), will not always match more instances (general by definition 3) than will a unit like *siamese* with many attributes.

Finally, a representational unit with few attributes does not necessarily have more evidence supporting it, and vice versa. Consider a system that builds a hierarchy of categories. Concepts higher in the tree, such as *animal*, summarize more evidence than those lower in the tree, such as *dog*. Unfortunately, this tells us nothing about how many attributes must be stored with each category. For instance, suppose we have a hierarchy of mammals and fish that organizes dogs, cats, sharks, and guppies. Some set of features determines whether a creature is a mammal or a fish, and some other set of features determines whether a

particular mammal is a cat or a dog. Neither set of features is necessarily larger than the other. As a simple example, we may know about the features *birth-method*, *body-covering*, and *produced-sound*. The first two attributes distinguish between the high-evidence categories fish and mammal, and only the last one distinguishes between cat and dog. Therefore, a representational unit that is general by definition 1 (amount of evidence) is not always general by definition 2 (number of attributes) and vice versa. It is important to note that although the above examples distinguished the different definitions for logical rather than probabilistic representational units, similar arguments apply for both types of formalism.

2.3 Changing the Specificity of Representational Units

During learning, some representational units are overly general and some are overly specific. The learner must exploit its experience to move toward optimally general representations. In this chapter we are addressing two simple approaches: general-to-specific learning and specific-to-general learning.

The first approach begins with general representational units by one of the two definitions and specializes the general units as necessary. According to the ‘amount of evidence’ definition, a learning method can specialize by splitting high-evidence units into multiple low-evidence units. Each of the resulting specialized units would then contain a proper subset of the evidence available to the original representational unit. According to the ‘number of attributes’ definition, a learning method can specialize by adding attributes to a representational unit. In contrast, the specific-to-general approach begins with specific representational units and generalizes those units as necessary. A system can achieve this either by combining low-evidence units (generalizing by definition 1) or by reducing the number of attributes (generalizing by definition 2).

By varying the direction of learning, generalization versus specialization, and by varying the definition of specificity, we generated four different learning methods. Table 1 organizes these four possibilities: amount of evidence plus specific to general; amount of evidence plus general to specific; number of attributes plus specific to general; and number of attributes plus general to specific. We expect differences between the specific-to-general and general-to-specific methods. For exam-

Table 1. The four learning methods.

	Amount of evidence Instance partition	Number of attributes Pattern composition
Specific to general	<p>Early predictions use single instances</p> <p>As necessary, use units with many instances</p>	<p>Early predictions use units with many attributes</p> <p>As necessary, use units with few attributes</p>
General to specific	<p>Early predictions use units with many instances</p> <p>As necessary, use units with fewer instances</p>	<p>Early predictions use units with few attributes</p> <p>As necessary, use units with many attributes</p>

ple, we expect the former to be superior for domains with predominately idiosyncratic relations and the latter to be superior for more abstract relations. However, there is reason to suspect that specific-to-general methods will be generally superior (Fisher & Chan, 1990). Furthermore, because the two definitions of specificity stress different aspects of learning, we expect differences between the methods that control amount of evidence and those that control number of attributes.

3. The Learning Methods

The four methods we generated are all incremental, unsupervised learners that accept instances as lists of attribute values but that, as mentioned, vary in their representational bias. They share many subroutines and they have a common foundation. We begin with an overview of all methods, and then follow with a detailed discussion of each in turn.

3.1 Overview of the Methods

As shown in Table 1, two of the four methods allow explicit variation of the amount of evidence in a representational unit. These two methods are based on a general technique that we call *instance partitioning*. This learning technique recursively partitions the input set into disjoint

categories of similar instances. The other two learning methods in Table 1 allow explicit variation of the number of attributes in a representational unit, based on a technique that we call *pattern composition*. This learning method maintains multiple partitions of simplified categories called *patterns* and combines the information from several patterns.

3.1.1 INSTANCE PARTITION METHODS

Under the first definition of specificity, a representational unit is specific if it has relatively little evidence to support it. The amount of evidence contained in an individual representational unit can be explicitly controlled by organizing representational units as nodes in a tree. In this tree, lower-level nodes have relatively little evidence and higher-level nodes combine the evidence of their children. For example, the category *mammal* would combine the evidence — the number of supporting instances — from *dog*, *cat*, *horse*, etc. A general-to-specific method initially uses higher-level nodes to make prediction decisions and gradually move down the tree. A specific-to-general method begins near the leaves of the tree and gradually moves up to higher-level nodes.³

An *instance partition* (IP) method builds a tree of categories by recursively partitioning instances into disjoint sets, and classifies an instance by sorting along a single path in the tree (Figure 1). For example, a basset hound would be classified as a mammal as opposed to a fish, then further classified as a dog as opposed to a cat, and finally classified as a basset hound. Instance partition methods have shown considerable promise (Fisher, 1987; Bobick, 1987; Anderson & Matessa, this volume) both in machine learning and in modeling human categorization. Borrowing from that success, we used two IP algorithms that are similar to Fisher's (1987) COBWEB and Bobick's categorization algorithm (1987).

3.1.2 PATTERN COMPOSITION METHODS

The second definition of specificity states that a representational unit is more specific if it maintains and uses a greater number of attributes. This quantity can be controlled by storing and using multiple patterns

3. None of the methods described in this chapter increases the generality or specificity of the system as a whole. Rather, they vary the specificity of the representational units that are actually used during performance. For the current study, we chose to ignore the possible confound of relearning information.

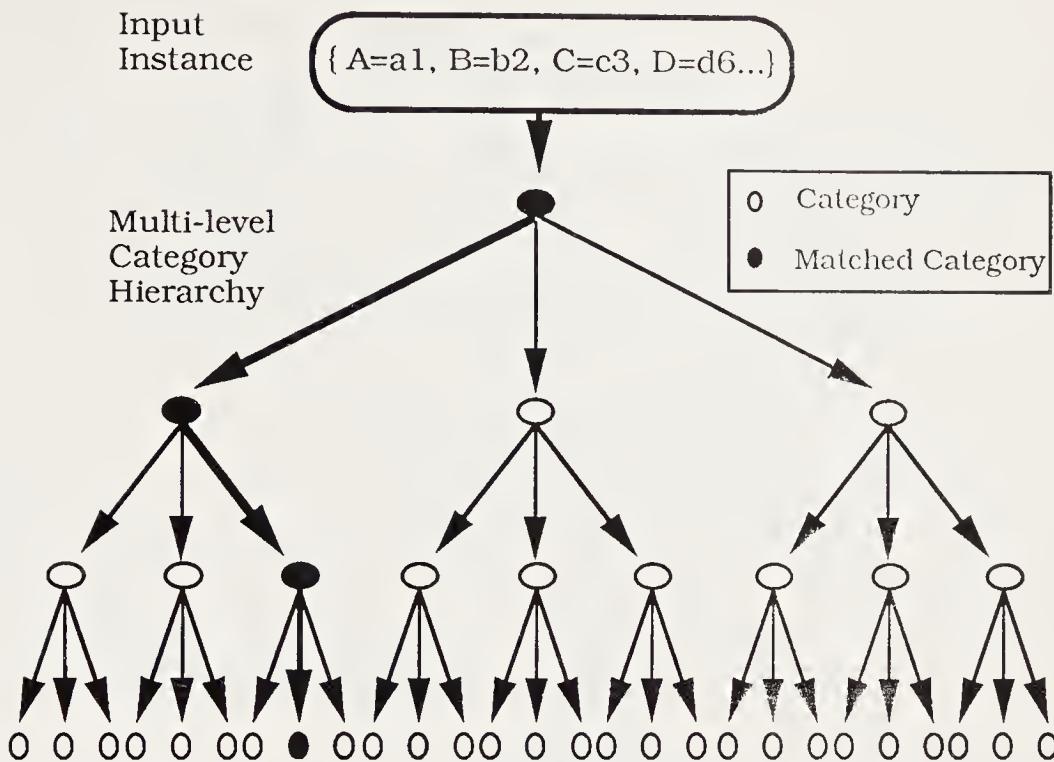


Figure 1. The IP processing architecture, which classifies an instance into exactly one category at each level of the category hierarchy.

that vary the number of attributes used during prediction. A simple type of pattern is a collection of attribute values that tend to cooccur, such as `color = white`, `fat-ratio = high`, and `habitat = cold`. This pattern specifies three attributes, `color`, `fat-ratio`, and `habitat`, a more general pattern might specify only two, and a more specific pattern might specify four or more. During prediction, if known attribute values of a new instance match one part of a pattern, then the learner could predict that the instance would also have the rest of the pattern's attribute values. Under this view, a specific-to-general method begins by matching patterns with many attributes and gradually reduces the number of attributes through learning. A general-to-specific method begins with detailed matches and shifts toward more general cues.

A *pattern composition* (PC) method constructs two levels of structure. First, the patterns are organized into several disjoint sets, and then activated patterns are combined by the second level (Figure 2). For instance, at the pattern level, the pattern `color = white`, `fat-ratio = high`, and `habitat = cold` would contrast with `color = bright`,

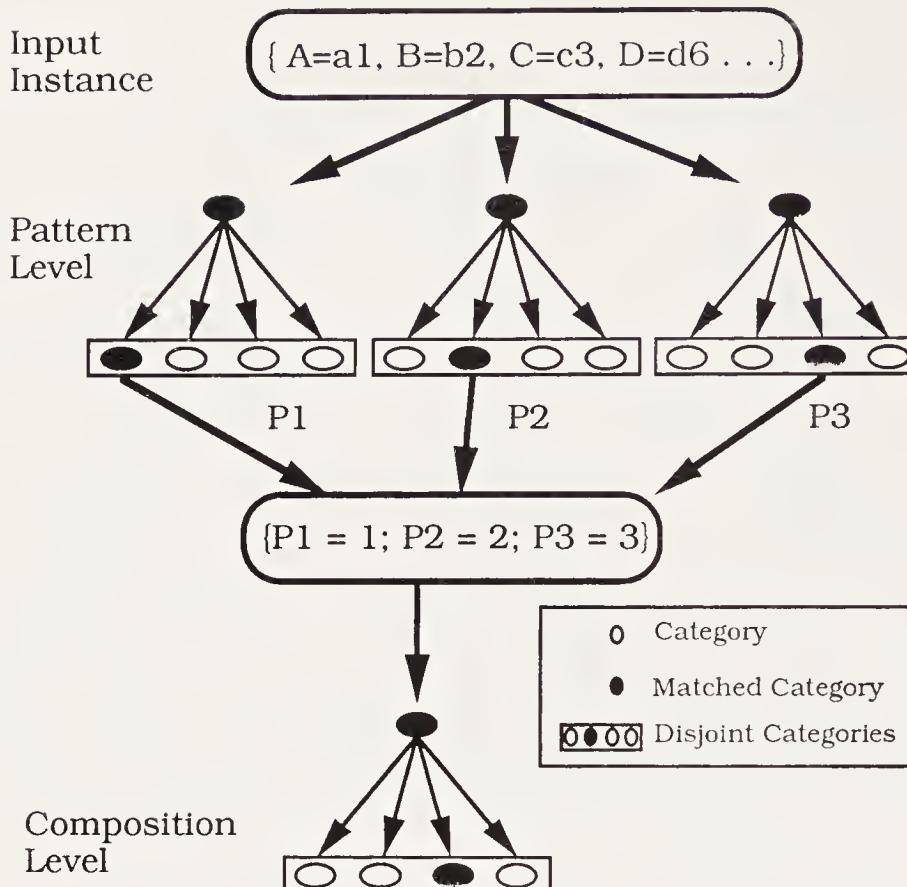


Figure 2. The PC processing architecture, which classifies an instance to one pattern from each disjoint set at the pattern level, and then combines the matched patterns by choosing one composition pattern.

`fat-ratio = low`, and `habitat = temperate`, as well as other possibilities.⁴ Significantly, PC methods create several of these disjoint sets of patterns, each set being roughly orthogonal and concentrating on different attributes. For instance, one set of patterns might link particular adaptations (`fat`) to different habitats (`far-north`) and another orthogonal set might relate aspects of an animal's relationship to humans (`pet`, `pack-animal`, etc.).

After some patterns have been activated, predictions about attribute values are made by combining the predictions of those patterns. Making predictions about attribute values for reindeer might combine a pattern describing animals living in the `far-north` and another orthogonal pattern describing `pack-animals`.

4. In general, PC methods could recursively divide a pattern into mutually exclusive subpatterns, but this study omitted the capability so that only the IP methods could explicitly vary the amount of evidence of representational units.

3.1.3 COMPARISON OF IP AND PC METHODS

PC methods share several characteristics with the IP methods. For example, one component of both systems is an algorithm for learning a set of disjoint representational units. Not surprisingly, the major difference between the methods is directly related to the difference between the two definitions of specificity.

IP methods organize their mutually exclusive sets of categories into a tree to allow explicit variation in the amount of evidence. Categories near the root have a large amount of evidence relative to those lower in the tree. However, IP methods do not provide a natural way for varying the number of attributes used for prediction. Because they learn multiple contrasting sets of variably-sized patterns, PC methods allow explicit variation in the number of attributes.⁵ However, the PC methods used in this study do not permit a natural method for varying the amount of evidence in representational units used for prediction.

3.2 Detailed Description of Methods

The two IP and the two PC methods all share an underlying technique for learning sets of disjoint representational units. We describe this common technique in some detail before sketching the learning methods.

3.2.1 LEARNING DISJOINT REPRESENTATIONAL UNITS

All four learning methods incorporate a component that learns a single set of categories or patterns. Below, we refer to them simply as *categories*. This component attempts to learn the set of disjoint categories by maximizing the estimated probability of making a correct prediction, $P(\text{correct})$. During learning, this component either adds an instance to one of the existing categories or creates a new category. It decides whether to add a new category or use an existing one by estimating which would most improve the probability of making correct predictions. This is the fundamental notion behind several effective IP methods.

5. We might have restricted the PC methods to learn only one contrasting set of patterns. Typically, however, good performance with small patterns requires combining information from multiple patterns, specifically those that specify values for different attributes.

In the rest of this section, we develop a heuristic estimator of the probability of making correct predictions, $P(\text{correct})$. First, we separate the probability of correct classification overall into a sum of the probabilities of correct classification for particular attribute values. Then from this, we produce a more practical formula to estimate $P(\text{correct})$.

Since the categories are disjoint, we can separate the $P(\text{correct})$ and describe the probability as a weighted sum across the set of categories:

$$P(\text{correct}) = \sum_k P(C_k)P(\text{correct} | C_k) . \quad (1)$$

Similarly, if we assume that each prediction trial requests the value of exactly one of the attributes, A_j , we can describe the probability of being correct on one prediction trial as a weighted sum across the set of attributes:

$$P(\text{correct}) = \sum_k P(C_k) \sum_j P(a_{A_j} | C_k)P(\text{correct}_{A_j} | C_k) . \quad (2)$$

In this equation, $P(a_{A_j} | C_k)$ is the *query probability* for A_j , the probability of requesting the value for A_j .

Finally, because the values for each attribute are disjoint, we can further divide the probability of being correct into the probabilities of being correct for each value:

$$P(\text{correct}) = \sum_k P(C_k) \sum_j P(a_{A_j} | C_k) \sum_i P(A_j = V_{ij} | C_k)^2 . \quad (3)$$

In this equation, the quantity $\sum_i P(A_j = V_{ij} | C_k)^2$ is the probability of correctly guessing some value of an attribute, A_j , given that we know the best category for the instance. The probability of guessing correctly depends on two independent events: first, something must be true in the world, and second, we must guess the world is that way. If we know that an instance best fits a category, C_k , then the probability that the attribute A_j has the value V_{ij} is the quantity $P(A_j = V_{ij} | C_k)$. For instance, given that a car does not crank, the probability that the car's malfunction is a dead battery, $P(\text{malfunction} = \text{dead battery} | \text{does not crank})$, might be 0.8. Suppose we choose to guess that something is true with the same probability that we expect it to be true. Then the probability that both $A_j = V_{ij}$ is true and that we guess it to be true (two independent events) is the square of the probability, $P(A_j = V_{ij} | C_k)$, as above.

Despite the relation to the probability of being correct, Equation 3 can be extremely inaccurate in practical use. In these equations, the probability of classifying an instance to a particular category, $P(C_k)$, was assumed to be known. Actually, $P(C_k)$ must be estimated from the proportion of the instances that have already been classified to the category. This estimate is fairly accurate when the probability of querying attributes is uniform across attributes, but it can be led astray otherwise. To rectify this problem, we divide the probability of classifying an instance into two parts: $P(I \in C_k)$, the probability that an arbitrary instance best fits⁶ in $P(C_k)$, and $P(I \rightarrow C_k)$, the probability of classifying a partial instance to $P(C_k)$.

These values can be estimated during learning. The first value is the same estimate as $P(C_k)$, the proportion of instances that have already been classified to the category. The second value, the probability of classifying the instance to a given category, depends on two choices. First, some algorithm must be chosen that ranks the strength of the match between an instance and a category. Classification uses this algorithm to determine which category to use during prediction. Second, an assumption must be made about which attributes are likely to be available during prediction for the matching algorithm to use. Given these two assumptions, the probability that an instance will be classified to a category can be estimated by the proportion of known instances that are classified to that category.

Introducing these new quantities into Equation 3, we generate a practical formula:

$$\sum_k \sum_l P(I \rightarrow C_k \wedge I \in C_k) \sum_j \sum_i P(a_{A_j} | C_k) P(A_j = V_{ij} | I \in C_k)^2 . \quad (4)$$

This formula produces a different and more accurate estimate of the probability of classifying an arbitrary instance because the probability $P(I \rightarrow C_k)$ is no longer just the proportion of instances classified to the category. Instead, it is sensitive to the match function that is used to classify an instance.

Other approaches have used similar equations. Equation 3 is roughly the same as criteria used by Gluck and Corter (1985) and Fisher (1987). In their ‘category utility’ measure, however, they assume that the query

6. A category is the best fit if adding the instance to it most improves the probability of being correct.

probability is uniform across attributes and categories. This yields the following measure of category utility:

$$CU = \sum_k P(C_k) \sum_j \frac{1}{n_A} \sum_i P(A_j = V_{ij} | C_k)^2 . \quad (5)$$

Furthermore, both Gluck and Corter (1985) and Fisher (1987) omit the reciprocal of the number of attributes, $\frac{1}{n_A}$, and hence use a metric that is proportional to the probability of being correct. The assumption of uniform query probabilities is sufficient for many IP methods. However, for a more general view of IP methods and for the PC methods, there are two reasons we cannot make this assumption.

First, Equation 5 does not acknowledge that some attributes can be more important to predict than others, such as ‘type of malfunction’ versus ‘car color’. If both of these attributes are specified in examples, Equation 5 prefers a partitioning that achieves good predictions for both attributes. However, the unimportant attributes can reduce accuracy when predicting important attributes.⁷ On the other hand, Equation 4 lets attributes vary in importance and does not have this accuracy problem. Some may be very important and have a probability of being queried close to one, whereas some may never be needed and have query probabilities of zero.

Second, in the PC methods, we need to distinguish between attributes that are provided only for classification purposes and those that need only be predicted. Equation 4 lets us do this, making it more versatile than Equation 5. Hence, Equation 4 is used by the IP and PC methods outlined below.

The division of C_k into $P(I \epsilon C_k)$ and $P(I \rightarrow C_k)$ in Equation 4 actually highlights the reason Equation 5 is inadequate. When using Equation 5 with non-uniform query probabilities, some instances are not classified to the category in which they best fit, because the best fit and classification are generally based upon different attributes. This motivated us to separately consider when an instance best fits a category and when it is actually classified to that category.

7. Pazzani, Ali, and Silverstein (personal communication) have independently developed a variant of Fisher’s COBWEB system that they call ‘Goal-oriented COBWEB’. It also addresses the problems associated with treating all attributes equally.

Table 2. A general algorithm for instance partition learning.

```

Function IP(Instance, Node)

If Node is a leaf,
  • Create a new leaf node using Instance.
  • Create a new node whose children are Node and the newly
    created leaf node.

Otherwise, for subcategories of the current node,
  • For each subcategory, temporarily add Instance to it,
    estimating predictive accuracy.
  • Temporarily create a new category, estimating its predictive
    accuracy.
  • Permanently add Instance to best-scoring subcategory, Best.
  • Call IP(Instance, Best).

```

3.2.2 INSTANCE PARTITION METHODS

In the IP methods, the optimality criterion from Equation 4 is used to determine where to add new instances and when to create new categories. As summarized in Table 2, when a new instance is encountered, an IP method temporarily adds the instance to each established category one at a time. The value of the quantity in Equation 4 is obtained for each temporary addition, and the category associated with the largest improvement is chosen as the best established category for the new instance. In addition, the system tries creating a new category with the new instance as its only member. If the quantity in Equation 4 is now greater than the maximum for established categories, then this new category, with its single member, is made permanent. On the other hand, if it is still highest for an established category, the system adds the instance to that category and updates all quantities needed to estimate Equation 4. This same process continues recursively down the tree of categories adding the instance to successive subcategories. It finishes when it adds a new leaf category based on only the new instance.

During prediction, an instance with some but not all values specified is recursively classified through the tree until it reaches a leaf node category. Once at the leaf, the value of an unknown attribute in the novel instance (e.g., *color*) is predicted from information stored at the

category (e.g., *polar bear*) by selecting the most likely value ($P(\text{color} = \text{white} | \text{polar bear}) > P(\text{color} = \text{brown} | \text{polar bear})$). No modifications to the tree are made in the process of generating a prediction.

This method, here called simply *IP*, was modified to produce both a specific-to-general version and a general-to-specific version. *IP* recursively creates a tree of categories. When the recursion that creates those categories unwinds, we can update counts to indicate whether a particular category or its subcategories generate better predictions (see Fisher, 1989). After learning, each category stores an additional two frequencies for each attribute: the frequency of its own correct guesses and the frequency of correct guessing from lower-level categories.

In the specific-to-general version, if a particular category produces more correct predictions about a particular attribute than all of its children, then that category is used to generate predictions for that attribute. Otherwise, classification continues to lower nodes in the tree. We also produced a general-to-specific version of *IP*. In this variant, predictions are initially generated using the root node and come to be generated using more specific nodes only if the immediate children consistently produce more accurate predictions than the parent. Thus, the two algorithms construct identical trees, and their behavior differs only in the level at which they make predictions.

3.2.3 PATTERN COMPOSITION METHODS

PC methods use Equation 4 to determine how to organize patterns and how to combine activated patterns. When a new instance is encountered, the PC methods temporarily add the instance to each possible pattern in every disjoint set, one at a time, as summarized in Table 3. The value of Equation 4 is obtained for each addition,⁸ and the pattern with the largest associated improvement in Equation 4 is chosen as the best established pattern for the new instance. Moreover, the system temporarily creates a new pattern in each disjoint set of patterns, one set at a time. If any one of these new patterns has a higher score for Equation 5 than does the best of the established patterns, then the best new pattern with its single member is made permanent.

8. To ensure that different sets of patterns capture distinct regularities, Equation 4 is applied to only those attributes not correctly predicted by already activated patterns. This is done by temporarily reducing the query probabilities $P(a_A)$ of correctly predicted attributes to zero so they are not important to predict.

Table 3. A general algorithm for pattern composition learning.

```

Function PC(Instance, Patterns, Composition-Categories)

Let Activated-Patterns be an empty set.
Repeat
  • For each Pattern, temporarily add Instance to it, estimating
    predictive accuracy.
  • For each set of disjoint Patterns, create a new Pattern,
    estimating predictive accuracy.
  • Permanently add Instance to the best-scoring pattern, Best.
  • Add Best to Activated-Patterns.
  • Remove Best and its disjoint siblings from Patterns.
Until Patterns is empty.
  • Translate Activated-Patterns into attribute values, where
    the Pattern is the value and its disjoint set is the
    attribute.
  • For each Composition-Category, temporarily add the
    translated Instance to it, estimating predictive accuracy.
  • Temporarily create a new Composition-Category, estimating
    its predictive accuracy.
  • Permanently add the translated Instance to the best
    Composition-Category.

```

Because the PC methods allow multiple patterns to match the instance, they must repeat and seek additional patterns. When each pattern is activated, the pattern and its disjoint siblings are removed from further consideration. This whole process completes when there are no more patterns.

Once several patterns have been activated, the composition level accepts these patterns in attribute-value form, with the disjoint set being the attribute, and the particular pattern being the value of that attribute. The pattern level essentially translates the instance into a different set of attributes, and the composition level learns higher-order patterns that organize the first-level patterns. Again, Equation 4 determines where to add the incoming instance, which in this case has been translated. As before, the instance is temporarily added to each existing composition pattern in turn and a new pattern is temporarily

created. The pattern, either an established one or the new one, that is associated with the greatest improvement in Equation 4 is chosen as the permanent destination for the translated instance.

Because composition addresses translated instances, Equation 4 is used slightly differently here from its use in IP methods, in which all attributes are assumed equally important to predict and equally likely to be omitted during prediction. In the composition level, however, only the initial attributes (those in the actual instances) are important to predict, and only the translated attributes (those from the pattern level) are available for classification during prediction.⁹ We therefore assigned large uniform query probabilities to each of the initial attributes, and we assigned query probabilities of zero to each of the translated attributes.

We created two variants of this pattern-composition method (which we call *PC*), a general-to-specific version and a specific-to-general version. To do this, a parameter was introduced that changes the number of attributes that must match in order to activate a particular pattern. Each pattern in the system maintains its own version of this parameter. For the specific-to-general method, the initial setting of the parameter is the number of attributes in the instances. As the system runs, it keeps statistics at each category to estimate the success of PC with both the current parameter setting and the setting made less specific by ignoring one more attribute. For the general-to-specific version, the initial parameter setting is two. The system also keeps statistics to estimate the success of PC with the current parameter setting and the setting that is one attribute more specific. In both algorithms, the parameter of an individual pattern changes when another setting yields better performance. As before, the two methods form the same knowledge structures but use them in different ways.

4. Empirical Interaction of Method and Domain

We expected methods with different biases to perform differently across domains that had different mixes of idiosyncratic and general relations. Methods that initially use more specific representational units may be superior when idiosyncratic relations predominate. Alternatively, an initial bias toward general representational units may be superior when

9. This is the reason we used Equation 4 instead of Equation 5, which only permits uniform query probabilities.

abstract relations prevail. These possibilities are assessed in Experiment 1. We expected to find that each bias would be useful, but under different circumstances.

Experiment 2 compared methods in their ability to transfer learning between domains. If the same general predictive relationships occur in two domains, a method that isolates general relationships and that uses general representational units would most easily transfer learning from one domain to the other. A system that uses specific units would suffer, because it has not isolated the pieces that can be transferred. For example, if some virus generates symptoms in all mammals but idiosyncratic symptoms in cats, then a method using specific representational units that learns about cats would not have ready access to the general set of symptoms when considering dogs. A method that allows only specific representational units (in terms of the number of attributes) can overspecialize, and therefore may not benefit from past learning in other domains.

4.1 Experiment 1: Degree of Idiosyncrasy

We created two artificial domains, one in which idiosyncratic relations predominated and another in which abstract relations predominated. Each instance in the domains was created using six attributes, each having three possible values.

In the first of these domains, four of the six attributes were highly interdependent. When asked to predict the value of any one of these attributes, a system must know the values of the remaining three attributes. For example, to predict a fruit's ripeness, one must know its color, texture, and shape; no single attribute is enough. Some green fruits are ripe; some are not. Sometimes a soft texture indicates that a fruit is ripe; sometimes it is overripe. In this type of domain, a system must learn and maintain several exceptions or special cases.

In the second domain, the value of every attribute is strongly predicted by the value of exactly one other attribute. When asked to predict the value of an attribute, a system must know only the value of that other attribute. For instance, the outer surface of a fruit is fairly predictive of its inner texture. Fruits with waxy dimpled surfaces have citrus textures, whereas fruits with thin smooth skins, like pears or apples, have crunchy inner textures.

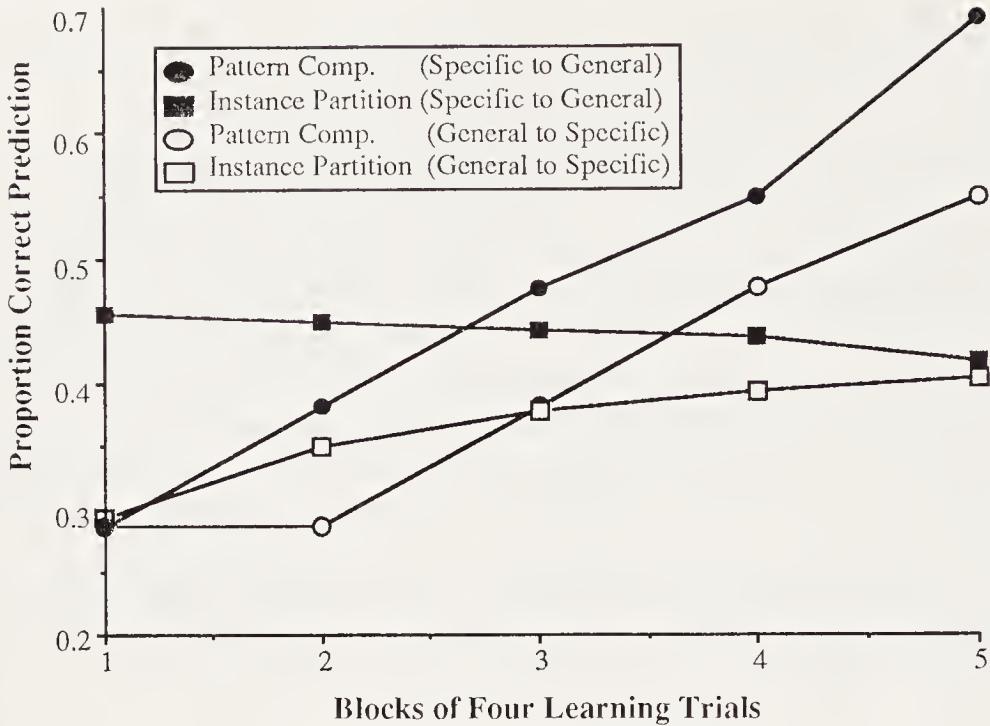


Figure 3. Proportion of correct predictions for all four methods in the abstract domain.

Procedure. We applied all four methods to both domains. The performance measure was predictive accuracy on unseen attributes, averaged over ten runs. For each run, an initial set of 27 instances was randomly divided into a training set of 20 instances and a test set of seven instances. The 20 training instances were presented in five blocks of four instances each. After each block, learning was turned off and predictive accuracy was averaged across all test instances. The test for a single instance consisted of successively dropping each attribute value and asking the system to predict the missing value. Predictive accuracy for the instance was the average across all attributes.

Results. All four methods performed equally poorly, and roughly at chance, in the domain with highly idiosyncratic regularities, but they did demonstrate learning in the abstract domain. Figure 3 shows the proportion of correct predictions for all four methods in the domain with abstract regularities. It highlights the early superiority of the specific-to-general IP strategy and the continued superiority of the specific-to-general PC method.

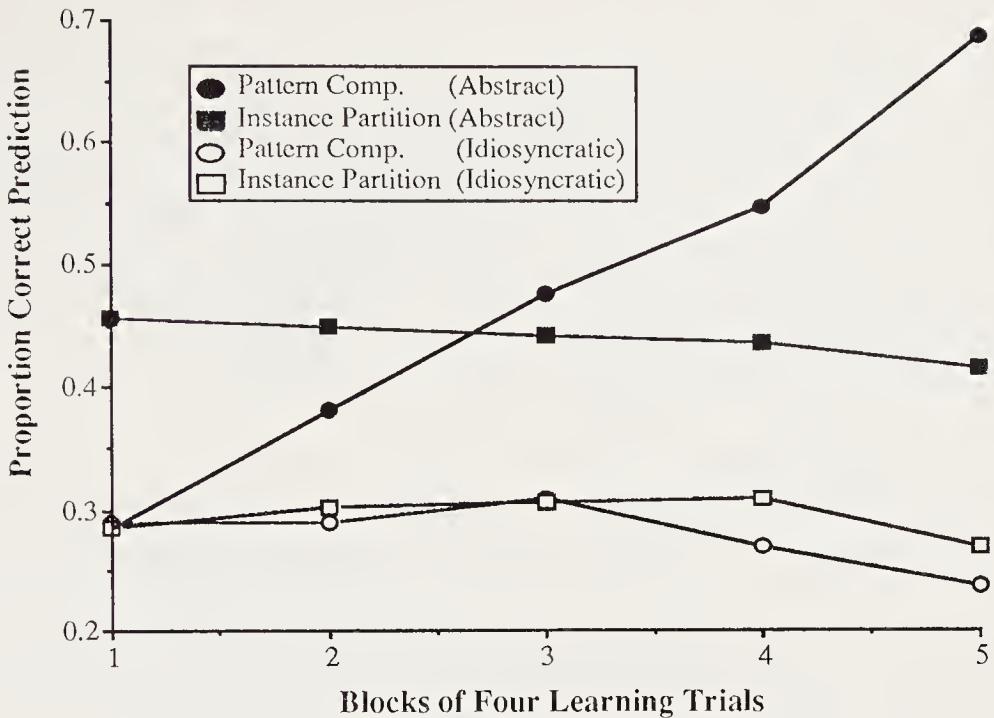


Figure 4. Proportion of correct predictions for the specific-to-general methods for both domains.

Figure 4 presents the proportion of correct predictions for the specific-to-general methods in each domain. PC methods were generally more beneficial than IP methods (after eight to twelve trials) for less idiosyncratic domains, but were equivalent to IP methods for highly idiosyncratic domains. The benefits were achieved because PC methods expect and appropriately handle multiple general relationships.

Recall that the various methods have different biases about the order in which one should consider hypotheses. The results of this experiment demonstrate that these performance biases produce different learning behaviors.

4.2 Experiment 2: Transfer Across Domains

For our second experiment, we created two artificial domains that shared some predictive relationships and differed in others. Each instance in the domains was created using nine attributes, each having three values. Both domains had multiple pairwise predictive relationships between A and B, B and C, B and D, etc. Thus, *each* value of A was highly predictive of a corresponding value in B, one in C, and one in D. The

difference between the two domains was that different particular values corresponded. For example, in one text editor, the first three function buttons might control deleting, inserting, and moving, in that order. However, a second text editor might assign the buttons in a different order. We conducted this experiment to determine the differential ability of the four methods to transfer expectations yielding positive effects for shared relations and negative effects for altered relations. Langley (1987), Schlimmer and Granger (1986), and Scott and Markovitch (this volume) report similar studies of cross-domain transfer.

Procedure. All four methods were applied first to one domain and then to the other. In contrast to Experiment 1, the previously learned representational units were not removed when the second domain was presented. This allowed the learning in the first domain to influence learning in the second. Again, the performance measure was predictive accuracy on unseen attributes, averaged over ten runs. For each run, we randomly divided each domain's initial set of 27 instances into a training set of 20 instances and a test set of seven instances.

The training set for the first domain was presented in five blocks of four instances each. After each block, learning was turned off and predictive accuracy was averaged across all test instances from this domain. As before, a test of a single instance consisted of successively dropping each attribute value and asking the system to predict the missing value. Predictive accuracy for the instance was the average across all attributes. After the five training blocks for the first domain were completed, the training set for the second domain was presented in five blocks of four instances each. Again, after each block, learning was turned off and prediction accuracy was averaged across all test instances from this new domain. For all performance tests, we examined attributes separately for positive transfer and for changing predictive relationships (negative transfer).

Results. This experiment revealed only minor differences between the specific-to-general and general-to-specific methods. Thus, in our discussion, we combine the two IP methods and combine the two PC methods to highlight differences between these basic approaches. The major result, as shown in Figure 5, is that PC methods discriminate between old and new regularities, whereas IP methods do not. That is, the IP methods show an almost equal drop in predictive accuracy for both pos-

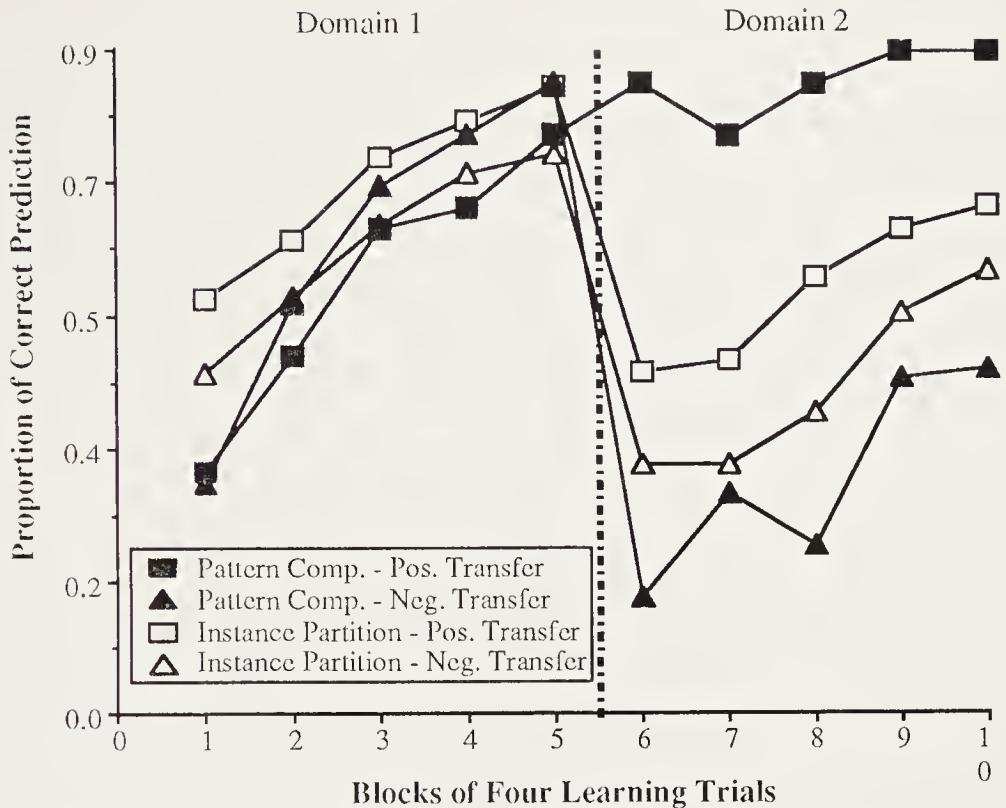


Figure 5. Proportion of correct predictions for IP and PC methods when the domain changes.

itive and negative transfer conditions. The PC methods, on the other hand, are able to separately extract and maintain general rules. Those rules that remain effective after a domain change positively affect performance, and those that change negatively affect performance. For some domain changes, PC methods provide a great benefit, whereas for others, the negative transfer swamps any potential value. For this domain, the PC techniques' benefit from positive transfer outweighs cost from negative transfer, though this may be tempered by floor effects for the negative transfer relations. As in the first experiment, the PC methods initially learn more slowly than the IP methods, suggesting that the benefits of generality require additional processing.

These results, like those for Experiment 1, demonstrate that the methods based on different specificity biases produce different results in different domains. In addition, this study showed that transfer can affect different biases in different ways.

5. Discussion

The experimental results demonstrate that variants of the two classes of methods — IP and PC — with different learning biases have different behaviors. Contrary to our expectations, however, there were only modest differences between the specific-to-general and general-to-specific methods for both definitions of specificity. Furthermore, the main effect was an apparent superiority of the specific-to-general methods in domains that have general, abstract regularities, the opposite pattern from what we expected.

5.1 The Specific-to-General Advantage

There are several possible explanations for this observation. Perhaps the specific-to-general methods for both definitions of specificity are inherently superior. The general-to-specific methods may commit themselves too early, when not enough information is known. For specificity as ‘amount of evidence’, this would mean that learning is too sensitive to early spurious correlations and insufficiently sensitive to some reliable, but as yet rare, relationships. For specificity defined as ‘number of attributes’, early overcommitment implies that the choice of attributes is sometimes made when some irrelevant attributes still look attractive. Thus, the advantage of the specific-to-general methods may lie in their more informed commitment to particular generalizations. When regularities are idiosyncratic, they maintain the necessary context, and when regularities are more abstract, they assume that any or all of the attributes could contribute. They might get the best of both worlds; this is essentially the conclusion of Fisher and Chan (1990).

Another explanation of the specific-to-general advantage is that some of the assumptions we treated as incidental may be critical in determining relative performance. For example, the algorithms differed in the specificity of the information used at performance time instead of differing in the specificity of the information accumulated at learning time. The methods maintained both highly specific and highly general representational units and varied the portion of that information that was used during performance. This was done to facilitate the movement from specific representations to general ones or from general to specific ones without requiring the systems to relearn some information that might have been forgotten. If we instead allowed some information to

be removed, that would likely affect the results. However, we expect this particular change would actually favor the specific-to-general methods because the general-to-specific approach initially loses information.

For our implementations, we also chose to use patterns or categories instead of conditional rules as the format for representational units. By ‘pattern’ we mean a unit specifying values of multiple attributes that can be activated by matching on many different combinations of the attributes. In contrast, a conditional rule is a unit with a condition part that specifies attribute values for activating the unit and an action part that specifies the attribute values to be predicted. We do not know how this assumption affected the results. For example, we do not know whether early commitment in the general-to-specific case would be detrimental for asymmetric representations such as rules.

5.2 The Pattern Composition Advantage

The PC methods showed a pronounced advantage over the IP methods in both experiments. We hypothesize that this advantage results from the ability of the PC methods to learn separate sets of categories for separate sets of covarying attributes. For example, if *habitat* and *color* covary and so do *food-type* and *tooth-type*, then the PC method would learn two sets of contrasting patterns. On the other hand, the IP methods would be limited to a single set of contrasting patterns unless they were changed into PC methods by allowing overlapping categories. Interestingly, the advantage of the PC methods is associated with slower learning when compared to IP approaches.

In addition to the above results, our studies make several broader contributions. First, the IP and newly proposed PC methods are significant in their own right, as is the distinction between them. Second, because the four methods contrasted in limited, controlled ways, we can partially understand the basis for their relative successes and failures and can generalize the results. Third, the studies illustrate the merits of the general research approach.

5.3 The IP/PC Distinction

The IP methods and the novel PC methods were presented to allow explicit variation of two definitions of specificity. Both draw on the predictive success criterion specified in Equation 4. Because this crite-

tion is quite general, it allows the methods to be applied in a variety of learning conditions, such as when some attributes are more important for prediction or when some attributes only appear as feedback. At one limit, all attributes are important for matching, producing COBWEB-like performance. At the other extreme, only one or a few attributes are important for matching, producing behavior more similar to that of nodes in a decision tree. By varying the attributes that are important for matching and those that are important for prediction, Equation 5 can handle both supervised and unsupervised learning, as well as many variations in between.

In addition, the PC method also provides a model for learning overlapping category structure. Cognitive psychologists have also traditionally focused on hierarchical structures of disjoint categories (Rosch et al., 1981; Anderson & Matessa, this volume) and have suggested that humans may be biased toward such structures. However, intelligent systems are not and should not be restricted to these hierarchies. It is common to have multiple sets of categories to describe natural and artificial kinds. For example, a particular horse might be a female, a pet, a mammal, and a beast of burden. Further, conceptual combination (Medin & Shoben, 1988; Osherson & Smith, 1982) demonstrates that people certainly can use overlapping category representations. Although most of the work in knowledge representation in AI has assumed multiple classification or multiple inheritance, most category learning work has focused on single classification structures. Below we highlight a few notable examples of systems that learn overlapping categories.

Recently, Shastri (1988) has suggested efficient connectionist representations that incorporate multiple inheritance and that can be used to explore learning. In fact, the PC approach incorporates many of the representational conventions of Shastri (1988), such as using overlapping sets of competing values and representing concepts as probabilistic distributions across the competing values.

Lebowitz's (1987) UNIMEM was one of the first methods that took advantage of the potential of multiple classification for increased predictive accuracy. It built a tree of categories but allowed an instance to belong to any number of the categories in the tree. Thus, it permitted overlapping concepts, such as ivy-league-university and urban-university. UNIMEM differs from PC in that it does not have a specified method for carrying out multiple classification at performance time.

More recently, Quinlan (1987) has modified the ID3 learning algorithm to handle multiple classification. The method converts tree structures to rules, thereby making the relationships independent of their original placement in the tree. The main differences between PC and Quinlan's approach is that the latter produces rule-like representational units instead of categories, because it is concerned with supervised learning rather than unsupervised learning.

There are also two notable connectionist approaches to multiple classification. Schlimmer and Granger (1986) presented one method, called STAGGER, for rule-like learning using representational units that contain conjunctions or disjunctions of values. The system then combines the predictions from multiple units in a weighted manner.

Multi-layer parallel distributed processing models (Hinton, Rumelhart, & Williams, 1986) have concept-like hidden nodes that classify instances in multiple ways. They have methods for combining the predictions from appropriate concepts, and they allow considerable flexibility in the degree of supervision for the learning task. In contrast to the PC methods, they do not generally permit explicit control of the information used from a representational unit. Further, it is difficult to interpret the meaning of an individual representational unit.

5.4 Basis for Generalizing the Results

The two classes of methods, IP and PC, behaved differently in the experiments. Typically, though, when methods are compared they differ in a host of ways, making appropriate allocation of credit difficult. However, the methods used here were very similar, and the differences in behavior probably resulted from the small number of systematically varied differences. Furthermore, most of the core assumptions that remained constant across the methods, such as the utility score, were unlikely to interact differentially with the varied aspects, such as the direction of search, which may have been optimal for one method but damaging to another. This gives us some basis for generalizing results across learning methods. However, our ability to generalize falls short of the ideal, because we had to introduce more differences than we wanted to test. As is often the case, one change to a program necessitated additional changes in order to maintain reasonable behavior.

Nevertheless, our attempts at generality have forced us to recognize additional forms of variation that were initially only implicit in our

methods. For example, we were at first unaware of how we might have colored our results by varying specificity at the time of use (prediction) rather than varying it during learning. As we identify more such variables, we will achieve greater and greater control of important variations, and as this control increases, so will our ability to generalize from experimental findings.

Of course, the eventual goal of this type of research is the determination of the characteristics of learning algorithms that are best for given types of domains. However, because so little is yet known, we must be satisfied with the discovery of some characteristics that interact with domain type. We have chosen to explore a traditional distinction between algorithms and have used that distinction both as a source of variation in itself and as a means for discovering other significant variables.

Acknowledgements

This research was supported in part by a Stelson Grant from the Georgia Institute of Technology to the first author and Grant No. 7R23HD20522 from the National Institutes of Health to the second author. The authors would like to thank Doug Fisher and Pat Langley for many valuable comments on earlier drafts.

References

- Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Lawrence Erlbaum.
- Barsalou, L. W. (1989). On the indistinguishability of exemplar memory and abstraction in category representation. In T. K. Srull & R. S. Wyer (Eds.), *Advances in social cognition* (Vol. 3). Hillsdale, NJ: Lawrence Erlbaum.
- Bobick, A. F. (1987). *Natural object classification*. Doctoral dissertation, Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA.
- Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking*. New York: John Wiley.
- Estes, W. K. (1986). Array models for category learning. *Cognitive Psychology*, 18, 500-549.

- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139–172.
- Fisher, D. H. (1989). Noise-tolerant conceptual clustering. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 794–799). Detroit, MI: Morgan Kaufmann.
- Fisher, D. H., & Chan, P. K. (1990). Statistical guidance in symbolic learning. *Annals of Mathematics and Artificial Intelligence*, 2, 135–148.
- Gluck, M., & Corter, J. (1985). Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum.
- Hintzman, D. L. (1986). “Schema abstraction” in a multiple-trace memory model. *Psychological Review*, 93, 411–428.
- Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.
- Lebowitz, M. (1987). Experiments with incremental concept formulation: UNIMEM. *Machine Learning*, 2, 103–138.
- Medin, D. L., & Schaffer, M. M. (1978). A context theory of classification learning. *Psychological Review*, 85, 207–238.
- Medin, D. L., & Shoben, E. J. (1988). Context and structure in conceptual combination. *Cognitive Psychology*, 20, 158–190.
- Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18, 203–226.
- Neumann, P. G. (1974). An attribute frequency model for the abstraction of prototypes. *Memory and Cognition*, 2, 241–248.
- Osherson, D. N., & Smith, E. E. (1982). Gradedness and conceptual combination. *Cognition*, 12, 299–318.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.
- Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27, 221–234.
- Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M., & Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, 8, 382–439.

- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing* (Vol. 1). Cambridge, MA: MIT Press.
- Shastri, L. (1988). A connectionist approach to knowledge representation and limited inference. *Cognitive Science*, 12, 331–392.
- Schlittner, J. C., & Granger, R. H. (1986). Incremental learning from noisy data. *Machine Learning*, 1, 317–354.

CHAPTER 4

Discrimination Net Models of Concept Formation

HOWARD B. RICHMAN

1. Introduction

In recent years, some computational models of concept formation (Gennari, Langley, & Fisher, 1989) have adopted characteristics of discrimination network models such as EPAM (Feigenbaum, 1959), the first computerized model of human perception and memory. As a result, there has been a growing interest in using discrimination nets in models of concept formation.

This chapter examines some of the earliest discrimination net systems, which have traditionally been forwarded as models of recognition and rote learning. The next section describes two particular systems that use discrimination nets for concept learning, one that is supervised and nonincremental, and another that is unsupervised and incremental, thus satisfying Fisher and Pazzani's (this volume) criteria for a model of concept formation. Section 3 describes how this latter model, EPAM, accounts for a variety of learning results in perception, as well as other psychological phenomena, such as typicality effects (Rosch & Mervis, 1975), that are usually given very different explanations from the ones that we propose here. Section 4 briefly discusses the relationship between EPAM and other computational models of learning, including 'chunking' systems that gradually combine pieces of conceptual information over a stream of data. The contention throughout the paper is that there is still life in early discrimination network models like EPAM.

2. Discrimination Network Models

This section focuses on two models that have used discrimination networks to organize and learn conceptual information. One of these, CLS, was constructed to model supervised concept learning in a nonincremental setting. In contrast, EPAM is incremental, unsupervised, and intended to model certain interactions between learning and perception.

2.1 The CLS System

The name CLS refers to nine versions of the Concept Learning System that were constructed and tested by Hunt, Marin and Stone (1966), extending an earlier system developed by Hunt and Hovland (1961). Although these versions differ in assumptions about short-term memory capacity and the strategies of human subjects, they all utilize a discrimination network to represent and learn concepts. The binary networks utilized by the CLS models provide two choices at each test node (a yes branch and a no branch).

The discrimination network is a data structure called a *tree* that consists of test nodes and leaf nodes. In the real world, trees branch upward, but when discrimination nets are pictured, the root test node is at the top and the leaf nodes are at the bottom. In the middle are test nodes that each ask a question about the stimulus. Depending on the answer to that question, the stimulus is sorted to one or another node. The leaf nodes include a chunk of information that presumably applies to the stimulus being sorted.

Figure 1 shows a simple discrimination net which describes the *conjunctive* concept that holds if A and B are both present. If a new object is perceived and the system wants to know whether that object is an example of this concept, the system sorts the object through the discrimination net. The system starts at the first test node, which asks if A is present in the object. If A is not present, then the system sorts the object to the leaf node containing the value (0), which indicates that the new object is not an example of this concept. On the other hand, if A is indeed present, then the system sorts to a new test node that asks if B is present in the object. If B is not present, then the system sorts the object to the leaf node containing the value (0); however, if B is present, it sorts the object to the leaf node containing the value (1).

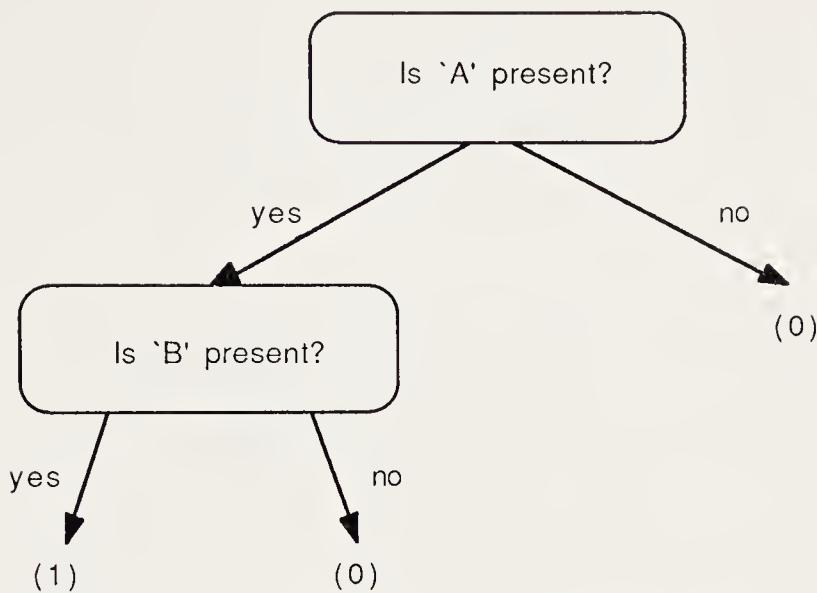


Figure 1. A CLS network for a simple conjunction.

Figure 2 shows a simple discrimination net which describes the simple *disjunctive* concept that holds if A and/or B are present. If a new object is perceived and the system wants to know whether that object is an example of this concept, the system would sort the object through this network. If a leaf node is reached that contains the value (1), then the object is indeed a member of the concept; however, if a leaf node is reached that contains the value (0), this indicates that the object is not a member.

CLS constructs its discrimination network step by step, at each stage ensuring that it correctly classifies the training instances (1) and non-instances (0) it has observed for a concept. If the current network sorts an instance incorrectly, the system erases the net and starts again. An important aspect of the CLS model is its use of heuristics or rules of thumb to direct search for useful tests. For example, one heuristic finds a test by counting the frequency with which characteristics appear in the observed positive instances (1), and picking the most frequent one. This heuristic is based on the implicit assumption that objects which belong to the same concept are more alike than objects which belong to different concepts. This scheme will always produce some tree, though not necessarily a simple one. Other researchers have continued to extend the CLS framework, producing systems like ID3 (Quinlan, 1979, 1986) and ACLS (Patterson & Niblett, 1983), which have been useful in a variety of artificial intelligence applications.

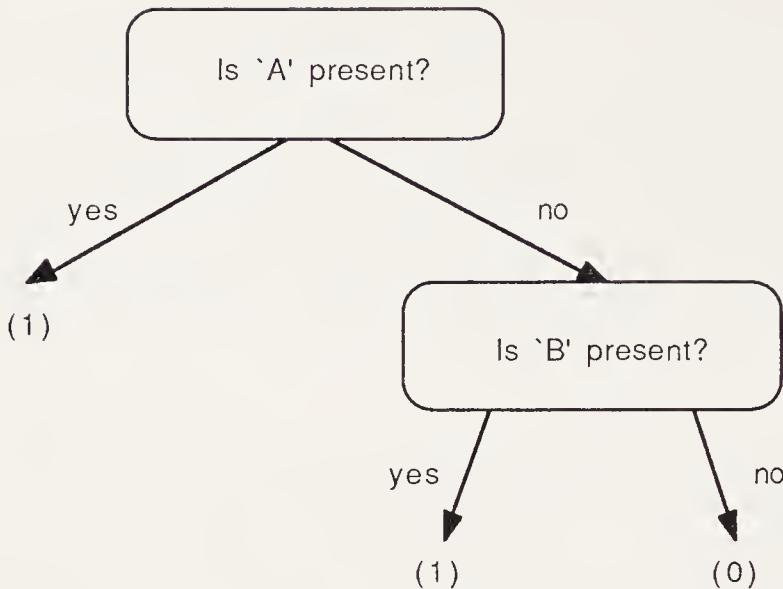


Figure 2. A CLS network for a simple disjunction.

CLS can represent any logical concept, including implications, exclusive disjunctions, and biconditionals (equivalence), but it states these as combinations of conjunctions and disjunctions. That is, each path is a conjunction of features, and choices between paths are inherently disjunctive. The complexity of a concept can be roughly calculated by the number of test nodes in the discrimination net that are required to sort examples of the concept to the correct leaf nodes. With a few exceptions, CLS provides a good predictor of the concepts that people will find difficult to learn and the time it will take people to learn concepts of varying complexities. In contrast, a model tested by Neisser and Weene (1962) begins with all possible concepts in memory, and then weeds out concepts whenever they are found to match a negative instance. This model actually takes longer to converge on simple concepts that involve only a single feature than to converge on more complex ones. The reason is that such one-feature concepts are compatible with many simple disjunctions that involve both the relevant characteristic and other characteristics, and thus require more evidence to identify them.

The CLS model provides a fairly good account of how people store the end results of learning in a simple discrimination net. However, it does not show how several concepts can be held together in a single memory. In addition, the system relies on appropriate supervision and assumes that all experiences can be observed and processed simultaneously. This

latter characteristic appears to limit CLS as a process model of human learning. However, let us now examine a model that relaxes the supervised and nonincremental assumptions of CLS, and that organizes multiple concepts in a single discrimination network.

2.2 The EPAM System

EPAM (Feigenbaum, 1959; Feigenbaum & Simon, 1964) was the first computer program designed to integrate perception and memory within a single system.¹ EPAM's discrimination network acts as an index into memory, much as an index acts as a way of getting to the information in a text. In both cases, the index locates an entry where additional information can be found. An EPAM network differs from a CLS network in several ways.

First, EPAM memory is represented by an *n*-ary net rather than a binary net. This means that the questions at a test node can have any number of answers, not just the 'yes' and 'no' answers of CLS. This aspect makes it possible for EPAM to sort more concepts with fewer test nodes. Some extensions of CLS, such as Quinlan's (1986) ID3, also have this capability.

Second, an EPAM network has a recursive structure. Whereas the questions at CLS test nodes ask for the presence or absence of a feature, the questions at EPAM test nodes usually ask for the name of a 'component' or subobject at a certain position within a larger object. Since an 'object' can itself be composed of smaller 'objects', a recursive recognition process is required to categorize experiences. For example, later we will describe experiments with two-component objects that represent stimulus-response pairs. Each stimulus and response is composed of a sequence of letters, numbers, and other symbols, and each of these are collections of line segments.

Finally, the leaf nodes of an EPAM net are more elaborate than the leaf nodes of CLS nets. Whereas leaves in CLS only indicate whether or not the concept in question has been discriminated, EPAM leaves include an object 'image' or concept description. The image is a list of components that characterizes a set of objects. In many cases, there is

1. Like CLS, the name EPAM (Elementary Perceiver and Memorizer) actually refers to a series of models that share a common approach to representing, using, and acquiring knowledge. In this chapter we focus on features shared by most versions of the system.

an ordering on these components. For example, objects and images may correspond to words or partial words, which are composed of an ordered set of letters. Minimally, the image contains information that was used to retrieve it, but other information can be included at the leaf as well. For example, the words 'Four score' may be sufficient to retrieve an image containing the entire Gettysburg Address. However, EPAM has never been used with such complex concepts. Examples of the concepts that it has learned include three-letter nonsense syllables (Feigenbaum & Simon, 1962; Simon & Feigenbaum, 1964; Gregg & Simon, 1967), chess patterns (Simon & Gilmartin, 1973), and words of four letters (Richman & Simon, 1989).

The main learning process in EPAM is called *discrimination learning*. This is the process through which new branches are added to the network. When a new object (e.g., a nonsense syllable) is presented, the system sorts it through the net until it can go no further. The object will either have stopped at a test node or at a leaf node. If it has stopped at a test node, EPAM simply creates a new branch for the object that branches from this test node. If the object has stopped at a leaf node, the system compares the new object with the old image that is already at the leaf node. If EPAM can find a difference between them, it creates a new test node that lets it discriminate the two by sorting them along different paths. In addition, the system creates a branch for the old image and a branch for the new object at the new test node. The new object becomes an image that initially includes only the components of the object that are found along the path to that image; that is, the image contains only those components that were used to discriminate the object from others.

Both perception and memorization interact with the discrimination network. Perception takes a stimulus from the outside world and uses the discrimination net to 'look up' information about that stimulus, whereas memorization adds new information to the discrimination net or augments existing information. In addition to discrimination learning, EPAM includes a second learning operator called *familiarization* that augments the image at an existing leaf. If a newly classified object matches an image in all of the image's aspects, then some components of the object that are not already present in the image may be added to it. Thus, an image starts off as rather general, but becomes increasingly detailed with more trials.

Table 1. High-level description of EPAM.

```

Epam (Current-node, Observation)
If Current-node is a leaf,
  Then if there is a difference between Observation
    and Current-node's image,
    Then Discriminate and return new image.
    Else Familiarize and return image.
Else let Test be the test at the Current-node.
  Find Component of Observation referred to in Test.
  If the Component is a list of components,
    Then set Component to Epam(Current-node, Component).
  Set Current-node to the child that corresponds
    to Observation's value on Test.
Return Epam(Current-node, Observation).

```

Table 1 summarizes the most important steps in EPAM. The first recursive call in the algorithm identifies a class to which the component being tested belongs. This category label is then used to guide classification deeper into the discrimination network. This component categorization process is similar in intent to the construction of new 'terms' explored by Thompson and Langley (this volume). The second recursive call simply classifies an observation deeper into the discrimination net by traversing an appropriate branch. We have given a recursive version of the system for simplicity, but Richman and Simon (1989) give an iterative (and more complete) version.

The following example simulates EPAM learning a very simple concept. In this case, EPAM is presented with a stimulus syllable and must respond with the appropriate response syllable. During the learning phase, the system is presented with four stimulus-response pairs — (JAD, BIL), (TOD, BER), (JEP, BIL), and (TAD, BER). The system learns the pairs in such a way as to let it produce the correct response if only given the stimulus. At the end of this learning, the net pictured in Figure 3 will have been created.

Imagine that a new stimulus, JID, is presented to the system. In order to find the response, the system creates a new object that it sorts down the net, in this case (JID, __). EPAM ships this object through

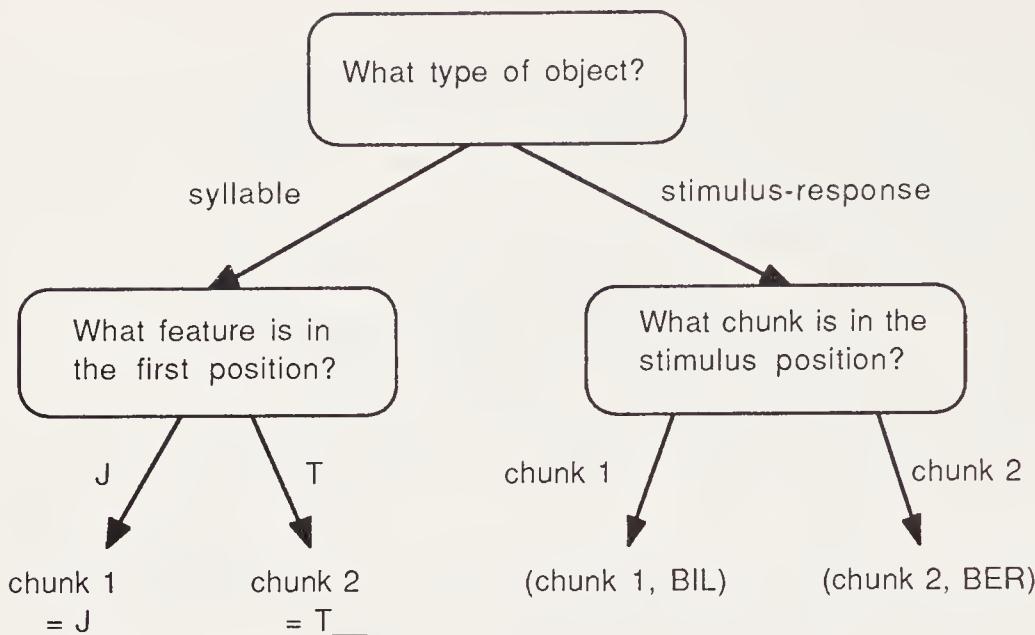


Figure 3. An EPAM discrimination network.

memory, and the image at the leaf node tells the system the responses that should fill the blanks.

Let us consider the path that this object, (*JID, ___*), follows when it is sorted through the discrimination net pictured in Figure 3. The system starts at the root node, which asks about the type of object being discriminated; in this case the type is a stimulus-response pair. The next question asks for the chunk in the stimulus position; in order to answer this question the system must recurse to find the chunk corresponding to *JID*, so it must sort *JID* through the net, restarting at the top node. Again the first question asks for the type of object, and since *JID* is a syllable, EPAM follows the branch for syllable. The next question asks for the feature in the first position, which is *J*; thus, the system follows the *J* branch to *chunk 1*. Now the system is ready to resume processing at the test node, which asks for the chunk in the stimulus position. It follows the branch for *chunk 1* to the leaf node (*chunk 1, BIL*). Using this node, it could name the concept (*BIL*) and also answer questions about whether the attribute *J* was present in object.

Now suppose that, after seeing the first four objects, EPAM is presented with the training object (*JID, BIL*). The system sorts the stimulus-response pair to the right, and then sorts the stimulus down the left path to the image corresponding to *chunk 1*, as before. At this point, there

is no disagreement between the new stimulus JID and the image J_—, so that a familiarization step occurs. No new images are created, but the existing image is specialized using values from the new stimulus.² Afterwards, the image is returned to the node requesting the chunk in the stimulus position, which uses this information to sort the stimulus-response pair to the leaf node (chunk 1, BIL). Since the image stored at this node matches the pair, no discrimination is required, and since the image is completely specified, no familiarization takes place.

Let us further suppose that familiarization for the fifth object yields (JID) at the left-most leaf. If EPAM now encounters a sixth object, (JIK, BER), this requires a straightforward discrimination between the image at the left-most leaf and the new object's stimulus. These syllables can only be discriminated by their values for the letter in the third position.

In EPAM, the learning processes operate under strategic control. In the terminology of computer programming, the learning processes are subroutines that can be called by the different strategies that the system uses in different situations. In the terminology of psychology, the strategy determines the focus of attention. Essentially, the strategy decides what should be learned and tells the learning process to learn it. The point is that EPAM explanations require an understanding of the strategy chosen by the subject.

For example, when EPAM is used to simulate human behavior in a serial anticipation experiment or context effects in a letter perception experiment, the system successfully simulates the human data because it attends to serial lists from the outside and works inwards during learning. Apparently, humans use something very much like this 'anchor point' strategy. When EPAM uses this strategy during the learning phase of these experiments, it produces very close qualitative and quantitative agreement with the serial position curves that arise from the pattern of human errors made during recognition (Feigenbaum & Simon, 1962; Richman & Simon, 1989). In general, errors in letter recognition appear to increase as one moves inward, yielding a rough U-shaped curve if one plots accuracy on the vertical dimension and serial position of each letter on the horizontal axis.

2. Familiarization is a simple process, but it does include a random element. Any value of the stimulus that is not already present in the image may be transferred to the image, and each value is evaluated for transfer independently. Thus, following familiarization, the image of chunk 1 may remain J_—, or it may become JI_—, J_D, or JID.

On the other hand, models that do not allow explicit consideration of subjects' attentional strategies can have a difficult time accounting for serial-position effects. Consider the interactive activation model of Rumelhart and McClelland (1982), which is a connectionist system that associates input features with output classes that are discovered through unsupervised means. To account for the serial position curves that arise from context effects in letter perception, they had to hand set certain system parameters that corresponded to the weights on input from different letter positions. When this version could not model some serial position effects, they assumed that the two end letters are read first, revising their initial parallel processing assumption that all letters in a four-letter word are read at the same time. When they were still unable to get a close quantitative fit with the serial position data, they noted that the mechanisms determining the shape of the serial position curve are "potentially subject to attentional control," and they suggested that "it may be difficult to gain control and understanding of these mechanisms until the factors that govern control of attention are understood" (p. 78). The EPAM model incorporates an attentional control strategy that appears to account for the data.

An even clearer case of the importance of attentional strategies arose when Gregg and Simon (1967) used EPAM to simulate rote learning under replacement and non-replacement conditions. In this study, EPAM incorporated one of two different strategies called 'one-at-a-time' and 'all-or-none'. When subjects were asked, after an experiment, whether they had attempted to focus on one or two syllable pairs at a time (one-at-a-time) or had tried to learn all or most (all-or-none), they showed the same pattern of errors during recall as the model did when using the named strategy. The two-strategy assumption let EPAM explain the sorts of experimental conditions that permit both one-trial learning and more gradual learning.

3. EPAM's Account of Psychological Phenomena

The previous section described two discrimination network models of learning. Along the way, we alluded to certain psychological findings that were explained by each system. We now elaborate the experimental evidence supporting EPAM, the second of these models. For purposes of psychological modeling, it is important to note that EPAM is based upon five assumptions:

1. Central information processing in humans is serial.
2. Primitive units of perception can be collected together into larger units called *chunks* (e.g., line segments can be perceived collectively as a letter). Chunks are the largest stimulus components that are familiar units of perception.
3. Learning an item requires a definite amount of time for each chunk contained in the item and perceived by the subject.
4. The capacity of the immediate memory in humans is a few chunks.
5. Humans can learn any part of a stimulus unit to which they attend (i.e., hold in immediate memory for sufficient time), and their attention may be modified by experimental instructions, attention-directing stimuli, habit, and strategies.

We will focus on some of the most recent findings that support the psychological plausibility of EPAM. First, we reexamine some existing experimental data on memorizing lists of word and nonword letter strings. After this, we consider the system's ability to model well-known typicality effects in concept learning. Feigenbaum and Simon (1984) provide a more complete account of the plethora of data accounted for by EPAM-like models.

3.1 Effects in Letter Perception

Recently, Richman and Simon (1989) used EPAM to simulate context effects in data on letter perception collected by Johnston (1974, 1978). For example, consider the fourth letter of two of the words used in Johnston's tachistoscopic experiment. The P in the word CROP only differentiates between two alternatives (CROP and CROW), so it does not carry as much information as the P in the word CLAP, which differentiates between six alternatives (CLAD, CLAM, CLAN, CLAP, CLAW, and CLAY). The human data do not support the information-theoretic prediction that there should be an advantage for letters that carry less information, and yet Richman and Simon show that EPAM simulates the data quite well. The important difference is that the information-theoretic model (Miller, Bruner, & Postman, 1954) holds that the total amount of information perceived in an interval of time is constant, whereas Richman and Simon's interpretation holds that the number of chunks perceived per perceptual moment is constant.

The interactive activation model (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982) was also able to simulate these data, and in fact a comparison of the two models (Richman & Simon, 1989) revealed that they produced similar explanations of many effects, with chunking in EPAM playing the same role as interactive activation in the connectionist model. However, the connectionist model required different parameter settings in different conditions in order to achieve the word-letter effect.

A second class of phenomena are word-frequency effects, or familiarity effects, which indicate that more frequently occurring words are easier to recognize. When Richman and Simon (1989) compared EPAM with a connectionist model that utilized this hypothesis, both models achieved a close quantitative fit with the word-frequency effect that was apparent in the data on context effects. EPAM produced the effect because its discrimination net included two paths to retrieve frequent written words (either as a whole or by parts), but only one pathway to perceive infrequent words (by parts). Essentially, discrimination nets produce familiarity effects if one assumes that the discrimination network contains more paths through which familiar phenomena can be reached.

Without any adjustments in parameters in order to accommodate different effects, EPAM was also able to simulate many other context effects that were apparent in Johnston's data, including a word-superiority effect, a serial position effect, a similarity effect, and a word-letter effect (Richman & Simon, 1989). We now turn to EPAM's account of another ubiquitous class of psychological phenomena: typicality effects.

3.2 Typicality Effects

Using both concrete nouns and abstract letter strings as stimuli, Rosch and Mervis (1975) demonstrated that the rated *typicality* of an item is related to two aspects of family resemblance — intra-category overlap (the features that are shared with other members of a category) and inter-category overlap (the features that are shared with the members of a contrasting category).

The abstract letter strings permitted Rosch and Mervis to demonstrate a cause and effect relationship. The two aspects of family resemblance not only caused typicality ratings; they also caused learning time effects and categorization time effects, which were similar to phenom-

Table 2. Sample data from an intra-category overlap experiment.

Class	String	Random Ordering	Predictive Features	Typicality Rating	Features Examined	Program Cycles
+	DLT83	D3LBTT	4	6.0	1	162
	DLT8A	LA8DT	3	4.5	1	162
	DLTPM	LDMTP	3	4.5	1	162
	DLGKI	GDLKI	2	3.0	2	260
	D9H60	OD69H	1	1.5	2	260
-	3YH7V	73HYV	1	1.5	2	260
	SXB25	25BSX	3	5.5	1	162
	SXB2Q	2BQXS	3	5.5	1	162
	SXBRE	EXBRS	2	4.0	2	260
	SXVFW	FWSVX	1	2.5	5	554
	S4Z1&	4SZ1&	0	1.0	5	537
	5JZCN	J5NCZ	1	2.5	1	162

ena that had been related to typicality ratings in other contexts (Rips, Shoben, & Smith, 1973; Rosch, 1973).

Rosch and Mervis argued that their results could not be explained by the classical view that a category is defined by an 'all or none' set of criterial features. In fact, they constructed their letter strings so there were no criterial features in common within a category. Instead, they proposed that a category is defined by a 'fuzzy' set of features, each of which has some probability of being present.

Another possibility, which they did not consider, is that a category can be defined by a disjunction of features. A model like CLS or EPAM, which form disjunctions as well as conjunctions, might be able to explain typicality effects in just this way. In fact, we have already carried out some preliminary simulations of Rosch and Mervis' experiments using EPAM to learn such disjunctive concepts.

Table 1 shows some the results from one run in our simulation of Rosch and Mervis' intra-category overlap experiment. The top half of the table presents the six letter strings that comprise one of their experimental categories, which we will call the '+' category; the bottom half shows the contrasting '-' category. The first column contains the strings as they

are listed by Rosch and Mervis (1975). The second column specifies the scrambled versions of the same letter strings as they were presented to subjects and to EPAM. Perusal of the table shows that the strings higher on each list share more letters with other members of the category and thus have a higher degree of family resemblance.

The subject's task was to correctly categorize a letter string by pressing the appropriate button corresponding to the category to which the string belonged. In general, Rosch and Mervis found that letter strings which share more features with other members of a category were learned more quickly (fewer errors), categorized more quickly (lower reaction time), and rated as more typical.

Our preliminary results suggest that EPAM can quantitatively predict Rosch and Mervis' findings. In this simulation run, EPAM found that the disjunction D or L or T or 3 predicted the + category, whereas the disjunction 2 or B or X or J predicted the - category. The letter strings that were more typical had more of the predictive letters.

The order in which the images were learned during the course of this run was: (L +), (D +), (2 -), (B -), (T +), (X -), (J -), and (# +). Note that letters occurring in several strings were associated with categories before letters appearing in few strings. This happened because we encoded attentional strategies which looked for features that were shared by at least two members of the same category. To detect such regularities, EPAM selected a feature from a stimulus, along with its associated class, and held this potential image in short-term memory until it encountered another stimulus with the same combination, at which point the system stored the image in its discrimination net. This strategy embodies CLS' assumption that objects in the same category are more alike than objects in other categories. As a result, strings that share many letters with other examples of the category are generally learned first.

Also notice that all concepts learned in this experiment are single letters associated with class names. This results from the nature of the stimuli in Table 2; there are no letters that are shared between members of different classes. Thus, individual letters can be used to discriminate classes, and we could reasonably restrict images to a single letter and class. However, in cases involving overlap in letters across classes, EPAM would need to form disjunctions of more detailed images to distinguish class members, and thus would require more complex data structures.

Rosch and Mervis calculated typicality by having their subjects arrange the strings in typicality order. We calculated typicality ratings by counting the number of predictive features that occur in each string, and then ordered the strings based upon this count. At the end of this run, the strings in the two categories had the numbers of predictive features and the typicality ratings displayed in the third and fourth columns of Table 1. Due to Rosch and Mervis' procedures, which halted the learning stage of the experiment after two passes in which all predictions were correct, and due to some 'lucky' guessing, the simulation run was completed before EPAM learned any features that would reliably categorize the string 4SZ1&.

We measured reaction times in terms of the number of program cycles that were required to categorize a stimulus. This number is closely related to the number of features that the program examines before it determines the category. Our model searched the string from left to right and stopped as soon as it found a predictive feature. The fifth column in Table 1 shows the number of features examined and the sixth column presents the number of programming cycles required to categorize each string.

An anomalous finding concerns J5NCZ, which appears to be very easy to recognize, despite its atypical nature. This finding is fortuitous: the unusually good score only occurs because the predictive feature J comes first in the string. However, over many random orderings, we would expect that the predictive feature would be evenly distributed through the string. More generally, given its attentional strategy, the model is more likely to find predictive features sooner for more typical objects.

In general, EPAM appears to fit the typicality data simply because it allows for disjunctions and uses a simple strategy of looking for features that members of a category have in common. Future papers will include a more complete account of this model and a more thorough comparison of its behavior with the data found by Rosch and Mervis.

4. Discussion

We have seen that EPAM can account for a wide variety of psychological phenomena, including typicality effects, for which it was not originally designed. In this section we briefly consider some additional findings that lend themselves to explanation in terms of EPAM-like mechanisms.

Along the way, we also consider some related models that incorporate ideas on chunking and discrimination networks.

4.1 Chunks and Short-Term Memory

Ever since 1956, when Miller's examination of short-term memory led him to hypothesize the existence of *chunks* of information, theorists have been trying to come to grips with a definition of this term. Perhaps the best definition is the most recent one, due to Newell (1990, p. 7):

A chunk is a unit of memory organization, formed by bringing together a set of already formed chunks in memory and welding them together into a larger unit. Chunking implies the ability to build up such structures recursively, thus leading to a hierarchical organization of memory.

A key element in the definition of a chunk has been their hierarchical nature. Large chunks are composed of smaller chunks, which are in turn composed of yet smaller chunks, and so on. One approach to understanding chunks has been to build models of memory that are designed to learn and memorize such complex structures. These structures must have recursive properties, so that they can deal with the multiple levels of descriptive granularity implied by chunks. The EPAM discrimination network can be viewed as a model of chunk memory, where images constitute chunks that are, in turn, composed of chunks that must be recognized by a recursive procedure. Of course, there are other models of chunking, which we review here, along with some of the phenomena that motivate them.

Miller (1956) introduced the chunking hypothesis with evidence that short-term memory has a capacity of about seven chunks. With recent refinements (Zhang & Simon, 1985) to accommodate Baddeley's (1986) finding that chunk capacity varies with the time required to pronounce chunk names, the chunking hypothesis continues to provide a good account of the main observed phenomena of short-term memory capacity.

4.2 The Power Law of Practice

In addition to short-term memory phenomena, an important behavioral justification for chunking models is the power law of practice, which has been obtained in nearly every domain that has been examined. This law states that the logarithm of the time to perform a task is linearly

related to the logarithm of the number of practice trials. The chunking explanation of this law is based upon the idea that people learn common chunks (which are typically small) before they learn infrequent chunks (which are typically large).

In their account of chunking, Newell and Rosenbloom (1981) simulate the ubiquitous power law by learning chunks within the stimulus on Seibel's (1963) task, in which there are ten lights (L_1, \dots, L_{10}) that are either on (+) or off (-). In this task, one must recreate the pattern of lights as quickly as possible by pressing buttons on a keyboard. Newell and Rosenbloom's model³ built up chunks in the following fashion:

Gradually, with learning, chunks will form: first chunks such as $(L_1 +)$, which we might also write as L_1^+ ; then chunks such as $(L_3^+ L_4^+)$ or $(L_1^- L_{10}^-)$, ..., and so on. (pp. 41-42)

According to this model, the power law occurs because the first chunks built, being small, are applicable in many situations, whereas later chunks are larger and thus applicable far less frequently. The larger chunks speed up performance when they are applied, but they are only rarely useful. Thus, as a person continues to practice, the rate of learning chunks continues at a constant rate, but reaction time decreases ever more slowly. If the model had instead learned the ten-light patterns as wholes, the power law of learning would not have appeared. Instead, response time would have decreased at a more nearly constant rate, as the chunks learned would only have been applicable when the observed ten-light patterns were repeated exactly.

We have not modeled the power law of practice, but we believe that EPAM's approach to learning discrimination nets is generally consistent with Newell and Rosenbloom's account of this phenomenon. In particular, larger EPAM chunks are themselves composed of smaller chunks. Also, we have noted in our account of typicality that 'common' or characteristic features of a class or action tend to be learned first. These tend to indicate a relatively large bulk of the appropriate responses. Gradual specialization of images (i.e., chunks) at leaves tend to hone responses, but the rate of convergence on optimal responses will slow; whether this expected slowing fits the power law must await further research.

3. Many of the ideas in this model have now been incorporated into the SOAR theory of chunking (Laird, Rosenbloom, & Newell, 1986; Newell, 1990).

4.3 Response Latencies

Although the EPAM model has provided a fairly good explanation of many verbal-learning phenomena, it has not given a coherent explanation of other significant findings, such as response latency effects. However, there does exist a discrimination net model that provides a good account of such phenomena — Hintzman's (1968) SAL. This model introduced the idea that a paired associate chunk at a leaf node could have more than one response. For example, BAJ could have two associated responses, such as JID or VOG.

With this new idea, Hintzman was able to explain a number of phenomena that EPAM does not cover, including proactive inhibition, multiple responses to a single stimulus, and response latency times. Although Hintzman did not intend to simulate response latency curves with his model, he found that it would quantitatively predict the result that responses get faster with learning. The essential idea is that responding is slower when stimulus-response chunks at leaf nodes contain several responses. Adding new tests and branches in the net speeds responses because it separates the several responses among separate chunks.

We believe that this account has wider implications as well. Consider that one type of response latency result, called the 'fan effect', is generally interpreted as resulting from the "limited capacity feature of the spreading activation process" (Anderson, 1985, p. 152). In this view, the fan-like spread of activation is thought to occur within the entire declarative portion of long-term memory. Anderson (1974) first discovered the fan effect in an experiment in which subjects learned propositions that associated people with locations (e.g., A hippie is in the park. A sailor is in the church.) After learning, subjects were asked to identify whether they had observed a given proposition previously. In general, responses were slower when a person was associated with several locations, or a location with several persons.

Anderson's (1974) original explanation of this effect attributed it to a serial search from one component of the proposition to the other. A similar serial search could be responsible for the response latency effects in SAL's simulations of paired associate learning, which showed that response latency slows when additional responses are paired with a single stimulus within a single chunk (leaf). In this interpretation, the fan effect is simply an expression of the fact that response latency depends upon the internal structure of chunks in the SAL model.

4.4 Alternative Models

Until recently, with the advent of connectionist systems like McClelland and Rumelhart's (1981) interactive activation model (see Section 3.1), the chief competition to chunking hypotheses has come from models related to information theory (Shannon, 1948), which hypothesize that people make accurate use of conditional probabilities. This is certainly not the case universally. For instance, Simon (1957) studied maze-learning experiments in which there were more than two possible paths that a subject could choose at a choice point. After factoring search time out, Simon found that the time it took subjects (both rats and people) to memorize the correct series of choices was not related to the number of alternative paths at a choice point, as information theory would imply, but was rather related to the number of choice points (i.e., the number of chunks that would need to be learned). However, it seems reasonable to assume that in many cases humans use probabilities to qualify the symbolic representations that are implied by most chunking models.

In fact, Feigenbaum and Simon (1984), Fisher and Pazzani (Chapter 1, this volume), and Gennari et al. (1989) describe hybrid models that have EPAM-like and information-theoretic aspects. This line of research has led to Kolodner's (1983) CYRUS and Fisher's (1987) COBWEB, both of which build on the ideas in EPAM. These systems retain EPAM's general strategy of top-down classification by matching features, but they employ information-theoretic and other heuristics to organize memory and categorize new experiences. In particular, COBWEB uses feature probabilities to guide classification and to generate predictions about the most probable responses to queries. Fisher and Langley (1990) describe how this hybrid accounts for a number of psychological phenomena, including the results on typicality and fan effects that we described earlier.

From a computational standpoint, EPAM was not designed as a concept formation system and, indeed, as currently implemented it would fare poorly compared to more recent models. Given the relationship between EPAM and COBWEB, a promising direction for future research is to investigate the extent to which the various extensions to EPAM that are embodied in COBWEB contribute to differences in the fit of each system to psychological data. However, it is important to note that COBWEB and most other recent systems also simplify some of EPAM's

original assumptions. Notably, they do not explicitly concern themselves with the possibility of different attentional strategies, nor do they allow componential representations of objects (i.e., where objects are themselves composed of objects), though this latter capability is present in Thompson and Langley's (this volume) LABYRINTH system.

5. Conclusion

This chapter has focussed on EPAM, an early learning system that has greatly influenced the line of research known as *concept formation*. We have shown that recent versions of EPAM explain certain perceptual and learning phenomena, notably context effects in letter perception, as well as more universal typicality phenomena. Together with its influence on more recent work, this suggests that discrimination net models continue to have something to offer as models of concept formation. Future work with EPAM will concentrate on further extending the behaviors that are explained through discrimination network models of chunk learning.

Acknowledgements

I thank Doug Fisher and Pat Langley for valuable comments on earlier drafts that greatly improved the quality of the chapter.

References

- Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology*, 6, 451-474.
- Anderson, J. R. (1985). *Cognitive psychology and its implications* (2nd ed.). New York: W. H. Freeman.
- Baddeley, A. D. (1986). *Working memory*. Oxford: Clarendon Press.
- Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking*. New York: John Wiley.
- Crossman, E. R. F. W. (1959). A theory of the acquisition of speed skill. *Ergonomics*, 2, 153-166.
- Feigenbaum, E. A. (1959). *An information processing theory of verbal learning* (Report No. P-1857). Santa Monica, CA: The RAND Corporation.

- Feigenbaum, E. A., & Simon, H. A. (1962). A theory of the serial position effect. *British Journal of Psychology*, 53, 307-320.
- Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, 8, 305-336.
- Fisher, D., & Langley, P. (1990). The structure and formation of natural categories. In G. H. Bower (Ed.) *The psychology of learning and motivation* (Vol. 26). San Diego, CA: Academic Press.
- Gregg, L. W., & Simon, H. A. (1967). An information-processing explanation of one-trial and incremental learning. *Journal of Verbal Learning and Verbal Behavior*, 6, 780-787.
- Hintzman, D. L. (1968). Explorations with a discrimination net model for paired-associate learning. *Journal of Mathematical Psychology*, 5, 123-162.
- Hunt, E. B., Marin, J., and Stone, P. J. (1966). *Experiments in induction*. New York: Academic Press.
- Hunt, E. B., & Hovland, C. I. (1961). Programming a model of human concept formation. *Proceedings of the Western Joint Computer Conference* (pp. 141-155).
- Johnston, J. C. (1974). *The role of contextual constraint in the perception of letters in words*. Doctoral dissertation, Department of Psychology, University of Pennsylvania, Philadelphia.
- Johnston, J. C. (1978). A test of the sophisticated guessing theory of word perception. *Cognitive Psychology*, 10, 123-154.
- Kolodner, J. (1983). Reconstructive memory: A computer model. *Cognitive Science*, 7, 281-328.
- Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1, 11-46.
- McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: Part 1. An account of basic findings. *Psychological Review*, 88, 375-407.
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63, 81-97.

- Miller, G. A., Bruner, J. S., & Postman, L. P. (1954). Familiarity of letter sequences and tachistoscopic identification. *The Journal of General Psychology*, 50, 129-139.
- Neisser, U., & Weene, P. (1962). Hierarchies in concept attainment. *Journal of Experimental Psychology*, 64, 640-645.
- Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.
- Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, NJ: Lawrence Erlbaum.
- Patterson, A., & Niblett, T. (1983). *ACLS user manual* (Technical Report). Glasgow: Intelligent Terminals Ltd.
- Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), *Expert systems in the micro electronic age*. Edinburgh: Edinburgh University Press.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81-106.
- Richman, H. B., & Simon, H. A. (1989). Context effects in letter perception: Comparison of two theories. *Psychological Review*, 96, 417-432.
- Rips, L. J., Shoben, E. J., & Smith, E. E. (1973). Semantic distance and the verification of semantic relations. *Journal of Verbal Learning and Verbal Behavior*, 12, 1-20.
- Rosch, E. (1973). On the internal structure of perceptual and semantic categories. In T. E. Moore (Ed.), *Cognitive development and the acquisition of language*. New York: Academic Press, 1973.
- Rosch, E., & Mervis, C. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7, 573-605.
- Rumelhart, D. E., & McClelland, J. L. (1982). An interactive activation model of context effects in letter perception: Part 2. The contextual enhancement effect and some texts and extensions of the model. *Psychological Review*, 89, 60-94.
- Seibel, R. (1963). Discrimination reaction time for a 1023-alternative task. *Journal of Experimental Psychology*, 66, 215-226.
- Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27, 379-423, 623-656.

- Siklóssy, L. (1972). Natural language learning by computer. In H. A. Simon & L. Siklóssy (Eds.), *Representation and meaning: Experiments with information processing systems*. Englewood Cliffs, NJ: Prentice-Hall.
- Simon, H. A., & Gilmartin, K. J. (1973). A simulation of memory for chess positions. *Cognitive Psychology*, 5, 29-46.
- Simon, H. A. (1957). Amounts of fixation and discovery in maze learning behavior. *Psychometrika*, 22, 261-268.
- Simon, H. A., & Feigenbaum, E. A. (1964). An information-processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning. *Journal of Verbal Learning and Verbal Behavior*, 3, 385-396.
- Simon, H. A., & Siklóssy, L. (1972). General introduction. In H. A. Simon & L. Siklóssy (Eds.), *Representation and meaning: Experiments with information processing systems*. Englewood Cliffs, NJ: Prentice Hall.
- Zhang, G., & Simon, H. A. (1985). STM capacity for Chinese words and idioms: Chunking and acoustical loop hypotheses. *Memory and Cognition*, 13, 193-201.

CHAPTER 5

Concept Formation in Structured Domains

KEVIN THOMPSON

PAT LANGLEY

1. Introduction

Most recent work on unsupervised concept learning has been limited to unstructured domains, in which instances are described by fixed sets of attribute-value pairs. Many domains can be described in this simple language. Frequently, however, instances have some natural *structure*; objects have components and relations among those components. In such domains, an attribute-value language is inadequate.

This chapter describes LABYRINTH, an implemented system that induces concepts from structured objects. We view LABYRINTH as an approach to *incremental concept formation*. Following Gennari, Langley, and Fisher (1989), we define this task as:

- *Given*: a sequential presentation of objects and their associated descriptions;
- *Find*: clusterings that group these objects into concepts;
- *Find*: a summary description for each concept;
- *Find*: a hierarchical organization for these concepts.

The goal of incremental concept formation is to find concepts that allow useful predictions from partial information. COBWEB (Fisher, 1987), UNIMEM (Lebowitz, 1987), and CYRUS (Kolodner, 1982) all incorporate approaches to this task, but these earlier systems are restricted to attribute-value languages. LABYRINTH can make effective generalizations by using a more powerful structured representation language.



Figure 1. Four instances from a simple relational domain.

Figure 1 shows a simple domain with four structured objects. Each object has three components, two LEFT-OF relations, and a single ON-TOP relation. Each of the three component objects is in turn described with the two attributes **SHAPE** and **COLOR**. We will sometimes describe each object as being “labeled” as a member of one of these two classes: **RIGHTSTACK** (instances that have a two-high stack to the right of another object), and **LEFTSTACK** (instances with a stack to the left). These labels are for expository purposes; they play no part in classification or learning. We will use the domain in Figure 1 throughout our discussion of related work and our sketch of LABYRINTH’s operation.

A structured domain can sometimes be converted to one described only by fixed attributes and their values. However, as Quinlan (1990) has argued, using an attribute-value language simplifies the learning task, but may prevent the learning of concise, effective generalizations. For example, if we enforce a consistent ordering of components in the four instances from Figure 1, one could in theory “flatten” the representation of each instance to a fixed set of attributes. However, to find the most concise representation of the two STACK concepts, the learner must be able to consider different bindings between the components of each object. With a structured representation, the learner can find a concept of the form: “There are three objects, *X*, *Y*, and *Z*; *X* is on top of *Y*, and both *X* and *Y* are to the left of *Z*”. Moreover, a system that uses a structured representation has the potential to recognize a LEFTSTACK in which the stack has three objects instead of two; a learner that is limited to a predetermined set of attributes would have limited flexibility in such situations.

In the following section, we summarize related research on concept learning in structured domains. We then describe LABYRINTH’s representation of objects and concepts, along with its memory organization. After this, we illustrate the system’s classification and learn-

ing algorithm by tracing through a simple example, and then discuss LABYRINTH's mechanisms for matching structured objects against concepts in memory. We conclude with a discussion of open issues and plans for future research.

2. Concept Learning in Structured Domains

LABYRINTH carries out incremental, unsupervised concept learning in structured domains. It learns probabilistic concepts and uses them to make predictions of missing attribute values, components, and relations. It also decomposes objects into sets of components to constrain matching. Many of these characteristics are found in earlier systems, but no one system has integrated all these traits. In this section, we review six systems in detail. The review is not intended to be exhaustive, but to highlight previous work that has examined some of the issues LABYRINTH addresses, and to argue for the importance of integrating these characteristics. Of the six systems we describe, two involve induction over objects described as attribute-value sets, and are important for their contribution to the unsupervised learning literature. The other four are important for their contribution to the understanding of induction in structural domains. For each system, we discuss its representation language for instances and concepts, its classification mechanism, and its learning algorithm.

2.1 SPROUTER: Incremental Learning with Structured Objects

Hayes-Roth and McDermott's (1978) SPROUTER is representative of several systems that carry out learning of maximally specific conjunctive descriptions from examples (e.g., Vere, 1975; Winston, 1975). These systems focus on finding characterizations, or descriptions, of classes given by an external teacher. Dietterich and Michalski (1981) present a careful comparison of a number of such systems.

SPROUTER's representation language for both instances and concepts is equivalent to quantifier-free first-order predicate logic; the system views atoms as existentially quantified variables denoting distinct objects. In addition, the language allows organization of predicates that are semantically related into *case frames* to reduce match costs. For example, the COLOR feature of each object in a scene would be put

in the same frame.¹ SPROUTER's representation of the RIGHTSTACK-2 instance from Figure 1 would be:

```
{ {grey:a,blue:b,red:c}
  {square:a,odd:b,circle:c}
  {on-top:a,below:c}
  {left:b,right:c}
  {left:b,right:a}} .
```

This representation asserts that there is a structured object composed of three component objects, *a*, *b*, and *c*, that the object labeled *a* has properties of being GREY, SQUARE, and so on. Because SPROUTER induces only conjunctive generalizations, its concepts are represented in the same language as the instances. Generalization arises from dropping terms and replacing constants with variables.

SPROUTER uses an incremental algorithm to carry out a heuristic beam search² through the space of hypotheses using a specific-to-general scheme. It uses the first instance as an initial hypothesis set and creates new sets of conjunctive generalizations (e.g., concepts) in response to later instances. The system conducts an *interference match* between each generalization and each new instance. This match identifies common properties, and replaces the current hypothesis with one or more new generalizations. SPROUTER constrains its search through the hypothesis space by limiting the number of partial matches stored and pruning those with low utility; the evaluation function that guides this search is defined to increase with the number of relations in a match and to decrease with the number of objects related.

To match a structured instance *I* and concept *C*, SPROUTER selects an arbitrary case frame *F_I* from *I* and then finds bindings between *F_I* and a case relation *F_C* from *C* with identical case labels. As we noted, the system uses its case frames to guide its selection of *F_C*. SPROUTER uses *F_I* and *F_C* and the bindings between them to form an initial partial match; it then selects a new case frame from *I* and repeats the process. If the bindings between the frames are consistent with the previous

-
1. This case frame representation appears to play a role similar to that of the "attributes" used by many inductive learning systems. Hayes-Roth and McDermott (1978, p. 402) use this idea of defining certain shared properties. In addition, SPROUTER appears to use the case frames to direct matching between properties that other systems would represent as *n*-ary predicates (e.g., ABOVE, BELOW).
 2. In contrast, Vere's THOTH (1975) considers *all* maximal generalizations.

partial match, SPROUTER adds the case frames and their bindings into the previous hypothesis. If the bindings conflict, the system forms a new generalization that contains only the bindings between the current case frames. In this way, each instance causes the system to extend or revise its set of hypotheses.³

SPROUTER is important because it is one of the most sophisticated of the early inductive systems able to learn in structured domains. It forms plausible characterizations in complex structured domains, using simple heuristic methods to limit an inherently exponential search problem for a maximal characterization. However, it is limited to a conjunctive concept language, and, like many early inductive learning systems, lacks a clear performance component. One can imagine SPROUTER being used in a recognition task in which the system uses a complete matcher to determine if a test instance matches any of the hypotheses. Most importantly for our current discussion, SPROUTER is a *supervised* algorithm, and therefore does not address issues of cluster formation and memory organization. We turn now to classification and learning in situations where the objects are unlabeled.

2.2 CLUSTER/2: Conceptual Clustering

In many situations, a learner cannot rely on direct labeling of each object; in these cases, one must autonomously organize observations into categories. Older work in this area, known as *numerical taxonomy* (Everitt, 1980), concentrates on what Fisher and Pazzani (Chapter 1, this volume) call the *clustering* task, that of determining useful subsets of an unclassified set of objects. With their system CLUSTER/2, Michalski and Stepp (1983) introduce the *conceptual clustering* paradigm. This task includes not only clustering, but also *characterization*: the formation of intensional concept descriptions from each extensionally defined cluster. This latter subtask is the focus of supervised learning systems such as SPROUTER; it is the combination of the clustering and characterization problems that distinguishes conceptual clustering.

3. Dietterich and Michalski (1981) divide the SPROUTER algorithm into two separate steps: finding all possible bindings between identical case frames, and finding consistent unions of them. Although this appears to be identical in principle to the description by Hayes-Roth and McDermott (1978), the latter argue (p. 405) that finding all possible bindings initially would be prohibitively expensive.

We review CLUSTER/2 here because it constitutes an early example of a machine learning approach to conceptual clustering, and is an important component of CLUSTER/S, which we describe in Section 2.3. However, we emphasize that the system does not carry out structured concept learning, since its representation language has only unary predicates. Both objects and concepts are represented in the *annotated predicate calculus*, an extension of the predicate calculus with additional operators for internal disjunction (e.g., $[shape(block) = odd \vee square]$) and internal conjunction (e.g., $[shape(block1 \& block2) = odd]$). Each predicate, variable, and function in this language has an associated *annotation*, giving “domain knowledge” about the type of the value set and related descriptors in a value hierarchy (e.g., ODD and SQUARE are subsumed by the value ANY-SHAPE).

CLUSTER/2 is nonincremental, using a divisive technique to generate a disjoint hierarchy of concepts. It starts with a root node consisting of all objects in the data set. It then splits that node into a set of mutually exclusive clusters and recurses to construct subhierarchies below each node. CLUSTER/2 is a complex algorithm, with several levels of nested search, each using a similar (user-supplied) evaluation function but using different search techniques. At the highest level, the algorithm searches through partitions consisting of different numbers of clusters k , from two up to a user-specified parameter K_{max} , finding a “best” partition for each value of k and then selecting the best of these $K_{max} - 1$ partitions.

The CLUSTER/2 system operates by transforming its unsupervised learning task into a series of supervised learning tasks. To find a partition for a single value of k , it begins by randomly selecting k “seed” objects for each seed, treating that seed as a positive instance and all others seeds as negative instances. For each seed, CLUSTER/2 uses the star-generating algorithm described by Michalski (1983) to find the set of alternative most general descriptions that distinguish the cluster based on that seed from those of the other seeds; it selects the best of these as the cluster for that seed.⁴ These descriptions form a disjoint clustering over the original set of objects. If this iteration produces a set of clusters superior to the previous one, seeds are selected from the central tendency of each of these clusters; otherwise, seeds are selected from instances at the borders of the clusters. This new set of seeds is used to

4. Because they are the most *general* definitions, the clusters can overlap; an additional search is used to make them disjoint.

generate a new clustering, with the algorithm terminating when some predefined number of consecutive iterations generate no improvement.

CLUSTER/2 is interesting to our current discussion primarily because it introduces the task of conceptual clustering, and aids the discussion of *CLUSTER/S* below. The algorithm is computationally expensive and relies on several user-supplied thresholds to control its search. One would prefer the algorithm to determine the proper number of clusterings without a complete search for each value of k . In addition, like most early inductive learning systems, *CLUSTER/2* lacks a performance mechanism with which to evaluate its clusterings, and instead relies on metrics like “quality of discovered classes” and “quality of fit to data” to evaluate the system’s performance. However, as with *SPROUTER*, one can imagine using the induced concepts and a complete matcher to recognize test instances.

2.3 CLUSTER/S: Clustering with Structured Objects

CLUSTER/S (Stepp, 1984; Stepp & Michalski, 1986) extends *CLUSTER/2*, combining a supervised learning algorithm for structured domains with the earlier work on attribute-based conceptual clustering to form concept hierarchies from structured objects. *CLUSTER/S* represents objects and concepts in the annotated predicate calculus, as with *CLUSTER/2*, but includes n -ary predicates along with simple attributes. The *RIGHTSTACK-2* object in Figure 1 would be represented as:

```

 $\exists p_1, p_2, p_3 \ [color(p_1)=grey] \ [shape(p_1)=square]$ 
 $\quad [color(p_2)=blue] \ [shape(p_2)=odd]$ 
 $\quad [color(p_3)=red] \ [shape(p_3)=circle]$ 
 $\quad [on(p_1, p_3)]$ 
 $\quad [Left-of(p_2, p_3)]$ 
 $\quad [Left-of(p_2, p_1)] \ .$ 

```

Like its predecessor, *CLUSTER/S* effectively reduces an unsupervised learning problem to a series of supervised learning subproblems. The system breaks the problem of structured object clustering into two segments: reducing each object description to an attribute-value representation using a supervised learner, then using an attribute-based method to cluster these redescribed instances. It thus circumvents the complexity of clustering structural descriptions by clustering only those parts of each object expressible in a common language of fixed attributes.

CLUSTER/S first finds a maximally specific generalization, or *template*, of the set of structured objects, using a characterization algorithm adapted from INDUCE/2 (Hoff, Michalski, & Stepp, 1983). This generalization M expresses the common substructure of all the instances, covering all objects while preserving enough information from each object to identify correspondences between objects. Using M , CLUSTER/S can extract a subset of the literals from each instance in a common language of quantifier-free attributes. In this way, a structured domain is converted to an attribute-value language by a search for common structural properties. The re-defined objects, described by a fixed set of literals, are then clustered with the CLUSTER/2 algorithm, and these clusters can easily be converted back to a structured form using M . A postprocessing step augments each cluster with those parts of each instance "left out" in the conversion to the template language.

Stepp (1984) describes how the matching of two objects in structured domains can be viewed as a graph-matching problem, and notes its computational complexities. The algorithm used to generate M appears to employ a beam search through a space of partial matches for the instances, starting with a single attribute and gradually extending the set of template hypotheses by adding more attributes. The algorithm contains a "trimming" step to limit the combinatorial explosion of match hypotheses, but Stepp fails to describe clearly the evaluation function.

Many of the comments applicable to CLUSTER/2 are applicable to CLUSTER/S as well. The latter system is important as the first machine learning approach to unsupervised induction of concepts from structural data. However, because it uses CLUSTER/2 as a main subroutine, it shares the disadvantages of being computationally expensive and non-incremental. Like CLUSTER/2, it lacks a clear performance component. In addition, because it clusters only over those relations and attributes that are found in the template M , it cannot find generalizations that use features shared only by a subset of the instances.

2.4 Levinson's Incremental Self-Organizing Memory

Levinson (1985) describes a database retrieval system for concepts represented as graphs. He applies his system to the domain of organic chemistry, but argues that it is widely applicable, and demonstrates it briefly on chess. In contrast to the other systems we review, Levinson's system does not learn at the knowledge level (Dietterich, 1986), but aims to acquire efficient indices for retrieving specific cases.

Instances are represented as labeled graphs. For example, a hydrocarbon molecule would be represented as a graph with edges for bonds and vertices for individual atoms. Concepts are described as logical conjunctions of relations that share arguments, as in most work on structured concept learning. Naturally, concepts are partially ordered by generality, but Levinson's system uses this ordering for memory organization, not just to constrain search. The system stores all concepts in a graph partially ordered by the relation SUBGRAPH-OF. The most general nodes are individual literals; the most specific concepts (terminal nodes) represent actual objects. If one concept (S) is more specific than another (G), then S is connected to G by a SUBGRAPH-OF link, unless there is some other concept that is more general than S and more specific than G .

Levinson's system uses this memory organization to retrieve efficiently the best matches in memory to a presented object. A new instance I is sorted "in parallel" down all paths in the concept hierarchy, starting at the most general node. If I matches the concept C , then the instance is recursively sorted to C 's children. This continues until I reaches a concept that it fails to match, or until it reaches a terminal node.

If an instance I reaches and matches a terminal node during sorting through memory, no learning occurs. However, if I has matched a concept G but does not match any of G 's children, the system considers forming generalizations based on I and each of those children. For each child S , it finds all maximal partial matches between I and S , then selects the best match according to efficiency concerns. It creates a new intermediate level concept L that is more general than S and more specific than G , inserting the appropriate SUBGRAPH-OF links. The system also determines whether L should be inserted between any other pair of concepts that are directly connected by SUBGRAPH-OF links. Note that the system can create multiple concepts for a given instance I , since I is sorted down multiple paths in the hierarchy. However, the algorithm does *not* move beyond the input data, but only summarizes the observed instances. The algorithm forms general concepts, but only uses them as an efficient indexing scheme for retrieving specific cases. Wogulis and Langley (1989) use different mechanisms to acquire a similar memory structure, and point out that such systems lead to more efficient classification by storing intermediate concepts; in Levinson's system, the subgraphs allow more efficient indexing of structured objects.

2.5 MERGE: Organizing Structured Objects into Components

Wasserman (1985) describes MERGE, a system that carries out incremental concept acquisition and organization for structured objects. Like Levinson's system, it uses a memory organization to facilitate incremental update of memory in response to new objects. In contrast to Levinson's work, MERGE moves beyond the data, making generalizations that summarize instances and using those generalizations to fill in missing information.

MERGE's instance representation is most interesting to the current discussion. The standard representation for structured objects, a predicate calculus formalism, is equivalent to arbitrary directed graphs. Determining a match between two structured objects represented as graphs is equivalent to the NP-complete subgraph isomorphism problem. In response to this combinatorial problem, both SPROUTER and the CLUSTER programs use heuristics to control the search for a characterization. Wasserman takes an alternate approach: representing objects in a language in which generalizations are more easily found. MERGE is described as a system for learning from *hierarchies*, rather than from arbitrary structured objects. An instance hierarchy is represented as a tree of nodes partially ordered by a *fundamental relation*. This relation is used to decompose the structured object into smaller "components", which in turn can have components, and so on. The representation bottoms out with primitive objects described only by associated object properties. To distinguish these instances from concept hierarchies, we refer to instance hierarchies as *partonomies*.

In the domain of physical objects, the fundamental relation would be the PART-OF relation, but in other applications, Wasserman uses relations like REPORTS-TO (for human organization charts) and Is-A (for biological taxonomies). Wasserman notes that there are several possible organizational concepts, or fundamental relations, for any given domain, but argues that a single outstanding relation gives a complete partonomy of each object. This basic partonomy is augmented by *non-fundamental* relations, which are predicates other than the specified fundamental relation,⁵ and which take as arguments any object in the instance tree. The RIGHTSTACK-2 object would thus be represented as:

5. It appears that MERGE is restricted to binary relations, although Wasserman never clearly states this constraint, and the extension to *n*-ary predicates seems straightforward.

```
(Rightstack-2 (component1 (color blue) (shape odd))
             (component2 (color red) (shape circular))
             (component3 (color grey) (shape square)))
             ((Left-of component1 component3)))
             ((Left-of component1 component2))
             ((on component3 component2)) .
```

Wasserman uses a graphical notation for instances; we have substituted an equivalent syntax for purposes of comparison.

Generalizations are essentially the logical intersection of the instances from which they are made; the system avoids the extra search required to make disjunctive generalizations. Abstractions are made over the structural information (relations); Wasserman deemphasizes the importance of abstractions over object properties, although an unspecified algorithm does generalize the object properties.

MERGE incrementally forms abstraction hierarchies from a sequence of instance partonomies. The system classifies not only the entire structured object, but each of its subhierarchies⁶ as well. Each of these subhierarchies is classified into a separate concept hierarchy, thus giving a forest of concept hierarchies, one for each “type” of object (apparently, each level of an instance is a different type). Like UNIMEM (Lebowitz, 1987), MERGE explicitly represents differences and similarities between a child and its parent with the use of inheritance to add, subtract, or substitute features. To classify each subhierarchy I of the instance, MERGE starts at the root of the concept hierarchy for that type of object and recurses through the tree. At each parent P , it finds the best candidate child C of that node. If C ’s score is no better than that of P , the algorithm stops and makes the object a new child of P . Otherwise, it incorporates I into C and recurses. MERGE’s evaluation function is a scoring scheme relying on several heuristics. Two components that have a common ancestor in a concept hierarchy are rewarded if that ancestor is low in the partonomy, and components are scored based on their literal similarity. In addition, components are weighted less in the overall score than the object itself.

Wasserman (1985) downplays the computational difficulties of matching structured objects. The augmented partonomy representation of MERGE lends itself to decomposition of the match problem into a series

6. Remember that these are PART-OF hierarchies that represent individual instances, not concept hierarchies.

of component-matching problems, unlike the arbitrary graph representation used by earlier systems. However, Wasserman does not promote this as an advantage of using partonomies. Although the system classifies each subtree of the instance, it does not use the results of component classifications in classification of instances. It thus faces the problem of generating abstractions from arbitrary trees. The MERGE matcher compares two trees by working its way bottom up through each partonomy, finding “best” matches at each level and recursing. It appears to use an exhaustive matcher that matches m instance components against n concept components, with a computational complexity of $O(n!)$. In addition, the matcher has operators for “level hopping” that involve checking whether a component at level x of the one partonomy matches well against a component at a different level y of another partonomy.

2.6 COBWEB: Probabilistic Concepts

Because the COBWEB system (Fisher, 1987; McKusick & Thompson, 1990) forms the basis for LABYRINTH, we review it in some detail. COBWEB is an incremental, unsupervised concept learner, like CYRUS (Kolodner, 1983) and UNIMEM (Lebowitz, 1987). It differs from its predecessors in its use of *probabilistic* concepts (Smith & Medin, 1981) and its use of a principled evaluation function that favors clusters that maximize the potential for inferring information. In addition, Fisher emphasizes the use of concept formation systems in the context of a performance task — missing attribute prediction — and explicitly evaluates his system using this task. This contrasts with most earlier unsupervised learners, which have been evaluated only in light of the concepts formed and their “comprehensibility”.

COBWEB represents each instance as a set of nominal⁷ attribute-value pairs, and it summarizes these instances in a hierarchy of probabilistic concepts. Each concept C_k is described as a set of attributes A_i and their possible values V_{ij} , along with the conditional probability $P(A_i = V_{ij}|C_k)$ that a value will occur in an instance of a concept. The system also stores the overall probability of each concept, $P(C_k)$. Thus, whereas CLUSTER/2 can represent an attribute *color* with alternate values *blue* \vee *red*, a COBWEB concept can represent the observed conditional

7. Gennari, Langley, and Fisher (1989) describe CLASSIT, a variant of COBWEB that accepts real-valued attributes. LABYRINTH’s mechanisms are independent of the feature types of primitive object attributes.

probabilities, $P(\text{color} = \text{blue}|C_k) = 0.6$ and $P(\text{color} = \text{red}|C_k) = 0.4$. The use of probabilistic concepts is crucial to COBWEB's design. As Hanson and Bauer (1989) point out, many categories are better represented as probabilistic concepts than as sets of common features. For incremental systems with a restricted hypothesis memory, probabilistic concepts are crucial to avoid brittleness in the face of noisy or approximate concepts. Probabilistic concepts allow gradual updating of descriptions and recovery from misleading training orders because they store more information about the instances that form the concept.

COBWEB organizes its acquired concepts in a probabilistic concept hierarchy, in which each node is indexed by Is-A links from its parents, rather than difference links as with UNIMEM and MERGE. Specific instances are stored as leaves of the concept hierarchy, and the root node summarizes all instances seen in the domain. Such hierarchies are crucial for focusing attention and allowing small local changes to memory during incremental processing.

The system integrates classification and learning, sorting each instance through its concept hierarchy and simultaneously updating memory. Upon encountering a new instance I , COBWEB incorporates it into the root of the existing hierarchy and then recursively compares the instance with each new partition as I descends the tree. At a node N , the system considers incorporating the instance into each child of N as well as creating a new singleton class, and evaluates each resulting partition. If the evaluation function prefers adding the instance to an existing concept, COBWEB modifies the concept's probability and the conditional probabilities for its attribute values and then recurses to the children of that concept. If the system decides to place the instance into a new class, it creates a new child of the current parent node, and the classification process halts. COBWEB also incorporates two bidirectional operators, splitting and merging, that make local modifications to the hierarchy structure. These mitigate sensitivities to instance orderings, giving the effect of backtracking in the space of concept hierarchies without the memory overhead required by storing previous hypotheses.

To choose among these operators, COBWEB uses the probabilistic information stored in memory in an evaluation function — *category utility* — which favors high intra-class similarity and high inter-class differences. Gluck and Corter (1985) derive this function from information theory, and Fisher modifies it slightly to control classification and learn-

ing behavior in COBWEB. Given a set of n categories, category utility is defined as the *increase* in the expected number of attribute values that can be correctly guessed over the expected number of correct guesses without such knowledge. The version used by COBWEB is

$$\frac{\sum_{k=1}^K P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij}|C)^2}{K}, \quad (1)$$

where k varies over categories, i over attributes, and j over values for each attribute. This function evaluates a *partition* — defined as a parent node C and its immediate children C_k . The term $P(C_k)$ refers to the *a priori* likelihood that an instance is a member of the child C_k , whereas $P(A_i = V_{ij}|C_k)^2$ is a measure of *within-class similarity*, that is, how well the instances summarized by C_k resemble one another. The subtraction of the parent's within-class similarity $P(A_i = V_{ij}|C)^2$ lets category utility measure the information gained by partitioning the parent class into a set of children. Dividing by K , the number of C 's children, biases the system against proliferation of singleton classes.

COBWEB has many positive characteristics, many of which will be important to the design of LABYRINTH. Its well-defined performance task, tightly integrated with its learning component, allows evaluation of the concepts learned. Its use of probabilistic concepts and a single evaluation function allows more robust performance than earlier concept formation systems. Its simple local reorganization operators give it the partial ability to overcome misleading orders of training instance with minimal reprocessing of previous instances. However, COBWEB can only learn in domains in which there are a finite number of unstructured attributes; LABYRINTH builds on COBWEB to overcome this limitation.

2.7 Issues in Structural Learning

Table 1 summarizes the six systems we have just reviewed, as well as LABYRINTH, across five important characteristics. LABYRINTH is the only system that exhibits all five characteristics: it is an incremental, unsupervised learning method that acquires probabilistic concepts from relational data, using the heuristic of breaking the instance into components for classification. From our review, we can see the origins of these ideas. SPROUTER and related systems were the earliest to face the problem of learning in structured domains. These programs are supervised,

Table 1. Issues addressed by LABYRINTH and its predecessors.

System	Probabilistic	Incremental	Unsupervised	Relations	Components
SPROUTER		⊕		⊕	
CLUSTER/2			⊕		
CLUSTER/S			⊕	⊕	
Levinson		⊕	⊕	⊕	
MERGE		⊕	⊕	⊕	
COBWEB	⊕	⊕	⊕		⊕
LABYRINTH	⊕	⊕	⊕	⊕	⊕

and learn only a single conjunctive concept at a time, avoiding issues of memory organization but explicitly proposing algorithms to form generalizations from multiple objects described in a structured language.

We have seen that CLUSTER/2 differs from most earlier inductive learning algorithms in that it is *unsupervised*; it discovers object classes and characterizes these classes as well. Its successor, CLUSTER/S, incorporates the advances of CLUSTER/2, but uses a structured object and concept language. Unfortunately, both these systems are nonincremental, requiring all instances in order to generate classes. Levinson is among the first⁸ to propose a method for incrementally generating a memory organization containing structured concepts from unclassified instances. However, his system, in using what is in effect a complete matcher, fails to go beyond the data and to enlarge the deductive closure of its knowledge base. Wasserman's MERGE can be viewed abstractly as a version of Levinson's system that uses a partial matcher, and thus makes accurate classifications of previously unseen instances. In addition, Wasserman introduces the heuristic of decomposing structured objects into a tree, thus using one fundamental relation to organize memory and direct learning.

The basic classification mechanism and memory structure of COBWEB anticipates that of the current work. This system's use of probabilistic concepts gives it power to make more effective predictions than earlier

8. EPAM (Feigenbaum, 1963) also acquires concepts from hierarchically decomposed objects, but this system does not handle relations among components.

systems. In addition, COBWEB adopts prediction as a performance task for unsupervised learning systems. We have noted that COBWEB has many good characteristics, but is limited to attribute-value languages.

3. Representation and Organization in LABYRINTH

Having described earlier systems that address many of the issues faced by LABYRINTH, we are ready to discuss the current system at length. We will see that LABYRINTH has distinct ties to COBWEB, adopting its basic principle of probabilistic concepts organized in a disjoint hierarchy, and its divisive concept formation algorithm. However, the current system extends the representation language for objects and concepts. As we have seen, a central obstacle to learning in structured domains is that of characterizing structured concepts. LABYRINTH uses a representation for structured objects that reduces search by decomposing structured objects into a *partonomy* of components,⁹ supplemented by additional relations among those components. In this section, we describe the system's representation for objects and concepts, and how these concepts are organized in long-term memory.

3.1 Instances in LABYRINTH

Following Wasserman (1985), we argue that in many domains the instances passed to a concept learner are naturally decomposed by a *fundamental relation*. For example, Marr (1982) has argued that the visual system parses physical object descriptions into a partonomy organized by PART-OF relations. Similarly, McNamara, Hardy, and Hirtle (1989) have found that memory for large-scale spatial environments has a hierarchical component. Many forms of sequential data also can be represented as an ordered set of components; Rubin and Richards' (1985) work on elementary motion boundaries presents evidence that humans perceive motion in distinct segments that are invariant with respect to speed and viewpoint. For continuity, we use physical objects for our example instances, but we will discuss alternative domains and fundamental relations in Section 5.3.

9. Recall from Section 2.5 that we use the term *partonomy* for object hierarchies, to distinguish them from concept hierarchies (taxonomies).

LABYRINTH treats one relation as fundamental and structures both objects and concepts by that relation. A structured object is represented as a partonomy whose constituents are linked together by the fundamental relation. Each object can be augmented by non-fundamental relations whose arguments are components of that object. Consider again the domain shown in Figure 1. We represent the rightmost instance in Figure 1 as:

```
(Rightstack-2 (component1 (color blue) (shape odd))
  (component2 (color red) (shape circular))
  (component3 (color grey) (shape square)))
  ((Left-of component1 component3)))
  ((Left-of component1 component2)))
  ((on component3 component2)) .
```

Note that the PART-OF relation is implicit in this representation and is used to organize the object into a partonomy, as in MERGE.

We distinguish between two types of objects. *Primitive* objects are leaves of an instance partonomy. They are represented as ordered sets of attributes whose values are directly observable object features, as in COBWEB. For example, RIGHTSTACK-2 has three primitive components: COMPONENT₁, COMPONENT₂, and COMPONENT₃. *Structured* objects are represented as unordered sets of attributes (components) whose values can additionally be either primitive objects or other structured objects. Here, RIGHTSTACK-2 is a structured object with three attributes, each of which has a value that is a primitive object. In addition, this object has three associated binary relations, ON and two different instances of LEFT-OF, which are treated as additional attributes during classification. LABYRINTH treats components, non-fundamental relations, and descriptive features as different forms of "attributes". It exploits the isomorphism among them to classify both primitive and structured objects using a similar algorithm.

3.2 Concept Representation and Organization in LABYRINTH

Like COBWEB, LABYRINTH represents concepts by storing an associated set of attributes, their values, and associated conditional probabilities; it differs by the types of data that can be tied to those attributes. We define a *primitive* concept as a concept whose attributes have directly observable values. In contrast, a *structured* concept is one whose

attributes correspond to “components”.¹⁰ Because these components are themselves objects, a structured concept’s “attributes” have associated values that point at other concepts summarizing those objects. In this way, a single structured concept is defined in terms of other, possibly structured, concepts. A structured concept is thus stored not as a monolithic structure, but as many concepts distributed through memory, decomposed by the fundamental relation for that domain.

Because of this distributed representation, a single component concept can take part in several structured concepts. The concepts “pointed to” are themselves acquired by LABYRINTH, so that one can view the system as learning new terms; only primitive concepts are represented with values present in the original instance language. In addition, because all the concepts are changing over time in response to new information, LABYRINTH can manage concept drift with respect to both structured object concepts and component concepts.

Figure 2 shows a snapshot of LABYRINTH’s memory after it has incorporated three instances into memory: LEFTSTACK-1, LEFTSTACK-2, and RIGHTSTACK-1. The two singleton children of the LEFTSTACK concept, as well as the mixed root, are omitted for brevity. Each concept has been given a name for expository purposes, and each has an associated probability $P(N)$ with respect to its parent, along with a set of attributes. Each of these attributes in turn has a set of values and associated conditional probabilities. Note that for some of the concepts these associated values are in italics to indicate that they are the names of other concepts in memory. Thus, the hierarchy of Figure 2 contains thirteen primitive concepts, which represent stack components, and five structured concepts (of which three are shown), which represent stacks. Concepts for both are indexed in the same memory structure. The root concept thus summarizes both stacks and stack components, and is used only as an index for the hierarchy.

In addition to components, structured concepts can have arbitrary relations associated with them. LABYRINTH represents these relations as ternary-valued attributes, with associated conditional probabilities for each of the possible situations CONFIRMED, NEGATED, and MISSING. If a relation is not found in an object description, the system

10. Some concepts are “mixed”, in that they generalize both primitive and structured objects; for example, the root of the tree will always be mixed. Because primitive and structured objects never have values in common, these mixed concepts rarely appear below the first level of the tree in LABYRINTH runs.

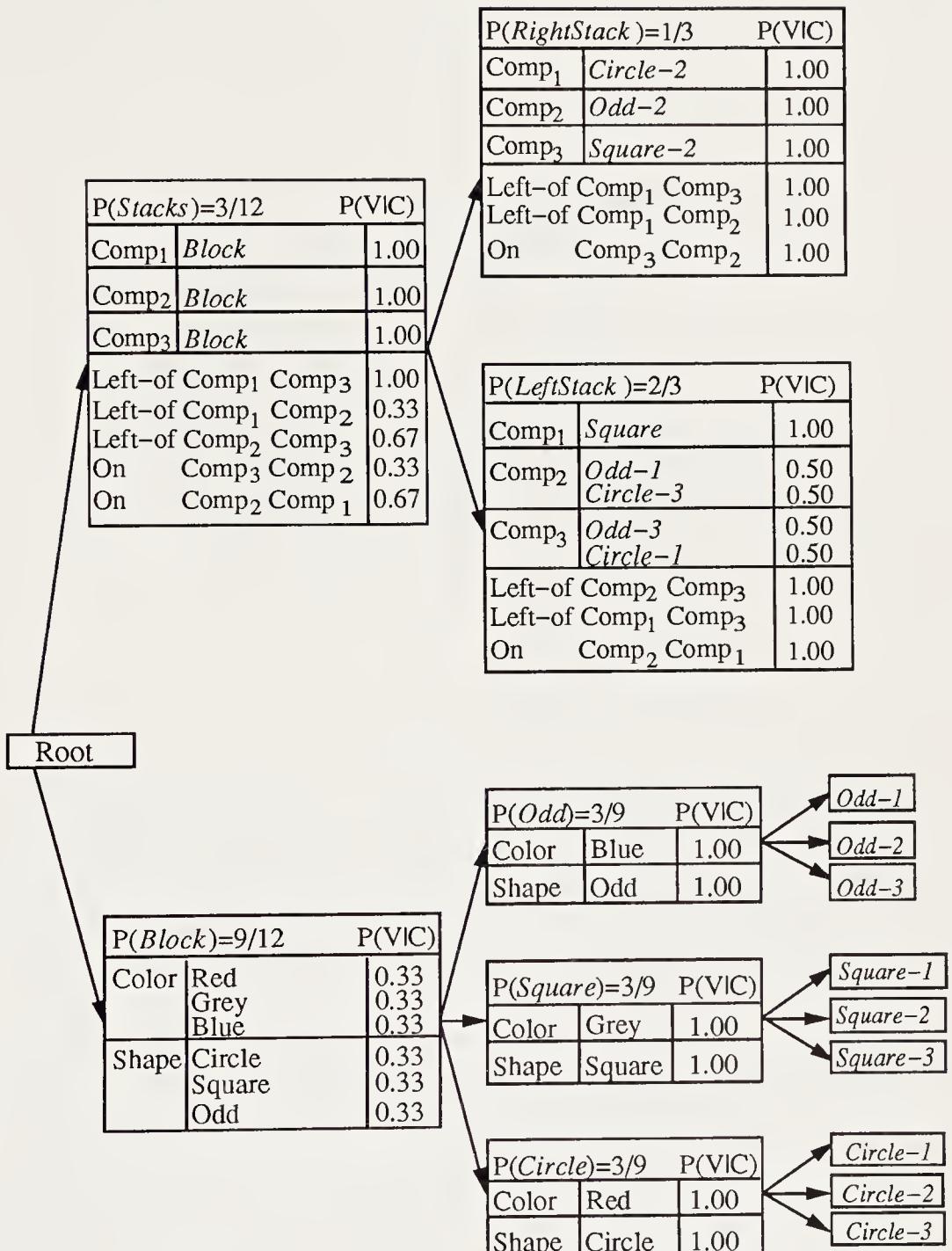


Figure 2. LABYRINTH's memory after processing three instances from Figure 1.

increments its MISSING probability. For brevity, we include only the CONFIRMED probability in our figures, but all three values are used for classification decisions. Note that there can be several different instantiations of the same relation at a concept; both (LEFT-OF COMPONENT₁ COMPONENT₃) and (LEFT-OF COMPONENT₁ COMPONENT₂) are associated with the RIGHTSTACK concept.

4. Classification and Learning in LABYRINTH

Having described LABYRINTH's memory structures, we can now describe how it classifies objects and updates its concept hierarchy. As in COBWEB, classification and learning are intertwined, with each instance being sorted through a concept hierarchy and altering that hierarchy in its passage. The system initializes its hierarchy to a single node based on the first instance. It then enters a loop of accepting new instances, classifying them and updating memory along the classification path. LABYRINTH differs from COBWEB in two important ways. It adds an outer loop to classify each component of a structured object. In addition, it introduces a new subroutine, COBWEB', to form predictive characterizations of structured concepts.

4.1 The LABYRINTH Algorithm

Table 2 shows the top-level LABYRINTH algorithm for classifying and learning with structured objects. To classify a single instance, the system uses a divide-and-conquer technique, breaking up the overall classification problem into a series of simpler classifications, one for each subtree of the instance partonomy. LABYRINTH processes structured objects in a "component-first" style, performing a complete postorder traversal of the partonomy. To classify a structured object in the partonomy, the system first classifies each of the object's components, returning the node in memory that the component most closely matches. LABYRINTH then "re-labels" the structured object, using each returned node as a label for a component. By performing this re-labeling operation, LABYRINTH reduces a structured object to a simple one with attributes and corresponding values; however, the values in this case are pointers to nodes in memory. The system then classifies this re-labeled object and recurses, until all the structured objects of the instance, including the instance itself, are classified.

Table 2. The basic LABYRINTH algorithm.

Input: OBJECT is a composite object, with substructure given.
 ROOT is the root node of the concept (is-a) hierarchy.
Side effects: Labels OBJECT and all its components with class names.

```

Procedure Labyrinth(OBJECT, ROOT)
  For each primitive component PRIM of composite object OBJECT,
    Let CONCEPT be Cobweb(PRIM, ROOT);
    Labyrinth'(OBJECT, PRIM, CONCEPT, ROOT).

Procedure Labyrinth'(OBJECT, COMPONENT, CONCEPT, ROOT)
  Label object COMPONENT as an instance of category CONCEPT.
  If COMPONENT is not the top-level object OBJECT,
    Then let COMPOSITE be the object that contains COMPONENT.
    If all components of COMPOSITE are labeled,
      Then let COMPOSITE-CONCEPT be Cobweb'(COMPOSITE, ROOT).
    Labyrinth'(OBJECT, COMPOSITE, COMPOSITE-CONCEPT, ROOT).
```

LABYRINTH uses two principal subroutines. The first of these is Fisher's COBWEB, which we have described in Section 2.6. LABYRINTH uses COBWEB to classify primitive components, treating it as a black box that returns the best match in memory to the object passed to it; we refer to this match as the *label* for that component. LABYRINTH relies on a second subroutine, COBWEB', to classify non-primitive objects. This routine is based on COBWEB; it uses the same evaluation function, basic control structure, and learning operators. However, COBWEB' incorporates additional mechanisms for finding the characterization of structured concepts.

We first illustrate LABYRINTH's processing on a simple two-level instance, RIGHTSTACK-2, from Figure 1. We then describe COBWEB' and its mechanisms for characterizing structured concepts.

4.2 LABYRINTH Classifying a Structured Object

We start with memory as in Figure 2, after three instances (two of LEFTSTACK and RIGHTSTACK-1) have been processed. To process the new instance RIGHTSTACK-2, LABYRINTH passes the description of COMPONENT₁ to COBWEB, which classifies and returns a label (the concept

ODD-4) for that component. The same procedure leads LABYRINTH to label COMPONENT₂ as a member of CIRCLE-4, and COMPONENT₃ as a member of SQUARE-4. So far, LABYRINTH has done no more than use COBWEB's existing mechanisms to "label" three primitive objects and update memory accordingly. However, whereas COBWEB stops there, LABYRINTH *uses* this information to classify the structured object. These labels are inserted into the structured object description, so that the instance now has the form:

```
(Rightstack-2 (component1 odd-4)
              (component2 circle-4)
              (component3 square-4)
              ((Left-of component1 component3)))
              ((Left-of component1 component2))
              ((on component3 component2)) .
```

LABYRINTH treats these labels from previous classifications as nominal values, enabling it to classify the structured object as though it were a primitive object (with the exceptions described in Section 4.3). In this case, LABYRINTH labels the structured object as a member of the structured concept RIGHTSTACK, resulting in the memory structure found in Figure 3. Here, we see that the concepts labeled BLOCKS, SQUARE, ODD, and CIRCLE have been updated in response to the components of RIGHTSTACK-2, and new leaves have been added to the concept tree for ODD-4, CIRCLE-4, and SQUARE-4. In addition, the stack itself has passed through the STACKS and RIGHTSTACK concepts, updating them accordingly. As in Figure 2, we omit the singleton concepts for the individual stacks, indexed by the LEFTSTACK and RIGHTSTACK concepts.

4.3 Integrating a Structured Object into a Concept

As we have seen in Section 2, the primary difficulty in learning from structured data is finding adequate *characterizations* of the concepts. LABYRINTH has a simplified characterization task because it learns from trees, not from the arbitrary graphs used by programs like SPROUTER and CLUSTER/S. However, LABYRINTH's subroutine COBWEB' still faces two extra searches to form characterizations. First, as we have noted, in many structural domains the components are *unordered*; in addition, whereas each object in Figure 1 has an identical number of components, some domains have objects with varying numbers of components. A

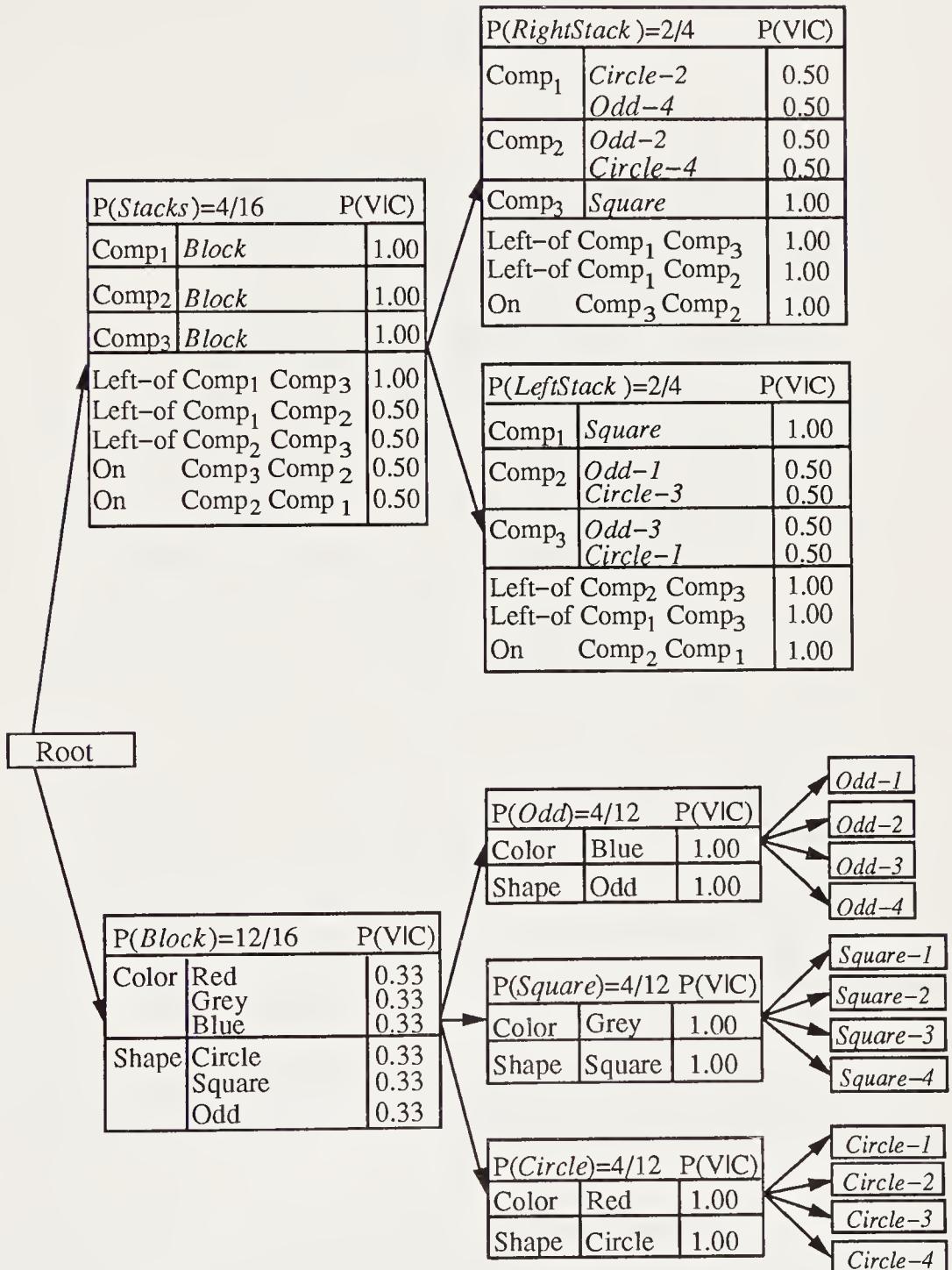


Figure 3. LABYRINTH's memory after processing four instances from Figure 1.

Table 3. Incorporating a structured object into a structured concept.

Variables: NODE is a node in the hierarchy.
INST is an unclassified structured object.

Incorporate(NODE, INST)

 Update the probability of category NODE.
 Let BINDINGS be all possible bindings between NODE and INST.
 For each possible set of bindings BIND in BINDINGS,
 For each relation REL in instance INST,
 Bind the arguments of REL according to BIND.
 If there is an equivalent relation N-REL in NODE,
 Then update the probability of N-REL;
 Else add REL to the characterization of NODE.
 For each attribute ATT in instance INST,
 Let I-VAL be the value of ATT.
 Let N-ATT be the corresponding attribute in NODE.
 For each value VAL of N-ATT,
 Update the probability of VAL given category NODE.
 Let VALS be the value list of N-ATT.
 Let the VALS be Attribute-Generalize (I-VAL, N-VALS, NIL).
 Evaluate the resulting node.
 Choose the best possible BIND and use it to store INST.

Attribute-Generalize(OBJ-VAL, NODE-VALS, CHECKED)

 For each VALUE in NODE-VALS,
 Let ANCESTOR be the common ancestor of OBJ-VAL and VALUE.
 Let REST be NODE-VALS with VALUE removed.
 If it is appropriate to replace OBJ-VAL and VALUE with ANCESTOR,
 Then call Attribute-Generalize(ANCESTOR, REST, CHECKED);
 Else call Attribute-Generalize(OBJ-VAL, REST, CHECKED).

characterization algorithm for structured concepts must thus determine a set of bindings between components in the object and those in the concept.

The second search arises from the nature of objects that COBWEB' processes. The routine classifies objects that are "re-labeled", in that their values are labels returned by earlier classifications. COBWEB' takes advantage of the hierarchical relationships between these labels to search for more predictive characterizations of structured objects. In this sec-

tion, we describe the approach COBWEB' takes to searching for the best characterization. Table 3 summarizes the function for incorporating an object into a concept in memory. This function replaces the simpler one used in COBWEB (see Gennari et al., 1989).

4.3.1 MATCHING COMPONENTS AND BINDING RELATIONS

Structured objects have unordered attributes; they lack *role* information that helps the learner match components from different objects. Whereas for primitive objects there is a unique maximal generalization for any pair of objects, structured objects require an extra matching process in order to determine the best match between the components and relations in the object and those in the concept. COBWEB' must bind the arguments of object relations to determine if they match with one of the concept relations. Once it has bound the arguments of each object relation, COBWEB' can update the correct probability (either CONFIRMED, NEGATED, or MISSING) of each concept relation based on the bound object. These concept relations are then treated as additional attributes of the concept in classification decisions.

We describe here an exhaustive algorithm to find the best mapping between a structured object and a structured concept.¹¹ COBWEB' matches a structured object with each concept in memory using a four-step process. First, it finds all mappings of components in the object to components in the concept. Second, for each mapping, the system rewrites the n -ary relations in the object by substituting each object component for its corresponding concept component. Third, COBWEB' compares the resulting instantiated relations to the relations in the concept description, treating each one as a Boolean attribute that may or may not match the concept. Finally, the system considers applying the attribute generalization operator described in Section 4.3.2 to each attribute. The resulting concept, with fully bound relations and attributes, is evaluated with a reduced form of category utility that evaluates the quality of a single concept:

$$\sum_i^{\text{Atts}} \sum_j^{\text{Values}} P(A_i = V_{ij} | C_k)^2 \quad . \quad (2)$$

11. For n components, this algorithm is $O(n!)$. Clearly, such a solution is impractical, and fails to take advantage of the simpler matching problem faced by COBWEB'. We discuss some less expensive solutions in Section 5.1.

This expression rewards matches that reinforce values already found in a node. The system selects the mapping that produces the node with the best score.

For example, when COBWEB' matches the fourth instance against the RIGHTSTACK concept from Figure 2, there are $3! = 6$ mappings between the three components of the instance and the three components of the concept. Some of these mappings reinforce the values in the components but not the associated relations in the concept; others reinforce the relations but not the components. LABYRINTH includes both components and relations as attributes in Equation 2. For the RIGHTSTACK concept, the system chooses an instantiation that reinforces all three relations found in the instance. This generates bindings for both the components and relation arguments in the object, producing the characterization found in the RIGHTSTACK concept of Figure 3.

4.3.2 ATTRIBUTE GENERALIZATION IN LABYRINTH

COBWEB' uses an additional mechanism to determine appropriate abstractions. For COBWEB, in which attributes take on only symbolic values, updating an attribute A_i after inspecting a new object is a simple matter of updating the correct conditional probability $P(A_i = V_{ij})$. However, in COBWEB', the values in the object are concepts stored elsewhere in the hierarchy (the results of previous classifications). In order to determine the best generalization between this structured object and an existing structured concept, COBWEB' uses the hierarchical relationships between the values in the object and those in the existing concept to determine the best values for the updated concept.

When incorporating an object into a concept, COBWEB' first adds the label from the object to the corresponding attribute A_i (as found by a step of the match process) in the concept, resulting in a set of values V_i . The system then evaluates whether to apply *attribute generalization* to the values on each attribute. We define attribute generalization as replacing a subset of the values V_i stored at attribute A_i with their common ancestor, resulting in a smaller set of values. Attribute generalization chooses between two possibilities. In the simple case, the operator can leave V_i intact, as would COBWEB. The COBWEB' routine also considers replacing a subset W_i of V_i with its common ancestor W_i^* . This results in a structured concept that can match more objects

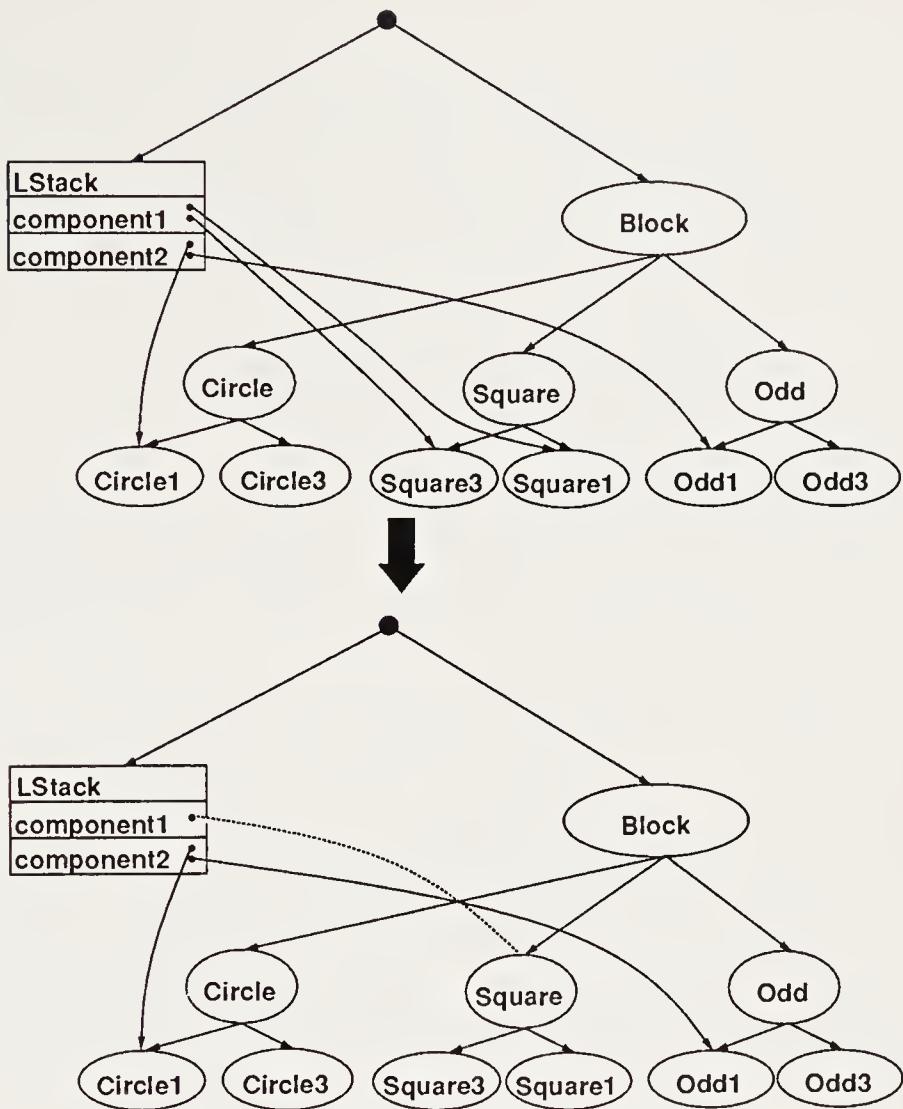


Figure 4. Generalizing the values of an attribute.

because of its more general values. For example, in Figure 3, we can see that for the attribute **COMPONENT₂** in the concept **RIGHTSTACK**, COBWEB' has simply added the object label **CIRCLE-4** to the previous label **ODD-2**. In contrast, note that **COMPONENT₃** has the single value **SQUARE**, the common ancestor of the two values **SQUARE-4** (from **RIGHTSTACK-2**) and **SQUARE-2** (from **RIGHTSTACK-1**). This reflects the fact that the two objects of **RIGHTSTACK** that LABYRINTH has classified to this point have a square stacked on another item. Figure 4 illustrates the application of this operator.

Determining when to apply the attribute generalization operator requires evaluating a tradeoff. Equation 2 favors a single value with high probability over several values with lower probability. Its application to attribute generalization would result in storing each attribute with a value V_i^* (the common ancestor of all the disjuncts) and probability of 1. While this would result in a higher score for Equation 2, it would result in less predictive power for the partition, because it would be difficult to discriminate such a node from other overgeneralized concepts. COBWEB' evaluates the tradeoff between forming concise characterizations with few values at one extreme, and over generalizing values so that concepts cannot be distinguished at the other extreme. It replaces a value set V_i with a shorter set W_i if doing so increases the information gain between a child C_k and its parent C . For an attribute A_i in C_k , COBWEB' replaces a set of values V_i with a new set of values W_i iff:

$$\sum_{l \in W_i} [P(A_i = W_{il} | C_k)^2 - P(A_i = W_{il} | C)^2] \geq \sum_{j \in V_i} [P(A_i = V_{ij} | C_k)^2 - P(A_i = V_{ij} | C)^2]. \quad (3)$$

The left side of this equation measures the information gain if the values are generalized at both the child C_k and the parent C . The right side measures the gain if the values are left as an internal disjunct.¹² COBWEB' considers all new value sets W_i that are strictly more general than the original set V_i , and stores with C_k the first set it finds for which Equation 3 holds.

In one sense, this attribute generalization process is simply an incremental approach to learning with *structured attributes* through climbing a “generalization tree”, as described by Michalski (1983), Mitchell, Utgoff, and Banerji (1983), and others. However, recall that LABYRINTH is constantly revising the structure of its concept hierarchy and introducing new symbols as it acquires new concepts. Since the descriptions of structured concepts refer to other concepts in the concept hierarchy, which LABYRINTH has acquired, the attribute generalization process operates over different knowledge structures at different points in the learning process. In effect, LABYRINTH is *dynamically changing* the representation used to describe its structured concepts.

12. Doug Fisher (personal communication) suggested the use of Equation 3.

5. Discussion

We believe that LABYRINTH constitutes a promising approach to concept learning in structured domains. However, the existing system has a number of limitations that we plan to remedy in future efforts. In addition, we must demonstrate the system on a variety of domains. We discuss these issues below, along with some related work.

5.1 LABYRINTH, Partial Matching, and Analogy

We have described an exhaustive algorithm to match components of an object to those of a concept in memory. Although this $O(n!)$ algorithm is guaranteed to find the optimal match according to Equation 2, more efficient alternatives exist. We plan to examine the Hungarian algorithm (Papademetriou & Steiglitz, 1982), a guaranteed matching algorithm that uses additional space to save partial solutions. Given a bipartite graph with $2n$ concepts, along with some function for evaluating the quality of a match, the Hungarian method finds the best match in $O(n^3)$ time. The algorithm works by creating an $n \times n$ cost matrix for all possible pairs of components and then solving an “ n rooks” problem over this matrix. We are also studying the use of a heuristic beam search (e.g., as used by SPROUTER) guided by Equation 2. We plan to use background knowledge about the data types of components to constrain this search as well.

LABYRINTH can be viewed as an approach to *partial matching* (Hayes-Roth, 1978). This task is usually defined as a comparison of two descriptions to identify their similarities, and is thus typically used in systems that use a specific-to-general search for hypotheses. SPROUTER (Hayes-Roth & McDermott, 1978) and its relatives (Winston, 1975; Vere, 1975) all use partial matchers as a principal subroutine in their search for generalizations. Some additional work has focused on partial matching outside the context of a concept learner. Kline (1981) emphasizes ordering the space of possible partial matches to reduce computation. In contrast, Watanabe and Rendell (1990) reduce computation by finding branches of the search tree that can be eliminated without loss of information by pruning redundant paths.

In determining the best match between a structured object and a structured concept, LABYRINTH is performing a crucial subtask in analogical reasoning. Falkenhainer, Forbus, and Gentner (1989) have de-

veloped the Structure-Mapping Engine (SME) for determining this best match, emphasizing structural integrity of the structured object over object similarity. Matching is constrained in structure mapping by higher-order relationships that are included as part of the instance. In contrast, LABYRINTH treats relations and object attributes as two contributors to the same evaluation function, rather than treating relations as preeminent. This should allow it to find some generalizations based on surface features that SME would not consider. However, the current version of LABYRINTH uses a far less constrained matcher than SME and will thus require far more computation in complex situations.

5.2 Using Context in Classifying Components

The current version of LABYRINTH takes a purely “component-first” or “context-free” approach to structured classification. An alternative approach would be to take context as the primary criterion, classifying a component only with respect to the *role* it plays in the greater whole. Clearly, a better approach would involve a combination of these two extremes (Fisher, 1986). We are investigating an extension to LABYRINTH in which the concept learner determines dynamically whether to classify an object based on its descriptive attributes only (as in the current system), or whether to consider its role as well.

The approach involves storing a *container* link with each object component. This link points from the component to the object of which it is a component (the partonomy parent) and is treated as an additional attribute for that object. Handa (1990) has explored one version of this approach. His system extends LABYRINTH to learn context-sensitive concepts by classifying each component twice: first to get a label used in classifying its container, and again after the container has been classified, using the container link as an additional attribute. In contrast, we plan to use ideas from Gennari's (1989) model of selective attention to determine dynamically whether to use the container attribute or others in classifying the object.

Another interesting extension to LABYRINTH involves forming concepts for the *roles* in its hierarchy. Consider the LEFTSTACK concept in Figure 3. The two values in COMPONENT₂ (ODD-1 and CIRCLE-1) are grouped by a simple kind of *functional* similarity; they play the same *role* in a structured concept. This grouping might occur in several structured concepts (although not in the STACK domain); however, the

current system has no means of recognizing this similarity across roles. We plan to investigate mechanisms through which LABYRINTH could recognize such shared structures.

5.3 Domains for LABYRINTH

We claim in Section 3.1 that a hierarchical organization is natural for many domains; we plan to demonstrate LABYRINTH's effectiveness in such domains. We have designed LABYRINTH as the fundamental memory organization scheme for ICARUS (Langley, Thompson, Gennari, Iba, & Allen, *in press*), an integrated architecture that treats storage and retrieval as central issues. Two components of ICARUS use structured object descriptions, and we plan to integrate each of these other systems with LABYRINTH.

DAEDALUS (Langley & Allen, 1990), the planning component of the architecture, uses plan knowledge stored in a probabilistic concept hierarchy to guide operator selection. We are currently integrating LABYRINTH into DAEDALUS; in this domain, a means-ends trace is represented as a hierarchical object linked by a SUBGOAL relation. In addition, we plan to apply LABYRINTH to the motor schemas formed by MÆANDER (Iba & Gennari, *this volume*), which represents limb motions as temporal sequences of joint positions and velocities. Each state corresponds to a LABYRINTH component, with joints serving as primitive objects.

6. Summary

In this chapter, we have described a system that learns concepts in structured domains. We have explained why the study of structured domains is important, and we have described six related systems that form a historical background for the current work. We have emphasized that all of the characteristics of LABYRINTH have been found in at least one of these systems. However, no single system shares all five features: the use of probabilistic concepts; an incremental algorithm; learning from unclassified instances; learning with objects that have relations; and using component structure to constrain matching.

LABYRINTH is an implemented system that extends COBWEB to structured domains. The system demonstrates a method for learning from hierarchically decomposed objects, using the results of component clas-

sifications to guide object classification. It learns in the presence of arbitrary relations in the object and concept language. LABYRINTH also introduces a new method for learning with hierarchically structured attributes. It demonstrates a form of representation change, in that it not only forms new terms as in previous concept formation systems, but also uses those terms in describing new concepts. In future work, we hope to establish LABYRINTH's applicability in a wide range of domains and to test its abilities in systematic experiments.

Acknowledgements

We thank Kathleen McKusick, Wayne Iba, Deepak Kulkarni, John Gennari, John Allen, and Doug Fisher for lengthy discussions that have influenced many of the ideas in this paper. All of the above and Sally Mouzon provided useful comments on an earlier draft.

References

- Allen, J. A., & Langley, P. (1990). Integrating memory and search in planning. *Proceedings of the 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 301–312). San Diego, CA: Morgan Kaufmann.
- Dietterich, T. G. (1986). Learning at the knowledge level. *Machine Learning*, 1, 287–316.
- Dietterich, T. G., & Michalski, R. S. (1981). Inductive learning of structural descriptions. *Artificial Intelligence*, 16, 257–294.
- Everitt, B. (1981). *Cluster analysis*. London: Heinemann.
- Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41, 1–63.
- Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought*. New York: McGraw-Hill.
- Fisher, D. (1986). A proposed method of conceptual clustering for structured and decomposable objects. In T. M. Mitchell, J. G. Carbonell, & R. S. Michalski (Eds.), *Machine learning: A guide to current research*. Boston: Kluwer.

- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139–172.
- Fisher, D. H., & Langley, P. (1990). The structure and formation of natural categories. In G. H. Bower (Ed.), *The psychology of learning and motivation: Advances in research and theory* (Vol. 26). Cambridge, MA: Academic Press.
- Gennari, J. H. (1989). Focused concept formation. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 379–382). Ithaca, NY: Morgan Kaufmann.
- Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11–61.
- Gluck, M., & Corter, J. (1985). Information, uncertainty and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum.
- Handa, K. (1990). CFIX: Concept formation by interaction of related objects. *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*. Nagoya, Japan.
- Hanson, S. J., & Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3, 343–372.
- Hayes-Roth, F., & McDermott, J. (1978). An interference matching technique for inducing abstractions. *Communications of the ACM*, 21, 401–410.
- Hayes-Roth, F. (1978). The role of partial and best matches in knowledge systems. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press.
- Hoff, W., Michalski, R. S., & Stepp, R. E. (1983). *A program for learning structural descriptions from examples* (Tech. Rep. No. UIUCDCS-F-83-904). Urbana: University of Illinois, Department of Computer Science.
- Kline, P. J. (1981). The superiority of relative criteria in partial matching and generalization. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence* (pp. 296–303). Vancouver, BC: Morgan Kaufmann.
- Kolodner, J. L. (1983). Reconstructive memory: A computer model. *Cognitive Science*, 7, 281–328.

- Langley, P., Thompson, K., Iba, W., Gennari, J. H., & Allen, J. A. (in press). An integrated cognitive architecture for autonomous agents. In W. Van De Velde (Ed.), *Representation and learning in autonomous agents*. Amsterdam: North Holland.
- Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2, 103-138.
- Levinson, R. A. (1985). *A self-organizing retrieval system for graphs*. Doctoral dissertation, Department of Computer Sciences, University of Texas, Austin.
- Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco: W. H. Freeman.
- McNamara, T. P., Hardy, J. K., & Hirtle, S. C. (1989). Subjective hierarchies in spatial memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15, 211-227.
- Mervis, C., & Rosch, E. (1981). Categorization of natural objects. *Annual Review of Psychology*, 32, 89-115.
- McKusick, K. B., & Thompson, K. (1990). *COBWEB/3: A portable implementation* (Tech. Rep. No. FIA-90-6-18-2). Moffett Field, CA: NASA Ames Research Center, Artificial Intelligence Research Branch.
- Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. Los Altos, CA: Morgan Kaufmann.
- Mitchell, T. M., Utgoff, P., & Banerji, R. B. (1983). Learning problem solving heuristics by experimentation. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Nordhausen, B., & Langley, P. (1990). An integrated approach to empirical discovery. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation*. San Mateo, CA: Morgan Kaufmann.

- Papademetriou, C., & Steiglitz, K. (1982). *Combinatorial optimization*. Englewood Cliffs, NJ: Prentice Hall.
- Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5, 239–266.
- Rubin, J. M., & Richards, W. A. (1985). *Boundaries of visual motion* (AI Memo 835). Cambridge, MA: Massachusetts Institute of Technology, Laboratory for Artificial Intelligence.
- Smith, E., & Medin, D. (1981). *Categories and concepts*. Cambridge, MA: Harvard University Press.
- Stepp, R. E. (1984). *Conjunctive conceptual clustering: A methodology and experimentation*. Doctoral dissertation, Department of Computer Science, University of Illinois, Urbana.
- Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering of structured objects: A goal-oriented approach. *Artificial Intelligence*, 28, 43–69.
- Vere, S. A. (1975). Induction of concepts in the predicate calculus. *Proceedings of the Fourth International Joint Conference on Artificial Intelligence* (pp. 281–287). Tbilisi, USSR: Morgan Kaufmann.
- Wasserman, K. (1985). *Unifying representation and generalization: Understanding hierarchically structured objects*. Doctoral dissertation, Department of Computer Science, Columbia University, New York.
- Watanabe, L., & Rendell, L. (1990). Effective generalization of relational descriptions. *Proceedings of the Eighth National Conference of the American Association for Artificial Intelligence* (pp. 875–881). Boston, MA: AAAI Press.
- Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.
- Wogulis, J., & Langley, P. (1989). Improving efficiency by learning intermediate concepts. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 657–662). Detroit, MI: Morgan Kaufmann.

Part II

Knowledge and Experience in Concept Formation

Theory-Guided Concept Formation

DOUG FISHER

MICHAEL PAZZANI

1. Introduction

Previous chapters in this volume concentrated on empirical or inductive approaches to concept formation. These models assume that observations are independent; the only link among them is implicit in the common language used for the intrinsic properties of observations, such as size and shape. Empirical concept formation is guided by similarities and differences among observations as expressed in this language.

However, psychological findings suggest that humans do not regard observations as independent, but as interacting in complex ways. The structural representations surveyed by Fisher and Pazzani (Chapter 1, this volume) and by Thompson and Langley (this volume) begin to address these issues, but they do not capture all the subtleties that are likely to confront a learner. For example, relationships among features may not be an explicit part of the environmental input to a system. Thus, a novice card player learning about straights (hands in which the cards can be placed in ascending order by rank) relies on internalized background knowledge about successive rank values (Vere, 1978). In addition, a learner must also be attuned to relationships across observations, not simply within observations. For example, `(successor(Ace, King))` is a relation that is internal to a card hand, whereas the expression `stronger(hand1, hand2)` defines a relation between card hands.

In sum, observations often cannot be treated in isolation. Rather, a rich set or ‘theory’ of interconnections link observations. These connections may vary in the granularity of relationships: features within or across observations can be connected, as can whole observations.

In addition, these relationships must often be inferred from a learner's background knowledge — they need not be an explicit part of the environmental input. This chapter briefly reviews the psychological literature that motivates this view, and examines computational models that have been proposed to deal with it.

2. Form and Function

Recall that studies by Medin, Wattenmaker, and Hampson (1986), Ahn and Medin (1989), and others found a strong unidimensional bias on the part of subjects during sorting. Fisher and Pazzani (Chapter 1, this volume) suggested extensions to these studies intended to encourage family resemblance clustering, but the experimenters (Medin et al., 1986; Ahn, 1990) were able to do this by suggesting the relevance of additional 'functional' features, together with theories about how intrinsic features and function were related. For example, they discussed 'hammering' prior to having subjects sort objects with features that would be relevant to this task: those with a majority of features like *graspable* and *flat head*. An explanation for the emergence of family resemblance sorting in this case is that by introducing the hammering descriptor, subjects were able to retain their bias for unidimensional concepts (i.e., those that one could use to hammer); this behavior was consistent with family resemblance sorting when attention was limited to the perceivable descriptors.

These studies suggest the strong role that *function* plays in human category development (e.g., Nelson, 1977). In particular, some evidence suggests that many categories arise because constituent members share similar functionality (e.g., a class of objects are used for hammering; a class of objects can be rolled), rather than shared *form* (*flat head*, *spherical*). Of course, similarity in structure is still a valuable heuristic guide in categorizing observations (Ross & Spalding, this volume), but at least in some cases this similarity is not the reason such classes arose originally. The importance of functionality becomes even more pronounced in categories for which there appears to be no structural similarity at all. In part, these correspond to what Barsalou (1983) defines as *ad hoc* categories, which arise spontaneously based on (roughly) shared function (e.g., things that will sell at a garage sale). Such categories tend to be transient and generated on demand,

in part because the lack of structural similarity makes categorization with respect to them difficult.

Some research on unsupervised machine learning has also recognized the importance of the form/function distinction (Nordhausen, 1986). In fact, an important rationale for distinguishing clustering and characterization in unsupervised systems is suggested by psychological findings that functional knowledge guides the search for clusterings, and structural information guides the search for characterizations. One example of ‘functional clustering’ is provided by GLAUBER (Langley, Zytkow, Simon, & Bradshaw, 1986), which is typically viewed as a model of scientific discovery. In this system, observations about molecular structures are represented by intrinsic features (e.g., taste), as well as by reaction relations among molecules. For instance, the interaction of different substances is represented by events like

```
(reacts inputs {HCl NaOH} outputs {NaCl}),  
(reacts inputs {HCl KOH} outputs {KCl}), and  
(reacts inputs {HNO3 KOH} outputs {KNO3}).
```

Upon seeing these and similar reactions, GLAUBER forms the clusters {HCl, HNO₃}, {NaOH, KOH}, and {NaCl, KCl, KNO₃}. Intuitively, these clusters represent the chemical classes of acids, alkalis, and salts, respectively. The system then uses these clusters to define the general relation (reacts inputs {acids alkalis} outputs {salts}).

GLAUBER has a number of limitations as a model of functional clustering, but it illustrates the promise of this paradigm: clusters of observations (e.g., molecular structures) bias the way that higher-level processes (e.g., chemical reactions) are characterized. Thompson and Langley (this volume), Reich and Fenves (this volume), Scott and Markovitch (this volume), and others (Fisher, 1986; Schlimmer, 1989; Handa, 1990) view this process as reformulating the description language that biases learning (Subramanian, 1989) and problem solving.

Much of the existing research on concept formation assumes that surface similarity indicates functional similarity and vice versa. However, we have seen that some psychologists hypothesize more than a heuristic connection between form and function. Rather, the assumption is that a ‘theory’ of a domain — a set of relations — serves to link the observable features with the function that these objects serve. We now turn to models that assume background knowledge and that require the learner to infer linkages among the observables.

3. Exploiting Background Knowledge

A landmark study by Chi, Feltovich, and Glaser (1981) required subjects to sort physics problems into categories of their own design. This unsupervised task revealed significant differences between the criteria that 'experts' (i.e., graduate students in physics) and 'novices' (i.e., undergraduates) used to create categories. Novices tended strongly to sort based on 'surface' features — those that were referred to in the problem statement (e.g., reference to an 'inclined plane' or 'friction'), while experts formed categories of problems that required the application of similar solution strategies (e.g., application of Newton's second law). In addition, experts generally required more time to produce an initial sort than novices. These findings suggest that expert subjects make an initial 'qualitative analysis' of a problem prior to classification. This preliminary analysis is responsible for inferring 'derived' (intermediate, deep) features from the surface features given in the problem statement.

The use of underlying principles or derived features to guide categorization by human subjects has been noted and qualified in many studies (Medin, 1989; Wisniewski & Medin, this volume). In addition to the Chi et al. studies, Faries and Reiser (1988) found that goal-directed, problem-solving situations accentuated reliance on inferred features. Importantly, inference of derived features need not be static and limited to a single preprocessing stage prior to classification. Rather, inference may interact continuously with classification (Seifert, 1989). Finally, despite the importance of inferred features, it would be a mistake to conclude that surface features are not useful. In fact, there are many cases when such features are exploited, even by experts (Ross, 1987; Ross & Spalding, this volume). Notably, humans appear to exploit surface features when they are well correlated with the derived features that are deemed most useful. For example, if one is calculating the time required to travel between two points, then the mention of 'boats' (and travel by water) versus 'cars' (and travel by land) indicates the relevance of a water current. To the extent that currents affect time, rate, and distance calculations, 'boats' will be exploited to classify the problem relative to past problems that were complicated by currents. Thus, as we have noted, surface features often provide a good heuristic guide to more fundamental, theory-based similarities, though this need not always be the case (e.g., travel by boat on a lake may not involve currents at all).

The importance of coupling inference and classification has not been lost on machine learning researchers. One paradigm that illustrates inference through background knowledge is *explanation-based learning* (Mitchell, Keller, & Kedar-Cabelli, 1986; DeJong & Mooney, 1986). In supervised settings, a learner searches a domain theory of inference rules in an attempt to link (i.e., explain) how an observation's surface features indicate membership in a teacher-provided target category. Recent research has explored the identification of derived features (i.e., a so-called boundary of operability) that should be inferred and exploited during classification (Braverman & Russell, 1988; Yoo & Fisher, this volume). In addition, Mooney (this volume) shows that the explanation-based paradigm need not be limited to supervised scenarios, just as empirical learning includes supervised and unsupervised representatives. Rather, objects that participate in similar ways across explanatory structures provide an opportunity for 'functional clustering' of the type described in the previous section — the difference is that relationships between observations are inferred from background knowledge rather than being an explicit part of the input as in GLAUBER.

Mooney (this volume) also points out that some explanation-based methods are easily adapted to facilitate attribute prediction. That is, an explanatory network may support inferences about many aspects of an observation, not simply its membership in a target category. For example, consider a voter who wants to categorize politicians based on their views along selected issues. A domain theory may contain very general inference rules such as

If a senator and the president are of the same party,
and the president is not a lame duck,
Then predict the senator's view equals that of the president,

or relatively specific rules like

If a senator is from the midwest,
Then predict that the senator supports farm aid.

In any case, a domain theory need not be directed at predictions along a single dimension, nor does this limitation apply to explanatory structures derived from such a theory.

To some extent, attribute prediction is also suggested by Stepp and Michalski (1986) and Mogensen (1987) in their nonincremental, conceptual clustering system, CLUSTER/CA. They suggest inference as a preprocessing step that fills in unknown values of observations prior to

clustering. CLUSTER/CA also includes a second, novel form of background knowledge known as a *goal-dependency network*, which links an agent's goals with observable features. For example, when a voter categorizes politicians, certain goals may play a greater role in clustering than others. If a high-level goal is to Limit-Soviet-Expansionism, then the voter's 'goal-dependency network' may indicate that issues such as MX-missile and Contra-aid should play a role in category construction, and a politician's views on issues such as Welfare-cuts should not. Thus, an agent's current goals will lead certain features to be used in clustering while others are not. This type of theory-driven attention promises to speed an agent's convergence on useful features for concept learning (Seifert, 1989), relative to purely empirical attention mechanisms surveyed by Fisher and Pazzani (Chapter 1, this volume).

In addition to explanation-based strategies, case-based approaches represent a second major theory-driven paradigm (e.g., Kolodner, 1987; Hammond, 1987). These models posit that many problems are best solved by appealing to specific problem-solving experiences (i.e., cases) of the past. An exemplar of this approach is a recent system by Shinn (1989) that incrementally 'clusters' problem-solving episodes (e.g., menu plans) into an abstraction hierarchy (somewhat like an EPAM discrimination net), labeling arcs of the tree by surface features (e.g., a lunch meal) and goal features (e.g., low-salt). New problems trigger the retrieval of similar past problems that are then modified to fit the particular constraints of the new situation. More generally, case-based approaches are concerned with a relatively tight coupling between inferencing and classification (Owens, 1990; Schank et al., 1990), like that noted in humans (Seifert, 1989). Many systems in this paradigm can also be cast as concept formation systems, just as some concept formation systems can be viewed as case-based systems (Langley, 1989).

A number of authors (Braverman & Wilensky, 1990; Porter, Bareiss, & Holte, 1990; Fisher, Yoo, & Yang, 1990) point to a merger of the two primary theory-driven paradigms: explanation-based and case-based learning. Briefly, case-based systems typically have some inductive capability that renders them less sensitive to imperfections in a domain theory than explanation-based systems (see Mitchell, Keller, & Kedar-Cabelli, 1986), while a domain theory, if available, provides a flexible avenue for problem solving when available cases do not provide sufficient experience on which to draw. An influential system in this regard is Pazzani's (1987) OCCAM, which incrementally clusters 'events'

(e.g., international sanctions) using a UNIMEM-like strategy (Lebowitz, 1987; Fisher & Pazzani, Chapter 1, this volume). It is worth describing OCCAM in some detail, since it illustrates several ways in which theory-driven and inductive mechanisms can interact within a concept formation system.

4. A Case Study in Theory-Driven Concept Formation

The OCCAM system clusters ‘events’ in an attempt to facilitate accurate predictions about future situations. For example, the system might be called upon to categorize international-sanction incidents like the following:

In 1983, Australia refused to sell uranium to France, unless France ceased nuclear testing in the South Pacific. South Africa sold France uranium at a premium and France continued nuclear testing.

In 1961, the Soviet Union refused to sell grain to Albania if Albania did not rescind economic ties with China. China sold Albania wheat at a below market price and Albania continued the ties with China.

The objective in this domain is to facilitate the system’s ability to accurately predict the outcome (i.e., successful or unsuccessful) of new sanction events.

Without background knowledge OCCAM uses UNIMEM’s approach to clustering, which is based on similarity over the surface features. Briefly, this strategy tends to group observations that share many features (e.g., selling country, buying country, selling price, decade of sale) and segregate observations that share few features. These clusters are characterized by conjoining the features common to *all* examples in the cluster.¹ Notice that clusters formed in this manner may include successful and unsuccessful incidents; as a consequence, the outcome feature is dropped from the cluster’s characterization, and the cluster will not be useful for purposes of predicting outcome.

Thus, consistent with psychological findings, reliance on surface similarity can be quite limiting in OCCAM. In response, the system incorporates explanation-based methods to increase the rate and accuracy

1. Early versions of UNIMEM did not allow exceptions to the ‘predictable’ features that make up a category’s characterization; concepts were strictly conjunctive. Later versions of UNIMEM overcame this limitation (Lebowitz, 1987), but OCCAM is based on earlier versions.

of learning by excluding irrelevant features that can lead to categories that hinder prediction of outcome (Seifert, 1989; Stepp & Michalski, 1986). Using background knowledge, OCCAM explains how the incident's surface features led to the success or failure of the sanction. In the first example above, South Africa provided a product (i.e., uranium) for economic reasons while China provided grain for political reasons. Thus, explanations for these events differ in some important ways. In general, each example with a distinct explanation creates a new cluster that is characterized by the features deemed relevant by explanation-based learning. As training proceeds, sanction incidents that succeed or fail for the same reason are grouped together. An alternative way of viewing this process is that instead of creating clusters of examples which are similar to each other in a space defined by the surface features, the clusters are created in an explanation space whose attributes are defined by the inference rules used to explain the examples (Mooney, this volume; Medin, 1989).²

However, explanation-based capabilities do not alleviate all problems. For example, a characterization that covers the two examples given above would not include the price of the commodity sold, since it is not present in the second example, even though the price is a critical contributor to success in the first example. Because categories are arranged hierarchically, this discriminating feature of the explanation structure is not used to initially guide classification at the most general levels of the hierarchy. More extreme examples of this effect occur when radically different explanations are clustered together by virtue of an identical outcome. This is undesirable because the different explanations basically define a disjunctive concept, but they are stored as a simple conjunction of one term (i.e., shared outcome).

In sum, OCCAM's conjunctive representational bias sometimes promotes clusters with irrelevant surface features, and overgeneralizes explanation structures, thus eliminating relevant features from its descriptions. Collectively, this can degrade predictive accuracy. We designed an experiment to analyze the extent of these tendencies in more detail. In each condition, training involved a series of 15 actual sanction incidents, and testing compared OCCAM's predictions to those of a Rand Corporation expert on a set of hypothetical incidents.

2. In the UNIMEM-like 'similarity' computation performed by OCCAM, a feature shared by a new event and cluster contributes a 1 to the match and a 0 otherwise. This occurs whether the features are surface or derived.

One condition was run in which all irrelevant surface features were deleted from the training examples and no domain theory was used. This removed any possible effect that irrelevant features could have during learning. Intuitively, we might expect that this would result in the same accuracy as that obtained by OCCAM with explanation-based learning, but accuracy was not as good. Thus, as Ahn (1990) found in experiments with human subjects, background knowledge does more than identify relevant features; it biases the way that relevant features are integrated into cohesive categories.

A second condition implemented a particular strategy of combining 'form' and 'function' in concept formation. In particular, OCCAM was run using background knowledge for clustering; thus explanation structures biased the formation of categories, in that derived features were included in the similarity computation that guided clustering. However, a 'knowledge-free' characterization strategy used all (relevant and irrelevant) surface features. Despite the biasing effect of background knowledge during clustering, the use of irrelevant features to characterize classes led to inaccurate classification of test cases. This highlights the variable contribution of surface similarity in guiding classification; in the sanctions domain it had little heuristic value.

Finally, when background knowledge was used to bias clustering (as above), and characterization occurred over only the relevant surface features, accuracy results were identical to those obtained in the standard OCCAM approach, in which features derived from background knowledge also participate in characterization. This last experimental condition highlights the dual role of derived features: they are valuable guides both in the formation of categories and in the characterization of these categories. Their presence in characterizations triggers inferencing during classification in OCCAM and related systems (e.g., Yoo & Fisher, this volume), and apparently in humans as well (Seifert, 1989).

5. Concluding Remarks

We have briefly surveyed psychological findings that motivate theory-driven approaches to concept formation, and examined some computational mechanisms that address this task. One important insight is that observations are often not independent. Rather, observations may interact, and these interactions influence clustering and characterization just

as intrinsic properties do. Often, form and function interact in subtle ways, each suggesting alternative categorizations.

For example, a traditional zoological partitioning of animals into mammals, birds, reptiles, amphibians, and fish is largely dictated by similarity over intrinsic properties, but interrelationships among animals (e.g., that lions eat zebras) dictate an ecological categorization that includes herbivores, carnivores, and omnivores. In fact, one often wants these orthogonal partitionings to coexist in a knowledge base, thus relating to Fisher and Pazzani's (Chapter 1, this volume) concerns with clumping and 'dag' organizations. Similar concerns from their discussions take on added complexity when form and function interact.

Finally, relations among observations are often not explicit in the environmental input. Instead, one must fill in the gaps from internal background knowledge. The explanation-based and case-based paradigms provide some guidance on how inference, categorization, and learning interact, though considerable research remains to be done before the field realizes a robust coupling of these processes within a single model.

Acknowledgements

The first author was supported by Grant NCC 2-645 from NASA Ames Research Center, and the second author was supported by Grant IRI-8908260 from the National Science Foundation. Discussions with Woo-Kyoung Ahn helped elucidate some issues, but she should not be held responsible for our interpretations of various data.

References

- Ahn, W., & Medin, D. L. (1989). A two-stage categorization model of family resemblance sorting. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 315-322). Ann Arbor, MI: Lawrence Erlbaum.
- Barsalou, L. W. (1983). Ad hoc categories. *Memory and Cognition*, 11, 211-227.
- Braverman, M. S., & Wilensky, R. (1990). Toward a unification of case-based reasoning and explanation-based learning. *Working Notes of the AAAI Spring Symposium on Case-Based Reasoning* (pp. 80-84). Palo Alto, CA: AAAI Press.

- Chi, M., Feltovich, P., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5, 121–152.
- Faries, J. M., & Reiser, B. J. (1988). Access and use of previous solutions in a problem-solving situation. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 433–439). Montreal, Quebec: Lawrence Erlbaum.
- Fisher, D. H. (1986). A proposed method of conceptual clustering for structured and decomposable objects. In T. M. Mitchell, J. G. Carbonell, & R. S. Michalski (Eds.), *Machine learning: A guide to current research*. Boston, MA: Kluwer.
- Fisher, D. H., & Chan, P. K. (1990). Statistical guidance in symbolic learning. *Annals of Mathematics and Artificial Intelligence*, 2, 135–148.
- Fisher, D., Yoo, J., & Yang, H. (1990). Case-based and abstraction-based reasoning. *Working Notes of the AAAI Spring Symposium on Case-Based Reasoning* (pp. 7–11). Palo Alto, CA: AAAI Press.
- Hammond, K. (1987). *Case-based planning: An integrated theory of planning, learning, and memory*. San Diego, CA: Academic Press.
- Handa, K. (1990). CFIX: Concept formation by interaction of related objects. *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*. Nagoya, Japan.
- Kolodner, J. L. (1983). Reconstructive memory: A computer model. *Cognitive Science*, 7, 281–328.
- Kolodner, J. L. (1987). Extending problem solver capabilities through case-based reasoning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 167–178). Irvine, CA: Morgan Kaufmann.
- Langley, P. (1989). Toward a unified science of machine learning. *Machine Learning*, 3, 253–259.
- Langley, P., Zytkow, J. M., Simon, H. A., & Bradshaw, G. L. (1986). The search for regularity: Four aspects of scientific discovery. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2, 103–138.

- Medin, D. L. (1989). Concepts and conceptual structure. *American Psychologist*, 44, 1469-1481.
- Medin, D. L., Wattenmaker, W. D., & Hampson, S. E. (1987). Family resemblance, conceptual cohesiveness, and category construction. *Cognitive Psychology*, 19, 242-279.
- Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Mogensen, B. (1987). *Goal-oriented conceptual clustering: The classification attribute approach* (Tech. Rep. No. UILU-ENG-87-2257). Urbana: University of Illinois, Department of Computer Science.
- Nelson, K. (1977). Some evidence for the cognitive primacy of categorization and its functional basis. *Merrill-Palmer Quarterly of Behavior and Development*, 19, 21-39.
- Nordhausen, B. (1986). Conceptual clustering using relational information. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 508-512). Philadelphia, PA: Morgan Kaufmann.
- Owens, C. (1990). Functional criteria for indices and labels. *Working Notes of the AAAI Spring Symposium on Case-Based Reasoning* (pp. 64-68). Palo Alto, CA: AAAI Press.
- Pazzani, M. (1987). Inducing causal and social theories: A prerequisite for explanation-based learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 230-241). Irvine, CA: Morgan Kaufmann.
- Porter, B., Bareiss, R., & Holte, R. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45, 229-263.
- Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5, 239-266.
- Ross, B. H. (1987). This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13, 629-639.
- Schank, R. C. (1982). *Dynamic memory*. Cambridge: Cambridge University Press.

- Schank, R., Brand, M., Burke, R., Domeshek, E., Edelson, D., Ferguson, W., Freed, M., Jona, M., Krulwich, B., Ohmaye, E., Osgood, R., & Pryor, L. (1990). Towards a general content theory of indices. *Working Notes of the AAAI Spring Symposium on Case-Based Reasoning* (pp. 36–40). Palo Alto, CA: AAAI Press.
- Schlümer, J. C. (1989). Refining representations to improve problem solving quality. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 457–460). Ithaca, NY: Morgan Kaufmann.
- Seifert, C. M. (1989). A retrieval model using feature selection. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 52–54). Ithaca, NY: Morgan Kaufmann.
- Shinn, H. (1989). *A unified approach to analogical reasoning*. Doctoral dissertation, School of Computer Science, Georgia Institute of Technology, Atlanta.
- Shrager, J. (1987). Theory change via view application in instructionless learning. *Machine Learning*, 2, 247–276.
- Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Stepp, R. E. (1987). Concepts in conceptual clustering. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 211–213). Milan, Italy: Morgan Kaufmann.
- Subramanian, D. (1989). Representational issues in machine learning. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 426–429). Ithaca, NY: Morgan Kaufmann.
- Vere, S. A. (1978). Inductive learning of relational productions. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press.
- Wisniewski, E. (1989). Learning from examples: The effect of different conceptual roles. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 980–986). Ann Arbor, MI: Lawrence Erlbaum.

CHAPTER 7

Explanation-Based Learning as Concept Formation

RAYMOND J. MOONEY

1. Introduction

At first glance, explanation-based learning (EBL) and concept formation appear to address quite different learning problems. The standard definition of explanation-based learning (Mitchell, Keller, & Kedar-Cabelli, 1986; DeJong & Mooney, 1986) presents it as a process of *operationalizing* an existing domain theory, i.e., making a theory more usable and efficient for the purpose of classification and problem solving. The standard definition of concept formation (Gennari, Langley, & Fisher, 1989) presents it as an incremental form of *conceptual clustering* (Michalski & Stepp, 1983), i.e., the formation of a conceptual hierarchy from unclassified examples.

The thesis of this paper is that, despite the apparent differences, EBL can be viewed as a type of theory-driven concept formation. Standard explanation-based methods can be seen as forming a new specialized concept from a single unclassified instance. This concept represents new knowledge that is not strictly within the deductive closure of the system's existing domain theory. Moreover, like the knowledge acquired by concept formation systems, the specialized concept learned by EBL can be used to predict missing information.

The remainder of this paper elaborates the view of EBL as concept formation. Section 2 provides some background material on explanation-based learning. Section 3 describes how standard explanation-based methods can be applied to the concept formation problem. Section 4

Table 1. The problem of explanation-based generalization.

Given:

- *Goal Concept:* A concept definition describing the concept to be learned. (We assume that this concept definition fails to satisfy the operability criterion.)
- *Training Example:* An example of the goal concept.
- *Domain Theory:* A set of rules and facts to be used in explaining how the training example is an example of the goal concept.
- *Operability Criterion:* A predicate over concept definitions, specifying the form in which the learned concept definition must be expressed. Generally, a concept definition that refers only to directly observable properties of the example is assumed to be operational.

Determine:

A generalization of the training example that is a sufficient concept definition for the goal concept and satisfies the operability criterion.

shows how portions of explanations can also be used to form new concepts. Section 5 reviews two existing explanation-based systems, GENESIS (Mooney, 1990a) and PURFORM (Kedar-Cabelli, 1988), and discusses how they can be viewed as concept formation systems. Section 6 shows how concepts formed using EBL are useful for predicting missing information. Section 7 discusses explanation-based concept formation from multiple examples. Finally, Section 8 presents some conclusions.

2. Background on Explanation-Based Learning

Explanation-based learning acquires an efficient concept definition from a single example by using existing background knowledge to explain the example and thereby focus on its important features (DeJong, 1988). The standard definition of the problem addressed by explanation-based learning is shown in Table 1 (Mitchell et al., 1986). The task involves using a single example and a domain theory to transform an abstract definition of a concept into an operational definition that is useful for

Table 2. A sample problem involving explanation-based generalization.

Goal Concept: $\text{Stable}(x) \wedge \text{Liftable}(x) \wedge \text{OpenVessel}(x) \rightarrow \text{Cup}(x)$

Training Example:

$\text{Owner}(\text{Obj1}, \text{Fred})$, $\text{Light}(\text{Obj1})$, $\text{Color}(\text{Obj1}, \text{Red})$, $\text{PartOf}(\text{H1}, \text{Obj1})$,
 $\text{Handle}(\text{H1})$, $\text{Bottom}(\text{B1})$, $\text{PartOf}(\text{B1}, \text{Obj1})$, $\text{Flat}(\text{B1})$,
 $\text{Concavity}(\text{C1})$, $\text{PartOf}(\text{C1}, \text{Obj1})$, $\text{UpwardPointing}(\text{C1})$

Domain Theory:

$\text{Bottom}(y) \wedge \text{PartOf}(y, x) \wedge \text{Flat}(y) \rightarrow \text{Stable}(x)$

$\text{Graspable}(x) \wedge \text{Light}(x) \rightarrow \text{Liftable}(x)$

$\text{Handle}(y) \wedge \text{PartOf}(y, x) \rightarrow \text{Graspable}(x)$

$\text{Concavity}(y) \wedge \text{PartOf}(y, x) \wedge \text{UpwardPointing}(y) \rightarrow \text{OpenVessel}(x)$

Operationality Criterion: Concept definition must be expressed in terms of structural features used to describe examples (e.g., Light, Handle, Flat).

efficient classification. Table 2 shows one of the standard examples of this task, learning a structural definition of a cup from a functional definition, a single example, and a domain theory relating form to function (Winston, Binford, Katz, & Lowry, 1983; Mitchell et al., 1986). The basic method for solving the explanation-based generalization problem consists of the following two steps:

- *Explain:* Construct an explanation using the domain theory that proves that the training example satisfies the definition of the goal concept.
- *Generalize:* Determine a set of sufficient conditions under which the explanation structure holds, stated in terms of the operationality criterion.

Standard theorem-proving methods, such as backward-chaining, are generally used to construct explanations. Several algorithms have been developed for correctly performing the generalization step (Mooney & Bennett, 1986; Kedar-Cabelli & McCarty, 1987). These procedures use unification to properly variabilize the explanation and thereby generalize the proof as far as possible while maintaining its correctness. For the ex-

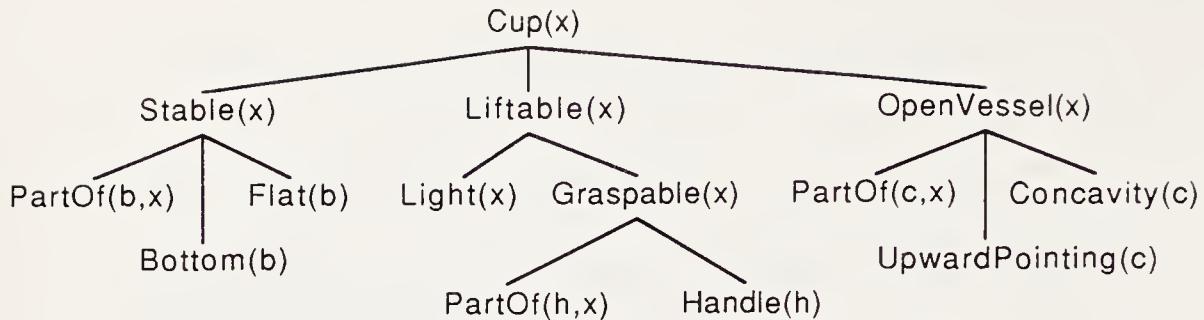


Figure 1. Generalized explanation for the cup example.

ample in Table 2, the proof and the generalization are straightforward. Figure 1 shows the generalized proof for this example.

An operational definition can be obtained by compiling the generalized proof into a new rule. The root of the proof tree forms the consequent of the new rule and the leaves form the antecedents. The compiled rule for the cup example can be stated as:

$$\begin{aligned}
 & \text{Bottom}(b) \wedge \text{PartOf}(b,x) \wedge \text{Flat}(b) \wedge \text{Handle}(h) \wedge \text{PartOf}(h,x) \\
 & \wedge \text{Light}(x) \wedge \text{Concavity}(c) \wedge \text{PartOf}(c,x) \wedge \text{UpwardPointing}(c) \\
 \rightarrow & \text{Cup}(x).
 \end{aligned}$$

Unlike the original definition, the learned definition satisfies the operability criterion since it only refers to observable, structural features. Therefore, instead of performing complicated inferencing, direct pattern matching can be used to classify future examples as cups. This can result in dramatic improvements in efficiency of classification, although the overhead of matching too many additional rules can eventually degrade performance (Minton, 1988). Similar mechanisms, such as *knowledge compilation* (Anderson, 1983) and *chunking* (Laird, Rosenbloom, & Newell, 1986), have been used to model psychological phenomena involving speedup, such as the power law of practice.

As another example of explanation-based learning, consider the following anecdote. Not long after moving from Illinois to Texas, I encountered an example of an interesting device called a *boot jack*. A rough sketch of a typical boot jack is shown in Figure 2. This device allows an urban cowboy to remove his boots easily and independently after a long, hard day at the office. A boot jack is used by stepping on the rear of the device with one foot, snugly inserting the heel of the other foot into the notch, and pulling one's leg upward to remove the boot. The

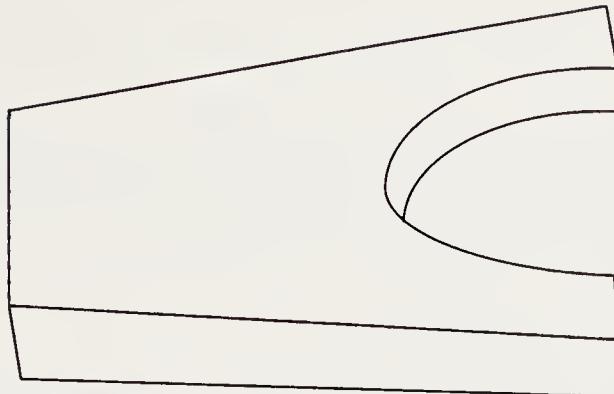


Figure 2. Sketch of a boot jack.

first example of a boot jack I encountered was made of brass and shaped like the head of a long-horned bull whose U-shaped horns formed the notch. After seeing this particular device used to remove a pair of boots, I immediately formed an accurate concept of a boot jack. I knew that certain properties such as shape, size, and rigidity were important but that the longhorn's image was superfluous and ornamental. A possible toy domain theory for this problem is shown in Table 3. Being an EBL researcher, I also immediately identified my acquisition of this concept as a type of explanation-based learning. Although the next example I encountered was very different, a simple piece of wood with the shape shown in Figure 2, I was able to quickly and accurately classify it as an example of the same concept.

Mitchell et al. (1986) present the following perspectives on explanation-based learning:

1. *Theory-guided generalization of a training example.* EBL can be seen as using a domain theory to generalize an example by focusing on its explainable aspects.
2. *Example-guided operationalization of a goal concept.* EBL can be seen as using an example to focus the operationalization of a domain theory to handle cases that actually occur in practice.
3. *Reformulation/operationalization/deduction from existing knowledge.* EBL can be seen as a purely deductive process in which information implicit in the domain theory is made explicit for the purpose of efficiency.

Table 3. Toy domain theory for a boot jack.

HoldBoot(x) \wedge CanHoldStationary(x) \wedge Rigid(x) \rightarrow BootJack(x)
HasNotch(x, n) \wedge FitHeel(n) \rightarrow HoldBoot(x)
Shape(x, U-shaped) \wedge width(x) < 4" \wedge width(x) > 3" \rightarrow
FitHeel(x)
MadeOf(x, metal) \rightarrow Rigid(x)
MadeOf(x, wood) \rightarrow Rigid(x)
HasPart(x, f) \wedge FootHold(f) \rightarrow CanHoldStationary(x)
Flat(x) \wedge width(x) > 4" \wedge length(x) > 4" \rightarrow FootHold(x)

Generally speaking, EBL as a means of improving problem-solving efficiency is by far the most common perspective. Most EBL systems, such as LEX/2 (Mitchell, 1983), LEAP (Mitchell, Mahadevan, & Steinberg, 1985), PROLEARN (Prieditis & Mostow, 1987), PRODIGY (Minton, 1989), EBL-LT (O'Rorke, 1987), ARMS (Segre, 1988), and BAGGER (Shavlik, 1990), concentrate on speeding up problem solving. Various other explanation-based methods, such as forming *macro-operators* (Fikes, Hart, & Nilsson, 1972), *chunking* (Laird et al., 1986), and *knowledge compilation* (Anderson, 1983), also focus on improving efficiency. Carbonell (1989) defines EBL as a type of *analytic learning* and states that "analytic methods focus on improving the efficiency of a system without sacrificing accuracy or generality, rather than extending its library of concept descriptions." Finally, Dietterich (1990) uses the term *speed-up learning* to refer to most work on explanation-based methods.

As a result, EBL has been criticized for its inability to learn at the *knowledge level* (Dietterich, 1986), which is defined as containing the deductive closure of a system's existing knowledge. Since standard EBL methods are purely deductive, they do not increase the system's deductive closure and therefore do not constitute knowledge-level learning. Since any real system operates with limited resources of time and space, improving efficiency of problem solving can frequently result in the ability to perform tasks that were previously impossible. Nevertheless, the inability of EBL to increase the theoretical capabilities of a system leads many to conclude that it does not constitute "true learning."

3. Explanation-Based Learning for Concept Formation

This section presents the concept formation view of explanation-based learning. Specifically, we show how standard EBL methods can be used in response to the problem of concept formation as defined by Fisher and Pazzani (Chapter 1, this volume). The argument proceeds in several steps, each step illustrating how EBL satisfies a different part of the definition of concept formation. Specifically, we need to show that: (1) explanation-based learning can be used to form an intensional definition of a new concept, (2) it can be used to form a hierarchy of such concepts that groups examples into clusters, (3) it can carry out these tasks using unclassified examples, and (4) it can perform these tasks incrementally.

3.1 Explanation-Based Learning as Theory-Based Concept Specialization

The first step is to show how EBL can be viewed as a form of theory-driven concept specialization. It is important to note that the operationalized definition produced by a standard explanation-based algorithm is only a sufficient rather than a necessary and sufficient definition of the goal concept. This is because the definition results from generalizing only a single proof of the goal concept and there may exist other proofs that would generate different operational definitions. In other words, if $Op(x)$ denotes the operational definition learned from generalizing a particular proof, then

$$Op(x) \rightarrow \text{Goal-Concept}(x)$$

is entailed by the domain theory, but

$$\text{Goal-Concept}(x) \rightarrow Op(x)$$

is not. Therefore, the operational definition actually represents a *specialization* of the original goal concept. For the cup example presented in Section 2, the operational definition represents the specialized concept of a “handled cup.” If the domain theory contained other ways of inferring graspability, such as

$$\text{Insulated}(x) \wedge \text{Width}(x, \text{ small}) \rightarrow \text{Graspable}(x),$$

then other specialized concepts, such as “insulated cups,” would be generated by generalizing different examples.

Table 4. The problem of theory-based concept specialization.

Given:

- A domain theory that defines a target concept, TC .
- A set of positive training examples of a concept C , where C is a specialization of TC .

Determine:

A correct definition of the specialized concept C .

Consequently, a new concept can be formed by defining it to be logically equivalent to the operational definition produced by EBL, i.e.,

$$\text{NewSpecializedConcept}(x) \equiv \text{Op}(x).$$

For example, one can define the concept G001 as

$$\forall x \exists b \exists c \exists h [G001(x) \equiv \text{Bottom}(b) \wedge \text{PartOf}(b,x) \wedge \\ \text{Flat}(b) \wedge \text{Handle}(h) \wedge \text{PartOf}(h,x) \wedge \text{Light}(x) \wedge \\ \text{Concavity}(c) \wedge \text{PartOf}(c,x) \wedge \text{UpwardPointing}(c)]^1$$

where G001 is a newly created predicate representing “handled cup.” It is important to note that this definition is not entailed by the existing domain theory and therefore constitutes knowledge-level learning. The newly introduced predicate did not even exist in the original theory. Therefore, standard EBL techniques can use a single example to form a definition for a new specialized concept. The new concept includes all examples of the original goal concept that have the same explanation as the training instance.

In other words, as illustrated by Flann and Dietterich (1989), standard explanation-based methods can be viewed as using a single example to solve the problem of *theory-based concept specialization* as presented in Table 4. This ability to form a specialized concept definition is an important aspect of viewing EBL as concept formation since such specialized concepts form the basis of a conceptual hierarchy.

1. In order to form a semantically correct definition, variables occurring in the operational definition but not in the goal concept must be existentially quantified.

3.2 Explanation-Based Learning as Hierarchical Conceptual Clustering

The standard EBL system adds new operational rules to the system's knowledge base and preferentially uses them to classify future examples in order to improve efficiency (Mooney, 1990a). This strategy automatically results in subsequent examples being classified as members or non-members of the new specialized concepts. Therefore, the new concepts naturally impose a grouping or clustering upon the observed instances. Instances that have the same explanation are grouped together in the same cluster. For example, handled cups are put into one cluster and insulated cups into another. In other words, EBL can be viewed as a form of conceptual clustering in which each explanation defines a cluster. Unlike clusters formed by standard empirical approaches, the clusters formed by EBL are based on functionally important differences, such as different ways of achieving liftability, rather than on arbitrary correlated sets of features.

Another requirement of concept formation is that the new concepts be arranged in an abstraction hierarchy. Since each operational definition acquired by EBL is a specialization of the goal concept, together they form a one-layer conceptual hierarchy. For example, explanation-based methods can be used to divide the original concept of "cup" into "handled cups" and "insulated cups." Multi-layer conceptual hierarchies can be formed by creating concepts for various pruned versions of explanations. Frequently, EBL systems prune branches from an example's explanation before generalizing it in order to increase the generality of the resulting definition (Mooney, 1990a). For example, imagine that the theory for cups includes the following rules:

```

Insulated(x) ∧ Width(x, small) → Graspable(x)
MadeOf(x, styrofoam) → Insulated(x)
MadeOf(x, ceramic) → Insulated(x)

```

Operational definitions for ceramic and styrofoam cups would be generated by generalizing explanations of specific examples. However, if the rule for *Insulated* were pruned from one of these explanations, a more general definition for "insulated cups" would be generated. Therefore,

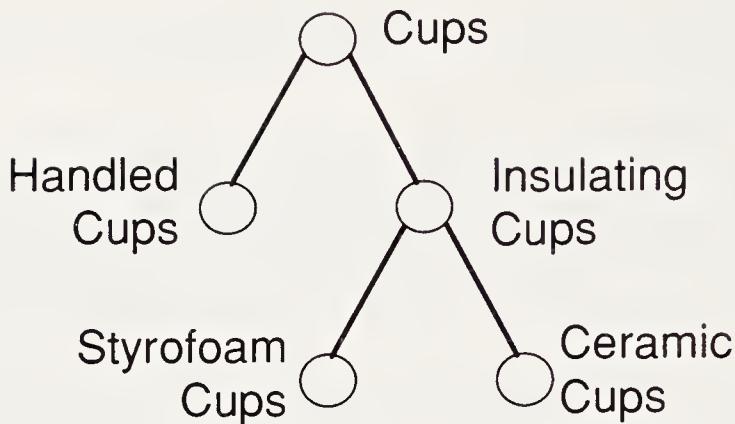


Figure 3. Sample conceptual hierarchy created by explanation-based learning.

by pruning and generalizing explanations of a number of cup examples, an EBL system could form the conceptual hierarchy shown in Figure 3.

3.3 Explanation-Based Learning as Unsupervised Learning

The definition of the explanation-based generalization problem presented in Table 1 specifies that the training instance is explicitly labeled as an example of the goal concept. This seems to violate the definition of concept formation, which specifies that training examples are unclassified. Although an example given to EBL is not labeled as a member of one of the specialized concepts that define clusters, its classification as an example of the more general goal concept implies that input examples are not completely unclassified.

Fortunately, classified examples are not actually required for explanation-based learning. If a system is given a domain theory that defines a number of different goal concepts, standard deductive techniques can be used to determine which of these goal concepts actually applies to an unclassified example. This can be done by either backward chaining from each potential goal concept or forward chaining from the instance description. The resulting explanations can then be generalized to form new specialized concepts.

Depending on the classification information available, EBL can actually operate in three different ways representing different levels of supervision:

1. *Example is classified as a member of the goal concept.* In this case, the training example is explicitly labeled as an example of a known goal concept and the domain theory is simply used to verify and explain this classification. This coincides with the standard definition of the explanation-based generalization problem given in Table 1. For example, a learner could be told directly that a boot jack is a device for removing one's boots and then explicitly told that a particular device is an example of a boot jack. EBL could then be used to operationalize the supplied functional definition.
2. *Example is classified as a member of an unknown concept.* In this case, the training example is labeled as an example of an unknown concept. The domain theory is used to independently deduce that it is a member of a known goal concept and the resulting specialized definition is interpreted as the definition of the unknown concept. This is similar to the definition of theory-based concept specialization shown in Table 4. For example, suppose a learner already has an abstract functional definition for a "boot remover" and is simply told that a particular instance is an example of an unknown concept called "boot jack." First, the learner independently constructs a proof explaining how the example meets the functional requirements of a "boot remover." Next, this proof is generalized to produce an operational description for a particular type of "boot remover." Finally, this operational description is inductively associated with the term "boot jack." Another example of this approach is presented in Section 5.1.
3. *Example is completely unclassified.* In this case, the training example is completely unclassified. The domain theory is used to independently deduce that it is a member of a known goal concept and the resulting specialized definition is used as the definition of a newly formed concept. For example, suppose a learner already has an abstract functional definition for a "boot remover" and independently observes an unclassified example of a boot jack. First, the learner independently constructs a proof explaining how the example meets the functional requirements of a "boot remover." Next, this proof is generalized to produce an operational description for a particular type of "boot remover." The learner now has a concept definition for "boot jack," although it is not associated with this particular English label.

The final approach corresponds to the environment assumed by most concept formation systems; however, all three methods result in the formation of new specialized concepts.

3.4 Explanation-Based Learning as Incremental Learning

The final requirement of concept formation is incremental processing of examples. Standard explanation-based methods naturally satisfy this constraint since they only consider one example at a time. Each new example is either classified by a learned rule as a member of a specialized concept created by an earlier example or explained by the underlying domain theory. If a new explanation is required, this explanation is generalized and compiled into a new operational rule that can be used to classify future examples of this subconcept more efficiently.

In summary, standard EBL methods are trivially incremental since they use a single example to create a new concept. The distinction between incremental and batch processing is only truly relevant when multiple examples are used to create concepts. The issue of incremental processing is more relevant to recently developed explanation-based methods that employ multiple examples (e.g., Flann & Dietterich, 1989).

4. Forming Concepts from Portions of Explanations

In addition to forming specialized concepts from complete explanations, explanation-based methods can be used to form new concepts from portions of explanations. Assume the example being explained is a structured object composed of a number of parts, each of which has various properties and relations to other parts. The explanation for this example may reference a particular part of the object along with a subset of its properties and relations. The constraints the explanation imposes on this part can be used to form a definition of a new concept. This is accomplished by choosing a particular variable referenced in the generalized explanation and collecting the literals in the operational definition that reference this variable.

For example, if we assume the cup theory contained the following rules in place of the Handle rule:

$$\begin{aligned} \text{PartOf}(y, x) \wedge \text{HasShape}(y, \text{semi-torus}) \wedge \text{FitFinger}(y) \wedge \\ \text{Rigid}(y) \rightarrow \text{Graspable}(x) \\ \text{diameter}(x) > 1'' \wedge \text{diameter}(x) < 3'' \rightarrow \text{FitFinger}(x) \end{aligned}$$

then the operational definition formed from generalizing a specific handled cup would be:

$$\begin{aligned} \text{Bottom}(b) \wedge \text{PartOf}(b, x) \wedge \text{Flat}(b) \wedge \text{PartOf}(h, x) \wedge \\ \text{HasShape}(h, \text{semi-torus}) \wedge \text{Rigid}(h) \wedge \text{diameter}(h) > 1'' \wedge \\ \text{diameter}(h) < 3'' \wedge \text{Light}(x) \wedge \text{Concavity}(c) \wedge \\ \text{PartOf}(c, x) \wedge \text{UpwardPointing}(c) \rightarrow \text{Cup}(x). \end{aligned}$$

The constraints that this definition imposes on the part denoted by the variable h could be used to form the following definition for a handle:

$$\forall h \exists x [G002(h) \equiv \text{PartOf}(h, x) \wedge \text{HasShape}(h, \text{semi-torus}) \wedge \\ \text{Rigid}(h) \wedge \text{diameter}(h) > 1'' \wedge \text{diameter}(h) < 3'']$$

This concept could then be used to classify parts of other examples with different explanations, such as the handle on a suitcase.

In general, forming a definition from only those literals that explicitly reference the chosen variable is overly restrictive. For example, if the chosen part is itself a structured object composed of a number of parts, literals that reference subparts of the chosen part should also be included. In other cases, the inclusion of all of the literals that reference the chosen variable is too liberal and heuristics are needed for eliminating some of them (see Section 5.2 for an example). Finally, heuristics are generally needed to select variables that will result in interesting concepts.

Unlike concepts learned from complete explanations, concepts acquired from portions of an explanation are generally not specializations of existing concepts. Consequently, such concepts are potentially more novel. However, there is also the potential to form large numbers of uninteresting concepts if the process is not tightly controlled. Both GENESIS and PURFORM, which are described in the following section, form new concepts from portions of an explanation and have specific heuristics for what portions to use. The acquisition of the concept “even number” by STABB (Utgoff, 1986) is another example of forming a concept from a portion of an explanation.

5. Explanation-Based Concept Formation Systems

A number of existing explanation-based systems are best viewed as performing concept formation rather than speed-up learning. This section discusses GENESIS and PURFORM as examples of existing systems that use explanation-based methods to form new concepts.

5.1 Concept Formation in GENESIS

GENESIS (Mooney, 1990a) is a text-understanding system that uses EBL to acquire plan schemata by explaining and generalizing specific plans presented in natural language narratives. Although the system can be viewed as operationalizing knowledge in order to improve the efficiency of plan recognition (Mooney, 1990b), it can also be viewed as performing concept formation. The specialized plan schemata acquired by GENESIS, such as kidnapping for ransom, arson for insurance, murder for inheritance, and entrapment for soliciting a prostitute, constitute legitimate new concepts. GENESIS forms these concepts from unclassified narratives and uses them to cluster similar events together in memory.

For example, GENESIS constructs an explanation for the following narrative and generalizes it into a schema for kidnapping for ransom.

Fred is Mary's father and is a millionaire. John approached Mary and pointed a gun at her. She was wearing blue jeans. He told her if she did not get in his car, then he would shoot her. 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 250,000 dollars at Trenos, then John would release Mary. Fred paid him the ransom and the kidnapper released Mary. Valerie is Fred's wife and he told her that someone had kidnapped Mary.

Before processing this narrative, the system has knowledge about bargaining, capturing and confining, threatening, and numerous other concepts. However, it does not have any knowledge about the concept of kidnapping for ransom nor any knowledge of the words "kidnap," "kidnapper," or "ransom." While processing this narrative, the system recognizes that John achieved a common goal of acquiring wealth and extracts an explanation for how this goal was satisfied. GENESIS gen-

eralizes this explanation and packages it into a new plan schema. The system's English summary² of the new schema is:

New Schema: (CaptureBargain ?x55 ?a34 ?b9 ?c4 ?r5 ?y5 ?l11)

?b9 is a person. ?c4 is a location. ?r5 is a room. ?c4 is in ?r5. ?x55 is a character. ?b9 is free. ?x55 captures ?b9 and locks him/her in ?r5. ?a34 is a character. ?x55 contacts ?a34 and tells it that ?b9 is ?x55's captive. ?y5 is a valuable. ?x55 wants to have ?y5 more than it wants ?b9 to be ?x55's captive. ?a34 has a positive relationship with ?b9. ?a34 has ?y5. ?x55 and ?a34 carry out a bargain in which ?x55 releases ?b9 and ?a34 gives ?x55 ?y5 at ?l11.

This schema is used to aid the understanding of subsequent kidnapping narratives and as an index to store kidnapping events together in memory.

In order to construct explanations for such narratives, GENESIS has several very general "goal concepts." Each represents a *thematic goal*, which is a top-level goal constituting a final explanation for a character's actions, such as acquiring wealth or food (Schank & Abelson, 1977). In the discussion of GENESIS presented in Mitchell et al. (1986), the goal concept representing a plan that achieves wealth is called WEALTH-ACQUISITION-ACTION-SEQUENCE. After processing each action, the system actively tries to infer that a thematic goal has been achieved. This allows the system to construct explanations for unclassified examples. When a thematic goal achievement is detected, its explanation is generalized and packaged into a new schema. The resulting schema is a specialization of one of GENESIS' initial goal concepts. After processing several narratives, the system has constructed a single-layer conceptual hierarchy like that shown in Figure 4. As in standard concept formation systems, specific instances are stored with the concepts of which they are a member.

In addition, GENESIS' ability to associate unknown words with its learned schemata (Mooney, 1987) can be viewed as applying EBL to an example classified as a member of an unknown concept, as discussed in Section 3.3. First, the system uses various heuristics to determine that an unknown word like "kidnap" actually refers to a schema that it just learned. The effect is similar to labeling the example with a category name like "kidnapping." Next, GENESIS constructs a definition for the

2. In this section, symbols with leading question marks denote pattern-matching variables. This is a common convention in LISP.

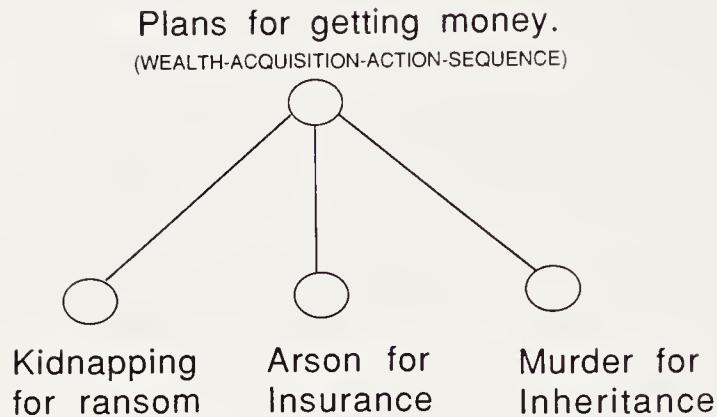


Figure 4. A conceptual hierarchy created by GENESIS.

unknown word so that the parser and text generator can use it as a label for the new concept. For example, after processing the above narrative, the system prints the comment

Unknown word '?x55 kidnap ?b9' refers to CaptureBargain.

GENESIS can use this association to label subsequent examples that were understood using the new schema. For example, the system uses its newly learned CaptureBargain schema to understand the novel narrative:

Ted is Alice's husband. He won 100,000 dollars in the lottery. Bob imprisoned Alice in his basement. Bob got 75,000 dollars and released Alice.

Afterwards, the system can summarize the event with the expression "Bob kidnapped Alice."

GENESIS' capacity for inductively associating unknown words with its learned schemata clearly establishes it as a system that uses explanation-based methods to carry out knowledge-level learning. After processing a single example, the system can correctly classify a number of events as examples or non-examples of kidnapping, a concept whose definition was not within the deductive closure of its original knowledge.

The ability to learn meanings for unknown words also illustrates the formation of concepts from portions of an explanation, as described in Section 4. Using methods similar to those employed by FOULUP (Granger, 1977), the system associates unknown words with particular

variables in a newly created schema. For example, the system makes two additional associations after processing the kidnapping narrative:

Unknown word 'ransom' refers to ?y5 in CaptureBargain.

Unknown word 'kidnapper' refers to ?x55 in CaptureBargain.

The role that a particular object plays in a plan can be considered as a concept in its own right. "Ransom" and "kidnapper" are clear examples of such concepts. Unlike those learned by most systems, these concepts are defined not by intrinsic properties of the object but by the events in which the object participates.³

The definitions for such words consist of two parts. First, GENESIS extracts properties of the variable required by the explanation. This requires a ransom to be a valuable object and a kidnapper to be a person. Second, the system adds a pointer from the unknown word to the new schema so that it is "reminded" of the schema whenever the word occurs in a subsequent narrative. For example, this link is crucial in letting the system classify another example as a kidnapping event:

Ted is Alice's husband. John took Alice into a room. Ted paid John the ransom and John released Alice.

Since GENESIS cannot correctly interpret this event without a concept for "ransom," it clearly illustrates the advantage of forming new concepts based on the role an object plays in a plan.

5.2 Concept Formation in PURFORM

PURFORM (Kedar-Cabelli, 1987, 1988) is another explanation-based system that is best viewed as involving concept formation rather than speed-up learning. The system forms new concepts for objects used as instruments in a plan. The general approach involves using portions of an explanation for a plan that impose constraints on a particular object.

Given a set of action definitions, an initial state, and a goal state, PURFORM uses a backward-chaining planner to determine a sequence of actions that achieves the goal. For example, given a goal state in which Robbie has ingested some hot tea, the system constructs the following plan:

3. Some inductive systems, such as Langley, Simon, Bradshaw, and Zytkow's (1987) GLAUBER, also learn concepts based on role information, but take a quite different approach from explanation-based methods.

```

pour(robbie, hot-tea, tea-urn1, mug1)
grasp(robbie, mug1)
pickup(robbie, mug1)
ingest(robbie, hot-tea, mug1)

```

The system then uses *goal regression* (Waldinger, 1977) to determine a set of generalized preconditions for the plan. The generalized preconditions⁴ computed for the above plan are:

```

open(From-Container)
can(pour(Grasper-Robot, Hot-Drink, From-Container, X))
can(contain(From-Container, Hot-Drink))
empty(X)
open(X)
can(contain(X, Hot-Drink))
grasper-empty(Grasper-Robot)
can(grasp(Grasper-Robot, X))
ungrasped(X)
can(pickup(Grasper-Robot, X))
on(X, Surface)
same-loc(Grasper-Robot, X, Location)
can(ingest(Grasper-Robot, Hot-Drink, X))

```

PURFORM then selects an object used in this plan and forms a concept from a subset of the generalized preconditions that reference this object. In order to take part in the new concept definition, a precondition must refer to an *intrinsic* property of the object. A property is intrinsic if and only if it cannot be altered by any known action, i.e., it does not appear on the add or delete lists of any of the system's operators. In the example, the variable X is chosen to form the basis of a new concept. Three of the constraints do not reference X and so are eliminated from consideration:

```

open(From-Container)
can(contain(From-Container, Hot-Drink))
grasper-empty(Grasper-Robot)

```

Moreover, four additional constraints are eliminated because they do not constitute intrinsic properties:

4. In this section, capitalized words are used to indicate variables. This is the convention in PROLOG.

```

empty(X)
ungrasped(X)
on(X, Surface)
same-loc(Grasper-Robot, X, Location)

```

The remaining constraints form the acquired definition for a cup:

```

can(pour(Grasper-Robot, Hot-Drink, From-Container, X)) ∧
open(X) ∧ can(contain(X, Hot-Drink)) ∧
can(grasp(Grasper-Robot, X)) ∧
can(pickup(Grasper-Robot, X)) ∧
can(ingest(Grasper-Robot, Hot-Drink, X)) → concept22(X)

```

Since the new concept is initially expressed in the functional terms used in the operators' preconditions, standard explanation-based generalization is used to transform it into an operational definition. The final operational definition for a cup can be stated as:

```

type(C, cylinder) ∧ has-part(X, C) ∧ open(X) ∧
material(X, ceramic) ∧ type(B, flat-bottom) ∧
type(B, sealed-bottom) ∧ has-part(X, B) ∧
type(H, handle) ∧ has-part(X, H) ∧ weight(X, W, oz) ∧
less(W, 16) → concept22(X)

```

Note that PURFORM executes two separate and slightly different EBL procedures, one to determine the generalized preconditions of the plan and another to operationalize the definition of the new concept. It should be possible to combine these two steps into the generalization of a single explanation. This would require a situation-calculus proof that combines the explanation for how the plan achieves the goal with the explanation for how the structural properties of any instruments fulfill the functional constraints specified in the plan's preconditions. Selecting intrinsic properties of the target object from the leaves of the generalized version of this proof would directly form an operational definition of the artifact. This rational reconstruction of PURFORM would directly correspond to forming a concept from portions of an explanation as described in Section 4.

As in GENESIS, the concept definitions acquired by PURFORM are not within the deductive closure of the system's initial knowledge. Since they are formed from portions of an explanation, they are not even specializations of existing concepts. However, unlike GENESIS' schema-role

concepts like "ransom," PURFORM's new concepts are defined strictly in terms of intrinsic properties of an object. This raises an interesting research question: When should a concept for an object be based on its intrinsic properties and when should it be based on the actions in which it participates?

Finally, it should be noted that my acquisition of the concept of a "boot jack" as presented in Section 2 is a real-world example of the type of learning used in PURFORM. What I actually observed was a plan for removing a pair of boots that used a boot jack as an instrument. I was not explicitly told a functional definition for a boot jack but rather extracted such a definition from my explanation of the sequence of observed actions. This example also illustrates the usefulness of forming concepts for artifacts used in an observed plan, in addition to learning from a plan generated by the learner.

6. Prediction in Explanation-Based Concept Formation

As noted by Fisher and Pazzani (Chapter 1, this volume), prediction of missing information is the performance task typically used to evaluate a concept formation system. If the concepts formed by explanation-based learning are worthwhile, they should also be useful for prediction. This section argues that EBL concepts are useful for prediction and presents GENESIS as an example of an explanation-based system that uses learned concepts to fill in missing information.

The standard approach to prediction in concept formation systems is relatively straightforward. First, one classifies an incompletely described instance as a member of a previously learned concept. Next, one uses the norms of previously observed examples and/or the concept definition itself to fill in missing information. Concepts formed using EBL can be used to support prediction in a similar manner. For example, GENESIS uses script-based understanding methods (Schank & Abelson, 1977; Schank & Riesbeck, 1981) to infer missing information in narratives. Since text understanding requires "reading between the lines," the ability to use plan schemata to support such inferencing is critical. GENESIS uses a relatively complex schema-selection process to classify a new narrative as an example of a known schema. It then instantiates the selected schema, filling in any actions or facts not explicitly mentioned in the input. As new schemata are learned, they

are naturally used to support prediction. For example, after learning a schema for kidnapping, the system uses it to understand the following narrative:

Input: Ted is Alice's husband. He won 100,000 dollars in the lottery. Bob imprisoned Alice in his basement. Bob got 75,000 dollars and released Alice.

Thematic goal achieved: Ted is happy that Ted has the \$100,000.

Thematic goal achieved: Bob is happy that Bob has the \$75,000.

Ready for questions:

>Who gave Bob the money?

Ted gave Bob the \$75,000.

>Why did Ted give him the money?

Because Ted believed that if Ted gave Bob the \$75,000, then Bob would release Alice and because Ted wanted Alice to be free more than he wanted to have the \$75,000.

The schema's predictions provide answers to some critical questions shown in the trace. Therefore, GENESIS nicely illustrates that explanation-based concepts are also useful for predicting missing information. The schemata it acquires by explanation-based learning are very useful for script-based understanding of incompletely described events.

The classification problem GENESIS faces is very difficult and reveals some important limitations of existing concept formation systems. Most concept formation systems use a representation language based on attribute-value pairs. Such a simple representation language makes classification of an incompletely described instance relatively easy. Similarity metrics such as the percentage of matching features or Euclidian distance can be used to classify an incomplete instance into the most appropriate concept.⁵ Once an instance is classified, the values of any missing features can be predicted using the majority or average value of other instances in the category. It is important to note that this approach makes use of the *single-representation trick* (Dietterich, London, Clarkson, & Dromney, 1982), that is, it assumes concepts and instances to be represented in the same language.

5. COBWEB and its descendants use a more complex metric called *category utility* to categorize new instances.

The relatively long history of work in text understanding has demonstrated that classifying incomplete instances into complex concepts represented in a rich structural language is a very difficult task. The problems of indexing and matching are more difficult because one must deal with variable bindings and relational predicates. In addition, the single representation trick is inadequate, since complicated inferencing is needed to connect the details of the example with the defining features of the schema. In text understanding, the problem is generally referred to as the *frame selection problem* (Charniak, 1978; Charniak, 1982). For example, imagine that the task is to classify an event as an example or non-example of the concept "murder for inheritance." Consider how the various pieces of information affect classification of the following instance:

Mary was murdered. John was Mary's husband. Mary was a major real-estate tycoon. Mary was about to divorce John. Mary and John had a prenuptial agreement that left John with virtually nothing. John had large gambling debts. The local mob boss had threatened to have John killed unless these debts were paid soon.

Obviously it is impractical for our concept for "murder for inheritance" to have predefined features such as VICTIM-IS-REAL-ESTATE-TYCOON, VICTIM-IS-DIVORCING-MURDERER, MURDERER-HAS-A-POOR-PRENUPTIAL-AGREEMENT-WITH-VICTIM, and MURDERER-HAS-LARGE-DEBTS-TO-IMPATIENT-MOB-BOSS. A great deal of additional inferring must be done to realize the connection between these facts and the definition of "murder for inheritance." Performing classification accurately and efficiently in such situations is clearly a very difficult problem that is not adequately addressed by existing concept formation systems.

7. Explanation-Based Concept Formation from Multiple Examples

Standard explanation-based learning methods form concepts from a single example, but there has been some research on using explanatory knowledge to form concepts from multiple examples. This work usually involves combining empirical and analytical methods. Here I briefly review several projects in this area to clarify how one might integrate explanation-based and empirical approaches to concept formation.

The CLUSTER/CA system (Stepp, 1986; Mogensen, 1987) is a nonincremental conceptual clustering system that uses a data structure called a *goal-dependency network* to focus attention on causally relevant attributes. This network encodes relationships between high-level goals and the features used to describe instances. The heuristic search procedure prefers to form conceptual clusters which are defined by features that are related to known goals. The goal-dependency network represents a relatively weak propositional domain theory that can be used to determine relevant features. However, it does not provide detailed explanations for examples like those typically used in EBL.

OCCAM (Pazzani, 1990) is a concept formation system that combines explanation-based and empirical techniques. Like GENESIS, OCCAM operates in the domain of narrative text understanding. If an example can be explained, its explanation forms the basis of a new specialized concept in a manner similar to GENESIS. If an example cannot be explained but is similar to a previous example, the common features of the two examples are used to form a new schema. This empirical learning process is modeled after concept formation in UNIMEM (Lebowitz, 1987). In OCCAM, empirically learned concepts can be used to help explain subsequent examples, and specializations of EBL concepts can be empirically learned by noticing additional similarities among their examples. However, explanations actually constructed by the system are not used in the empirical concept formation process.

Yoo and Fisher (this volume) have proposed an unsupervised version of ‘induction over explanations’ (IOE) (Flann and Dietterich, 1989), which I will refer to as “clustering over explanations.” This approach forms clusters of explanations that support accurate prediction. A conceptual description is formed by using the IOE procedure to compute the most specific explanation structure that matches all of the explanations in the cluster. The resulting conceptual hierarchy is used to index explanations for efficient retrieval and reuse. One problem with Yoo and Fisher’s method is that it does not take into account similarities among attributes not present in the explanations.

8. Conclusions

This paper has shown that standard explanation-based learning techniques can be used to address the concept formation problem. In fact, several EBL systems, such as GENESIS and PURFORM, focus more on

forming new concepts than on operationalizing or compiling existing knowledge. Also, when viewed as concept formation, EBL is a form of knowledge-level learning since the resulting concept definitions are not within the deductive closure of the initial domain theory. Finally, concepts learned using EBL can be used to fill in missing information in incompletely described instances, and thus are useful for prediction.

Acknowledgements

I would like to thank Pat Langley for initially suggesting that I present a talk and write a chapter on this topic. I would also like to thank Doug Fisher, Mike Pazzani, and Pat Langley for their helpful comments on an initial version of this paper. While preparing this paper, the author was supported by Grant NCC 2-629 from NASA Ames Research Center.

References

- Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.
- Carbonell, J. G. (1989). Paradigms for machine learning. *Artificial Intelligence*, 40, 1–9.
- Charniak, E. (1978). With spoon in hand this must be the eating frame. In *Proceedings of the Second Conference on Theoretical Issues in Natural Language Processing* (pp. 187–193). Urbana, IL: ACL.
- Charniak, E. (1982). Context recognition in language comprehension. In W. G. Lehnert & M. H. Ringle (Eds.), *Strategies for natural language processing*. Hillsdale, NJ: Lawrence Erlbaum.
- DeJong, G. F. (1988). Explanation-based learning. In H. Shrobe (Ed.), *Exploring artificial intelligence*. San Mateo, CA: Morgan Kaufmann.
- DeJong, G. F., & Mooney, R. J. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1, 145–176.
- Dietterich, T. G. (1986). Learning at the knowledge level. *Machine Learning*, 1, 287–316.
- Dietterich, T. G. (1990). Machine learning. In J. F. Traub, B. J. Grosz, B. W. Lampson, & N. J. Nilsson (Eds.), *Annual review of computer science* (Vol. 4). Palo Alto, CA: Annual Reviews.

- Dietterich, T. G., London, B., Clarkson, K., & Dromney, G. (1982). Learning and inductive inference. In P. R. Cohen & E. A. Feigenbaum (Eds.), *Handbook of artificial intelligence* (Vol. 3). San Mateo, CA: Morgan Kaufmann.
- Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence*, 3, 251–288.
- Flann, N. S., & Dietterich, T. G. (1989). A study of explanation-based methods for inductive learning. *Machine Learning*, 4, 187–226.
- Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11–61.
- Granger, R. H. (1977). FOULUP: A program that figures out meanings of words from context. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 172–178). Cambridge, MA: Morgan Kaufmann.
- Kedar-Cabelli, S. (1987). Formulating concepts according to purpose. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 477–481). Seattle, WA: Morgan Kaufmann.
- Kedar-Cabelli, S. (1988). *Formulating concepts and analogies according to purpose*. Doctoral dissertation, Department of Computer Science, Rutgers University, New Brunswick, NJ.
- Kedar-Cabelli, S., & McCarty, L. T. (1987). Explanation-based generalization as resolution theorem proving. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 383–389). Irvine, CA: Morgan Kaufmann.
- Laird, J. E., Rosenbloom, P., & Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1, 11–46.
- Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2, 103–138.
- Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Minton, S. (1988). Quantitative results concerning the utility of explanation-based learning. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 564–569). St. Paul, MN: Morgan Kaufmann.

- Minton, S. (1989). *Learning search control knowledge: An explanation-based approach*. Hingham, MA: Kluwer.
- Mitchell, T. M. (1983). Learning and problem solving. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 1139–1151). Karlsruhe, W. Germany: Morgan Kaufmann.
- Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1, 47–80.
- Mitchell, T. M., Mahadevan, S., & Steinberg, L. I. (1985). LEAP: A learning apprentice for VLSI design. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 573–580). Los Angeles, CA: Morgan Kaufmann.
- Mogensen, B. N. (1987). *Goal-oriented conceptual clustering: The classifying attribute approach*. Masters thesis, Department of Computer Science, University of Illinois, Urbana.
- Mooney, R. J. (1987). Integrated learning of words and their underlying concepts. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 974–978). Seattle, WA: Lawrence Erlbaum.
- Mooney, R. J. (1990a). *A general explanation-based learning mechanism and its application to narrative understanding*. San Mateo, CA: Morgan Kaufmann.
- Mooney, R. J. (1990b). Learning plan schemata from observation: Explanation-based learning for plan recognition. *Cognitive Science*, 14, 483–509.
- Mooney, R. J., & Bennett, S. W. (1986). A domain independent explanation-based generalizer. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 551–555). Philadelphia, PA: Morgan Kaufmann.
- O'Rorke, P. V. (1987). LT revisited: Experimental results of applying explanation-based learning to the logic of Principia Mathematica. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 148–159). Irvine, CA: Morgan Kaufmann.
- Pazzani, M. J. (1990). *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods*. Hillsdale, NJ: Lawrence Erlbaum.

- Prieditis, A. E., & Mostow, J. (1987). PROLEARN: Towards a PROLOG interpreter that learns. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 494–498). Seattle, WA: Morgan Kaufmann.
- Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ: Lawrence Erlbaum.
- Schank, R. C., & Riesbeck, C. (1981). *Inside computer understanding*. Hillsdale, NJ: Lawrence Erlbaum.
- Segre, A. M. (1988). *Machine learning of robot assembly plans*. Boston, MA: Kluwer.
- Shavlik, J. W. (1990). *Generalizing the structure of explanations in explanation-based learning*. San Mateo, CA: Morgan Kaufmann.
- Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Utgoff, P. E. (1986). Shift of bias for inductive concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Waldinger, R. (1977). Achieving several goals simultaneously. In E. Elcock & D. Michie (Eds.), *Machine intelligence* (Vol. 8). London: Ellis Horwood.
- Winston, P. H., Binford, T. O., Katz, B., & Lowry, M. (1983). Learning physical descriptions from functional definitions, examples, and precedents. *Proceedings of the Third National Conference on Artificial Intelligence* (pp. 433–439). Washington, DC: Morgan Kaufmann.
- Yoo, J. P., & Fisher, D. H. (1989). Conceptual clustering of explanations. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 8–10). Ithaca, NY: Morgan Kaufmann.

CHAPTER 8

Some Influences of Instance Comparisons on Concept Formation

BRIAN H. ROSS

THOMAS L. SPALDING

1. Introduction

Throughout our lives we have new experiences and continue to learn about events and objects. Each new experience requires us to use our knowledge from past experiences to understand and make predictions from the current experience. In addition, each new experience has the potential to modify the knowledge that we will use in understanding and predicting from still later experiences. Thus, how we form *concepts* — the knowledge about these experienced events and objects — is a central issue for understanding or constructing any intelligent system.

Recently, as this symposium and book attest, much work has been conducted in the area of concept formation. Although the research has focussed on computational approaches, some of the investigators have also been concerned with keeping models consistent with a number of psychological results (e.g., Fisher, 1987; Fisher & Langley, 1991; Gennari, Langley, & Fisher, 1989). This reasonable practical approach, trying to accomplish a goal by making use of one method that we know does accomplish it, also has the potential for enlightening our conceptions of human cognition.

Before we proceed, it is important to consider how this work fits in with much of the recent interest in machine learning and cognitive science on the role of theories, explanations, or other knowledge effects

(e.g., DeJong & Mooney, 1986; Murphy & Medin, 1985; Murphy & Wisniewski, 1989; Pazzani & Schulenburg, 1989). Clearly, our ability to learn from instances and use this knowledge involves both the application of such prior knowledge and the learning from similarities among instances (i.e, inductive learning). In this chapter (as in most of the chapters in this book), our concern is mainly with the inductive effects, although we do consider how knowledge effects may interact with this type of learning. We believe that such a focus is important for a number of reasons. First, people often learn by comparing instances and are able to do so easily. Second, some of the uses of other knowledge will often not be applicable in early learning, especially if the domain theory is weak or incomplete. Third, detecting similarities may be an important precursor to the application of more top-down knowledge (e.g., Billman & Heit, 1988; Hanson & Bauer, 1989). As an example, similarities can be used to generate hypotheses and focus the explanation component (e.g., Lebowitz, 1986).¹ Fourth, inductive and knowledge-based techniques may be useful for different learning situations (e.g., Lebowitz, 1986; Pazzani, 1990) or for learning different types of features (e.g., Mooney, Ourston, & Wu, 1989). We believe that each of these arguments provides sufficient reason for the study of how instance comparisons affect concept formation.

In this chapter, we present some further psychological results on concept formation that we believe address some fundamental questions about the representation, use, and learning of concepts. Our purpose in this chapter is to examine one particular way in which concept formation may occur through the comparison of instances. We will first present the motivations, ideas, and results, and then come back to discuss the implications for other work.

1.1 General Motivation

Much work in psychology and cognitive science takes as a starting point that human thought relies upon knowledge of past experiences. The issues then become the nature of this knowledge and how it is used to perform the task at hand (e.g., prediction, inference, problem solving). The modal view has been that these experiences allow the development and application of abstract knowledge. Recently, however, across a wide variety of tasks and domains, this assumption of abstraction has

1. Wisniewski and Medin (this volume) argue for a more interactive relation.

been questioned (see Medin & Ross, 1989, for a review). Many results that seem to arise from the use of abstraction may not involve abstraction at all, but rather the use of knowledge of specific instances (e.g., Hintzman, 1986; Jacoby & Brooks, 1984). The demonstrations of how non-abstraction can simulate behavior that appears to be caused by abstraction has created a tension between abstraction and non-abstraction views of human cognition and performance.

In categorization work within psychology, this tension has primarily been between abstraction views (such as prototypes or feature frequency theories) and exemplar views. The abstraction views assume that experience with instances leads to the formation of some summary representation, which is then used for categorizing new instances (e.g., Posner & Keele, 1970; Rosch, 1975; Neumann, 1974). These views stress the stability of the knowledge being used to make categorizations. Exemplar views propose that categorizations are not made on the basis of a single summary representation, but rather on the basis of disjunctive specific knowledge, such as the representations of instances (e.g., Hintzman, 1986; Medin & Schaffer, 1978). The simple exemplar view is that learning occurs by the storage of new instances and that test instances are categorized by their similarity to one or a set of old instances. Thus, the knowledge used to categorize is flexible, in that it may depend upon the instance being categorized.²

Each view has had its victories and defeats on the empirical battlefield during the last decade. Abstraction theories often have had difficulty in accounting for people's sensitivity to category size, exemplar frequency, and feature correlations within a category, and thus have rarely provided any mechanism by which the summary representations might be learned (Nosofsky, 1988; Malt & Smith, 1984). Simple exemplar models have encountered difficulties in explaining encoding effects on new exemplars and the circumstances in which generalizations are formed (Elio & Anderson, 1981, 1984; Medin, Dewey, & Murphy, 1983).

The difficulties with each approach have led some researchers to propose that very different representations and processes may underlie categorization performance (e.g., the PDP work, such as that of Gluck & Bower, 1988; Knapp & Anderson, 1984; McClelland & Rumelhart, 1985). However, other researchers have viewed the earlier results as suggesting that both general and specific knowledge are used in cate-

2. See Ross, Perkins, and Tenpenny (1990) for an elaboration of this distinction.

gorization. This view leads to the proposal of hybrid models, with the intent of incorporating the positive aspects of each model type. However, such hybrid models lead to a new set of problems. In particular, these models must now deal with the issues of (1) when the generalizations are formed, and (2) the nature of these generalizations.

1.2 Reminding-Based Generalizations

The current proposal is a hybrid model in which the generalizations are formed from the comparison of instances, but in a non-automatic way. Two primary motivations behind this view may help in understanding it. One motivation comes from the exemplar proposals, which have shown that people sometimes categorize new items by their similarity to old items (Brooks, 1978, 1987; Jacoby & Brooks, 1984; Medin & Schaffer, 1978; Whittlesea, 1987). The importance of such instance-based performance in people is also seen in a variety of prediction and reasoning studies (e.g., Gilovich, 1981; Read, 1983). However, these views typically stress the use of instances for the current task but do not examine how such use may affect still later performance.

A second motivation for the current proposal comes from research in problem solving. In the learning of complex new domains, novices will often solve current problems by using earlier examples of which they are reminded (e.g., Anderson, Farrell, & Sauers, 1984; Reed, Dempster, & Ettinger, 1985; Ross, 1984). More importantly, the use of one problem to solve another requires the novice to abstract over a variety of differences between the problems. If this abstraction is stored, it may be available for use in later problem solving (e.g., Gentner, 1989; Kolodner & Simpson, 1989; Pirolli & Anderson, 1985; Ross, 1984, 1989; Ross & Kennedy, 1990; Schank, 1982).

Thus, combining these two motivations, we have the following: when people are categorizing new instances, they may be reminded of some earlier instance and use this earlier instance to categorize the new instance. Such reminding-based categorization may lead to a generalization (essentially incorporating the commonalities of the two instances) that can then be used for categorizing later instances. Thus, *reminding-based categorizations can lead to generalizations*. This proposal goes beyond the exemplar theories by allowing the use of exemplars to affect the knowledge stored. In particular, the generalizations formed by these reminders may, with sufficient experience, come to largely con-

trol categorization performance. However, as we will argue later, such a view suggests very different answers from the abstraction view as to the nature of concept representation. Clearly the framework is presented here at only a general level and many questions remain. Even so, it has a number of interesting implications and predictions that have not been empirically tested. Before considering some of the implications of such a view, we turn to a fictitious example to help make some of the ideas a bit clearer.

1.3 An Example of Reminding-Based Learning

Suppose you are at a workshop, symposium, or conference session and you listen to a number of speakers. Let us suppose that you are trying to get various information, principally what the main points are and what the implications are for your interests. However, you are also encoding much beyond that. These further encoded features include not only features that may turn out to be irrelevant, but also features that you are sure are irrelevant. Simply by attending to the speaker and the talk, much superficial and contextual information will also be encoded. For instance, you might encode information about the talk's motivation, approach, and the tightness of its argument; about the speaker's presentation style, humor, and appearance; and about the context such as the coldness of the room, your need for coffee, or the scenery through the window.

As an example, suppose one of the later speakers had green spiked hair. It might not be surprising that such hair may make you think of another speaker, either in that or another session, or one you heard at some other time, with such a coiffure. You may understand that hair style has little to do with intellectual contribution, but then you may notice that both speakers had the same graduate advisor and both the talks were on very boring topics. What then? If you find out that a later speaker also was trained by the same advisor, would you have any prediction about the talk?

We hope that the obvious silliness of the example does not cause the reader to miss the point. The point is that this similarity between speakers, although of questionable heuristic value as an indication of scientific importance, leads to further comparisons that affect what is learned about the speaker and what may be considered about future speakers.

1.4 Implications of Reminding-Based Generalizations

In this section, we consider some implications of this general idea of reminding-based learning, with references back to the example. Clearly, much remains to be specified about this view, but our purpose here is to show that this general class of models has some interesting implications. Later we will entertain some speculations. Here we wish to consider some implications that have at least some preliminary empirical support, which we present next. In particular, we will discuss how this view proposes which generalizations are formed, what aspects may be included, how such a process may focus the learner, and how this view may incorporate effects of other knowledge.

1. *Which generalizations?* Some systems of concept learning form generalizations automatically from the instances (e.g., Anderson, Kline, & Beasley, 1979; Carbonell, 1983). However, the current proposal, among others, suggests that generalizations are not automatically formed, so it is necessary to explain when (and which) generalizations are produced. The reminding-based view makes two important points about this *selective* aspect of generalization. First, the generalizations are not formed in a single step over large classes of instances, but rather between two instances (or an earlier generalization and the current instance). Second, the generalizations are formed as a function of use. That is, the generalization comes about as a by-product of comparing an earlier instance to the current instance, not as the result of a separate generalization process. Thus, one cannot just look at a set of items and determine what the generalizations will be. Rather, it is necessary to understand how (and which) earlier instances will be used to deal with the new instances. Clearly, this view assumes an incremental processing of instances and the generalizations formed will depend on the particular order of the instances.

Using our speaker example, there may well be a cluster of speakers who match the current one on important features, but the superficial feature green spiked hair may remind the listener of another such speaker and it is *that* speaker who is used for comparison and then as a basis for generalization.

In the related work on problem solving and comprehension, a number of studies have investigated what factors affect the reminders that people have (e.g., Faries & Reiser, 1988; Gentner & Landers, 1985;

Holyoak & Koh, 1987; Ross, 1987, 1989). For the current purposes, it is most important to note that, although relevant aspects affect remindings (as one might hope), it is also true that some irrelevant or superficial aspects can have a large influence on remindings. Even aspects that people know are not relevant can affect the earlier example that they retrieve; for example, Ross (1984) uses the story lines of word problems. Thus, superficial characteristics that are unusual and salient, such as green spiked hair, may be sufficient to make one think back to a similar hairstyle, even in the absence of relevant similarities.

2. *What is learned from a reminding?* If an earlier instance is used in dealing with the current instance, some generalization may be created. What is the nature of this generalization? Most importantly, *the basis of the reminding may be very different from what is learned from using the reminding*. That is, even if the reminding is based on superficial features, the comparison of the instances may lead one to learn more important similarities. For example, the green spiked hair may lead one to notice that the talks were of similar levels of interest and the speakers had the same graduate advisor. The reminding restricts what earlier information is accessed for dealing with the current instance, but it does not restrict it simply to the features that led to the reminding.

This point is crucial for two reasons. First, there is often a correlation between superficial and deeper aspects. Medin and Ortony (1989) argue that people often use an *essentialist* heuristic, believing that things that look alike tend to share deeper properties as well. In many cases, this heuristic is correct. Thus, although the influence of superficial characteristics on remindings may appear to be a great impediment to the operation of an intelligent system, often it leads to appropriate remindings and generalizations.

Second, this view provides one possible means by which learning may occur. One of the great challenges in concept learning is that the features used for concepts may change as people become more familiar with them. Such a novice-expert shift is commonly found in more complex problem-solving domains (e.g., Chi, Feltovich, & Glaser, 1981) and involves changing from a reliance on more superficial aspects to a reliance on deeper, more principled features of the domain. Thus, one needs some means by which more structured knowledge is learned as a function of superficial similarities.

In addition, this view helps to relate the learning of categories to later categorization performance. Many current categorization theories in psychology are performance theories, specifying how performance is governed once the knowledge has been acquired. By the reminding-based view, the later performance depends critically upon the earlier performance, because the knowledge that is used to categorize earlier instances affects the generalizations that are made (and that are available for categorizing subsequent instances). Thus, the performance modifies the knowledge available for later processing.

3. *Focus of the generalization.* One of the defining characteristics of work in concept formation (see Gennari et al., 1989) is that the learning occurs in an *unsupervised* environment, without any external feedback about the correctness of the predictions or classifications. Such learning forces us to consider the multiple uses to which the knowledge may be applied. So far, reminding-based learning has been principally studied within supervised settings both in problem solving (Ross & Kennedy, 1990; much of the case-based work discussed later) and categorization (Medin & Edelson, 1988; Ross, Perkins, & Tenpenny, 1990). How might such an idea be extended to an unsupervised setting?

We have recently begun to examine such an idea and will describe one of the experiments in the next section. In earlier work, we assumed that the reminding, when used to categorize (i.e., when the category label was the same), would promote the noticing of other commonalities as well. However, by some accounts there is nothing special about the category label in terms of the representation — it is simply one of the features (e.g., as argued by Anderson & Matessa, this volume). Although we believe that category labels may have some special properties (e.g., see Barsalou & Ross, 1986), we do not believe that the effect of reminders is dependent on the presence of a category label. Thus, if learners did not have available a category label, we assume that they would still notice other commonalities and such noticed commonalities might be used in making predictions about later instances.

For example, listening to talks is partially unsupervised. Only internal feedback is available (except in the rare occasions when a talk prompts ovations or catcalls) and, most important, there may be no single goal in mind for the listener. Reminders, especially when they lead the listener to notice an interesting commonality, may serve to focus the listener on some aspects of the talk. Later, the representations of these

talks and any noticed commonalities may be used in making predictions about any of a large number of features.

4. *Knowledge effects.* Until now, we have been discussing the reminding effects as if the instances consisted only of lists of features. However, such a restriction was used simply for convenience of exposition. Although the details of such a process will become more complicated with more complex representations, there is no reason that remindings and noticing of other commonalities cannot involve more structure. The evidence in problem solving is that remindings and their use are affected by both superficial and deeper aspects that involve structured knowledge (e.g., Holyoak & Koh, 1987; Ross, 1987, 1989). In addition, much of the related work in case-based reasoning (e.g., see Kolodner & Simpson, 1989) makes use of structured knowledge in remindings and their use.

Returning to our trusty speaker example, much knowledge may underlie the commonalities noticed. The green hair may lead one to notice graduate advisors, but such knowledge may access other information about the institutions in which they received graduate training or the traditions in which their research proceeds. The point here is that the reminding does not require one to deal only with simple information; it simply focuses the listener. Thus, one may still get effects of knowledge, but the exact forms they take will be influenced by other factors affecting learning, such as the remindings; Experiment 4 in Ross et al. (1990) shows such an interaction.

1.5 Summary

In this first section, we have proposed, in admittedly general terms, a reminding-based view of how instance comparisons affect concept formation. We then discussed some implications of this view, which we believe are central for any theory of concept learning.

2. Empirical Results on Reminding-Based Generalization

We present here some brief descriptions of empirical results related to the four implications of reminding-based generalization that were discussed in the previous section. Those results available in published form (Ross et al., 1990) will be mentioned only briefly.

1. *Which generalizations?* An important implication of the reminding-based view is that generalizations are formed as a function of use. Thus, which generalizations are formed depends upon which earlier instances are compared to the new instances.

In fact, Ross et al. (1990) tested this idea in several experiments (also see the study in the next section). The focus in these studies was on the effect of instance comparisons, so the situation was set up such that subjects had to use earlier instances to categorize new instances. For example, using abstract notation, two members of a category might be Person A, with features A1 and A2, and Person B, with features B1 and B2. The reminding manipulation occurred by having subjects categorize a new instance, which had features A2' and B2' (where the ' indicates that the features were associated with A2 and B2). In addition, the new instance either had A1' (a salient feature associated with A1) or B1' (a salient feature associated with B1). The logic was that if this manipulation of presenting A1' or B1' affected remindings used in categorization, it would lead to different knowledge about the category.

More specifically, if the test instance (A1' A2' B2') led people to think of Person A (with features A1 A2), they might believe the category had A1-like and A2-like things. However, if the test instance (B1' A2' B2') were presented and reminded them of Person B, then the category might be viewed in terms of B1-like and B2-like things. To test this idea, the final test asked them to rate the representativeness for the category of various features, including A2'' and B2'' (i.e., A2'' were items related to A2 and A2'). The results were as predicted, with A2'' ranked as more representative for subjects having (A1' A2' B2') test instances and B2'' ranked higher for subjects receiving (B1' A2' B2') test instances. Which earlier instance (i.e., Person A or B) was used to categorize a new member affected what subjects learned about the category representativeness of the features.³

2. *What is learned from a reminding?* The main point brought out in the earlier discussion of this issue was that the basis of the reminding may be very different from what is learned from using the reminding. In particular, superficial similarities that lead to remindings may result in the learner noticing other non-superficial regularities. This idea is

3. Note that these tests were on features, A2 and B2, that were exactly the same for all subjects.

Table 1. Abstract design and example test from Spalding and Ross (1990). Italics indicate the generalization features for subjects in Color Set 1, whereas bold font denotes the generalization features for subjects in Color Set 2.

	INSTANCE	FEATURES	COLOR SET 1	COLOR SET 2
CATEGORY 1	1	A B E	RED	RED
	2	A C F	RED	ORANGE
	3	D B G	ORANGE	RED
	4	D C H	ORANGE	ORANGE
CATEGORY 2	5	E F D	BLUE	BLUE
	6	E G C	BLUE	PURPLE
	7	H F B	PURPLE	BLUE
	8	H G A	PURPLE	PURPLE

For example, study instance 7 would have the three features H F B presented in either purple (if the subject were in Color Set 1) or in blue (if the subject were in Color Set 2).

crucial to the reminding view presented here, because of the influence superficial aspects have on what instance is remembered.

We investigated a simple situation in which the reminders would occur on the basis of superficial information, the color in which the instances (lists of features) were presented (Spalding & Ross, 1990). The logic was as follows: If the subject sees an instance which is printed in red, she or he may think back to an earlier instance of the same color, and then compare the features in the two instances. By the reminding view, similarities between the two instances would then be included in a generalization that could be used in later category-related judgements. Table 1 presents an abstract design of this study, in which two instances were presented in each of four colors (two colors per category), and the colors were perfectly predictive of category membership. However, subjects were told there would be later tests that would not have any color. Thus, color was known to be unavailable during the target task.

The two instances presented in a given color had one feature in common. For example, half the subjects had two instances printed in red, both of which had feature A (e.g., likes gardening). Features that were correlated with color and were expected to be included in a generaliza-

tion are referred to as *generalization features*. For the other half of the subjects, the two red instances had feature B in common. All subjects saw the same sets of features. The only difference between the two conditions was which instances were presented in which colors.

The effect of remindings was measured using four tests, but because the effects were similar, only one test will be described here. In the categorization tests, the subjects were asked to categorize new instances composed of one or two generalization features from one category and one or two non-generalization features from the other category (e.g., A F or B C E H). If the color similarity affected what was learned, generalization features should be used to make the categorization more often than non-generalization features. The results strongly supported this prediction. On 74% of the trials (chance = 50%), the subjects categorized these test instances as predicted. In addition, 11 of the 12 subjects showed this effect.

Clearly, the manipulation of the superficial feature, color, affected what was learned about the categories. In studies without this type of manipulation, it seems reasonable to us to believe that some instances make people think back to earlier instances and that something may be learned as a result of these remindings. However, because superficial similarities may influence these remindings and because different people may be reminded by different features, the effects of such remindings may not be evident from measures of categorization performance.

3. *Focus of the generalization.* We proposed earlier that another implication of reminding-based category learning is that remindings focus learners on some aspects of instances, those aspects in common to the instances compared. As materials become more complex, it becomes harder for the learner to know what may be important. Thus, the focussing function of remindings might be important in more complex learning situations. In an experiment to test this possibility (Spalding & Ross, 1991), we also extended the previous findings to a new paradigm, in which there is only one category. As pointed out earlier, we have no reason to believe that remindings and comparisons of instances depend on the presence of category labels, so the effect of remindings should occur in the single category design. This study investigated whether a more complex learning situation may show more effect of remindings. The design is given in Table 2. Each attribute (e.g., a hobby the person enjoys) had three values (e.g., painting, photography, weaving).

Table 2. Abstract design of focussing experiment for the nine instances. Attribute A was the generalization feature for subjects in Color Set 1, whereas attribute B was the generalization feature for subjects in Color Set 2. All of the attributes were used in Condition 4A, but only A and B were used in Condition 2A.

ATTRIBUTE VALUES				COLOR OF PRESENTATION	
A	B	C	D	COLOR SET 1	COLOR SET 2
1	1	1	1	BLUE	BLUE
1	2	2	2	BLUE	GREEN
1	3	3	3	BLUE	RED
2	1	2	3	GREEN	BLUE
2	2	3	1	GREEN	GREEN
2	3	1	2	GREEN	RED
3	1	3	2	RED	BLUE
3	2	1	3	RED	GREEN
3	3	2	1	RED	RED

For the very simple condition (referred to as 2A), the instances had only two attributes. The instances were created by combining the three values on each of the two attributes, resulting in nine (3×3) instances. For the more complex condition (4A), we simply added two more attributes, each with three possible values, to each instance. Each new attribute value was pairwise independent of each of the other attribute values. Remindings were again manipulated by the color in which the instances were presented, and subjects were again told that there would be later tests in which there would be no color. As can be seen in Table 2, for half of the subjects, attribute A-value 1 was always presented in blue, attribute A-value 2 was always presented in green, and attribute A-value 3 was always presented in red. For the other half of the subjects, attribute B's values were correlated perfectly with color.

As a test of their category knowledge, subjects ranked the six attribute values from A and B with respect to their importance to the category. If the colors influenced what was learned about the category, then the attribute values correlated with color should be ranked higher than the attribute values not correlated with color.

The interesting comparison here is the 2A condition versus the 4A condition, a between-subjects comparison (half the subjects were assigned to each condition). The focussing prediction was that the influence of the color remindings should increase with the complexity of the material, and this result was obtained. In the simple 2A condition, subjects showed no effect of the color manipulation, with the average rank difference only 0.1 of a possible 3. Further evidence that color had no effect was that ten subjects responded in the predicted direction, but ten also responded in the opposite direction. However, in the more complex 4A condition, the color manipulation had a large effect, with 1.2 of a possible 3 average rank difference and 18 of 20 subjects responding in the predicted direction. The remindings cause subjects to focus on the reminding attribute values in the 4A condition, but not in the extremely simple 2A condition.

4. *Knowledge effects.* The final implication discussed earlier was that remindings can involve structured knowledge and interact with knowledge-based effects. We hope to extend our work in this area in further research, but one experiment (Ross et al., 1990, Experiment 4) does show how remindings may interact with effects of knowledge. In this experiment, we manipulated which study instances were likely to be used to make category judgements about the first set of test instances.

After the subjects had learned the study instances (e.g., Alice has a pet dog, swims, reads drama, and eats peaches; she is in Club X), they were given information about the relevance of features (e.g., that category membership depends on the type of pet and the type of sport). For the first test, some new instances were presented for categorization (e.g., a person who has a pet dog, jogs, reads mysteries, and eats peaches). In later tests, subjects were shown each feature and had to categorize it as quickly as possible. The knowledge of feature relevance interacted with the remindings we predicted would be used in the first test categorizations. In particular, for features held in common between first-test instances and the studied instance that we predicted would be used to categorize it, relevant features (e.g., dog) were responded to faster than irrelevant features (e.g., eats peaches) on the later tests. For features not in common (and so predicted not to be used in the first test remindings), there was no difference in responses to relevant (e.g., swims, jogs) and irrelevant (e.g., reads drama, reads mysteries) features. Thus, a simple knowledge effect interacted with the remindings.

In this section, we have briefly described some empirical work that bears on the four implications discussed in the previous section. These studies show that which generalizations are made depends upon the remindings that occur. In addition, remindings based on superficial aspects can influence the generalizations. Thus, what is learned from a reminding can be very different from the basis of the reminding. We then described how remindings can help people focus on particular features during the learning of complex materials and how they can influence the way in which more top-down knowledge is used.

3. Discussion

The first section of this chapter presented the general idea of reminding-based learning of categories, and the second section provided a flavor of the experimental program demonstrating and analyzing this learning. In this last section, we speculate on some further issues that arise from considering this type of learning and then discuss the relation between this view and extant theories.

3.1 Speculative Points Related to Reminding-Based Learning

We address here a few further issues that are important, but have not yet been addressed by our research studies. In particular, we discuss the implications of this view for the representations of well-learned categories, the learning of natural concepts, and the continued influence of superficial aspects in later performance.

1. *The representation of categories.* The reminding-based view eschews strict exemplar or abstraction representations in favor of a representation of categories that includes both. If a category is well defined (e.g., has necessary and sufficient properties), then the generalizations from the remindings (and comparisons to earlier generalizations) will eventually lead to the defining features. However, in cases of ill-defined categories, the representation will usually involve no single summary representation. Rather, the category will be represented often by a number of overlapping generalizations at varying levels of generality.

Consider the basic level category *birds*. Although at the basic level, this category actually includes a few subclusters in which there is a very strong correlation of features, such as songbirds, birds of prey, and

water birds (Malt & Smith, 1984). To view the representation of birds as consisting of a single prototype may simply mean that when people say birds they generally mean songbirds. The representation of songbirds, however, is not a sufficient representation of the category of birds and is unlikely to be used in making predictions about the eating habits of a new large bird with sharp talons. In addition, people may also have overlapping clusters, such as pet birds or colorful birds, which could be used in some circumstances to make predictions or categorize new instances. Thus, a single instance can be used in multiple reminders and may be the source of multiple generalizations, as might happen, for example, if one had a familiar pet bird.

This view, then, shares with exemplar views the idea that different knowledge may be used in categorizing different instances, but also allows generalizations to derive from earlier categorizations or other comparisons. Speculatively, this way of representing categories may also allow some necessary context sensitivity (Barsalou, 1982; Brooks, 1987; Roth & Shoben, 1983). If one believes that an important function of concepts is prediction, then concept use must be sensitive to context.

2. *Concept formation in a rich environment.* We have been arguing for a particular means of learning about categories. However, our learning of concepts in life differs in many ways from these simple psychological or computational studies. Although we do not want to argue that one needs to study concept formation only in the "natural world", it is instructive to examine how such situations may differ from the ones we commonly assume. We mention here some differences that we think point to an increased role for reminding-based learning; Brooks (1978, 1987) discusses this issue at greater length.

Our experiences of objects and events differ from our experimental assumptions in a number of ways. First, the instances tend to be much more complex, with a relatively large number of relevant and irrelevant features. Instances, and our ways of interacting with instances, may be more distinctive than in laboratory studies, making reminders of particular instances more likely. Second, every event or object occurs in some context, and often the instance may interact greatly with the context. Thus, instances are often more distinctive and context-tied than in many experimental studies. Third, we rarely are presented with many similar instances of a given type successively. Rather, especially for objects, we may have lots of experience with relatively few instances.

Thus, we may have available a few especially familiar instances of various types, a useful condition for reminding-based learning.

It is also instructive to consider how we might use these concepts. First, rather than categorize new instances or make predictions about one feature given all the others, we often need to classify and make predictions about instances on the basis of incomplete information. Second, such predictions often are needed quickly. These two conditions make it unlikely that we are depending upon one or a small set of criterial deep features in accessing categories or making predictions. Rather, we may be able to access the relevant information from a variety of probes, often using ones that are readily available (and so superficial in at least that sense). Brooks (1987, p. 143) argues that this use of instances is often justified because "ecologically, most conceptual neighborhoods contain more friends than enemies, more useful than misleading analogies among stimuli that seem very similar".

Our point here is not to argue that natural concept formation is strictly based on remindings. Rather, we wish only to point out that many decisions about the stimuli and procedure that are made for experimental convenience may bias the learner towards analytic strategies and overestimate such learning in concept formation (see Allen & Brooks, 1991; Brooks, 1978, 1987).

3. *Superficial influences in later performance.* We have argued for a role for remindings in category learning and for the idea that these remindings may often be influenced by superficial characteristics of the instances. Although the later effects of such influences have not been extensively examined, some studies have shown that they can persist with greater learning or expertise. We mention here two very different types of results. First, Gilovich (1981) showed that experts (professional sportswriters) asked to predict the success of hypothetical college football players were influenced by superficial characteristics of the players that made them more similar to particular professional players. Thus, experienced judges sometimes will make instance comparisons and which comparisons are made may be influenced by superficial aspects. Second, Hinsley, Hayes, and Simon (1977) found that proficient solvers of algebra word problems appeared to identify and solve the problems differently depending upon whether the superficial story line of the word problem was one typically used for this type of problem.

It is likely that experts have specialized schemas. Such schemas go beyond encoding simply the most abstract generalization of the knowledge, by including additional information that may frequently occur (e.g., Fisher, Yoo, & Yang, 1990; Mooney et al., 1989; Shavlik, DeJong, & Ross, 1987), even if that information is superficial. One advantage of such superficial specialization is easier access and use when the new instance matches the specialization (e.g., Brooks, 1987; Medin & Ross, 1989). For instance, using the Hinsley et al. (1977) items, knowing that the word problem begins with "A river steamer ..." allows the quick access of some knowledge about distance-rate-time problems and allows an easy application.⁴ Specializations of frequently occurring superficial information would be expected if learners were using remindings and generalizing from them. As is the case with concepts, superficial and deeper information are often correlated. Given this correlation, Lewis and Anderson (1985) note that experts have learned when it is useful to rely upon superficial features. Allen and Brooks (1991) discuss how specializations of explicit rules may occur with practice.

3.2 Relations to Extant Theories

In this section, we consider a number of extant theories of concept learning. The goal here is not to provide a comprehensive introduction and analysis of these theories, but rather to use them to bring out some important aspects of concept formation. For exposition, we break these down into reminding-based, case-based, pattern composition, and hierarchical concept formation views.

1. *Reminding-based views.* The current proposal is related to a number of recent ideas in psychology. In particular, Medin and Edelson (1988) present a modification to a well-known exemplar model, the context model (Medin & Schaffer, 1978), based on some data they collected on the use of base rate information. The context model classifies new instances by their similarity to old instances, using a multiplicative function of the feature similarities to capture the co-occurrence of features. The extension to the context model included the incorporation of feature competition and context-sensitive retrieval. They suggested that when an instance is presented and an earlier instance is used to clas-

4. In fact, for this problem, subjects often realized many more details would be forthcoming, such as the rate of the current.

sify it successfully, common properties are rehearsed or strengthened in the current instance. If the classification is incorrect, then the features distinctive to the current instance will be strengthened.

The current proposal has much in common with the ACT generalization model (Anderson et al., 1979) and the elaborations of it based on the work of Elio and Anderson (1981, 1984). This model proposes that frequently occurring feature combinations are automatically abstracted during learning and used for later classification.

The current proposal differs from the elaborated ACT generalization model in three important ways. First, rather than assuming generalizations are formed by instances that happen to reside together in working memory, the current view stresses the idea that the reminding and use of an earlier instance to categorize a new instance *forces* the two instances to be in working memory together (i.e., thought about together) and such remindings may be influenced by superficial features. This difference is crucial, especially given the arguments presented earlier about how objects and events may be experienced outside the laboratory. Second, the ACT generalization model requires comparing two instances that have previously been categorized, while the reminding-based view requires only one earlier categorized instance. Third, unlike the ACT view, the current proposal assumes that generalization is not automatic.

In fact, Anderson (1986) has made these same last two points, among others, in arguing for a major modification to the ACT generalization model in which the "generalizations emerge as a by-product of the analogy process without a separate generalization phase" (p. 304). Although he suggests a very different generalization mechanism (applying knowledge compilation processes to the trace of an analogy), this view is similar to ours in that remindings of earlier instances used by analogy can lead to a generalization.

Thus, both the Medin and Edelson proposal and the Anderson view are quite similar in spirit to the reminding-based proposal presented here. Because of the difference in focus, we have been more concerned with how superficial features may influence the learning and with providing direct experimental demonstrations and tests of these ideas.

2. *Case-based reasoning.* The idea of accessing and using the knowledge about earlier instances has also been incorporated into a number of computational views, such as the *case-based reasoning* systems. The general idea (see Hammond, 1989; Kolodner & Simpson, 1989) is that

the analysis of the current case is used to probe memory to retrieve an earlier case. The solution of this earlier case is then adapted and applied to the current case, and its use is verified. Finally, memory is updated both by adding this new case and by adding any generalized knowledge learned in the course of applying the old solution. Both of these modifications are available for the processing of later cases.

As may be seen by this description, there is considerable similarity between the current proposal and case-based reasoning systems. The case-based systems usually deal with more complex domains and more structured concepts, leading to a focus that is rather different from the current work.⁵ Perhaps the most important difference in focus is the role of superficial similarities. The case-based systems, as in Schank (1982), have tended to emphasize the analysis of current instances to get the deeper, underlying description of events, while our work has focussed on the way in which even remindings based on superficial features can lead to reasonable learning. The reasons for this difference in emphasis, we think, have to do with our focus on people learning complex new domains. In such situations, novices often rely on superficial features of the problems (e.g., Chi et al., 1981). Our reasons for carrying this idea into the work on categorization are (1) because of ideas such as the essentialist heuristic suggested by Medin and Ortony (1989) that people assume superficial and deep features are correlated, and (2) because of the usefulness of superficial features in problem-solving research in helping to analyze exactly what people were doing (Ross, 1984, 1987, 1989; Ross & Kennedy, 1990). This use of superficial features, as we argued earlier, has important implications for how categories come to be represented and for how learning proceeds. In addition, our view allows the direct access of instances without first traversing the memory generalizations, although some new directions in case-based systems may be compatible with such an idea (Kolodner & Thau, 1988).

3. *Pattern composition.* Martin and Billman (this volume) provide a useful analysis of two different interpretations of generalization (see the chapter for details and tests). First, they distinguish generalizations in terms of the number of attributes, with more generalized knowledge containing fewer attributes. This interpretation is consistent with many

5. However, the research in problem solving (e.g., Ross, 1984, 1987, 1989) that provides one of the primary motivations for this categorization work has been influenced directly by case-based systems.

models in which irrelevant features are deleted when the generalization is formed. Second, they point out that generalizations also vary in terms of the amount of evidence or number of instances supporting them, with more generalized knowledge usually supported by more evidence. In many models, these two meanings of generalization are both used, often varying together. Although this distinction does not perfectly separate the different generalization theories, Martin and Billman argue that it does contrast some important classes of models. In this section, we will examine one class of models, which they call *pattern composition* models.

Pattern composition models assume that the representations can vary in the number of attributes specified (i.e., the first interpretation of generalizations mentioned above). In such models, the instances are not partitioned by a hierarchy; rather, the regularities are represented separately. A number of concept learning models include separate representations for generalizations that contain fewer attributes than do the instances, although again such representations may also vary in the number of instances supporting them (e.g., Anderson et al., 1979). One example of a pattern composition model can be found in the work of Billman and Heit (1988). They describe a focussed sampling model in which the regularities consist of simple rules stating that if one attribute has a particular value, then another attribute has a particular value. Each rule has a strength indicating the validity of its inference, based upon its successes and failures in prediction. In addition, each attribute has a strength indicating its likelihood of being sampled as the basis of a prediction. This strength increases as an attribute proves useful as a predictor.⁶

The current reminding-based proposal has a number of similarities to pattern composition views. In particular, generalizations will vary in terms of their number of attributes. In addition, although the current proposal is not detailed, the regularities are not viewed as explicitly partitioning the instances into disjoint sets. The primary difference may best be described as one of chunk size (Billman, personal communication). Billman's work has tended to focus on small regularities, such as one attribute being used to predict another, as in Billman and Heit (1988). In this case, larger patterns are realized through the mutual

6. Chalnick and Billman (1988) include more complex conditional rules, effectively allowing for overlapping categories.

facilitation of these small rules (because of the feature strengths). The reminding view allows such simple generalizations to be formed, but initially it tends to promote larger regularities because of the influence of superficial similarities. That is, usually an instance one remembers will have a number of features that overlap with the current instance, all of which will be included in the generalization. However, the commonalities noticed and included depend, of course, upon the commonalities present in the instances.

4. *Hierarchical concept formation.* Models of hierarchical concept formation are well described throughout this volume and also in Gennari et al. (1989). For the present purposes, four characteristics are important: such models represent knowledge in a concept hierarchy; this hierarchy is used to make top-down classifications of new instances;⁷ learning is incremental; and learning occurs without supervision. Although current concept formation models vary in their details and are still being developed, it is useful to consider the current proposal's relation to this general description. We think two differences are interesting to discuss.

First, the concept formation models were designed with the idea of avoiding extensive reprocessing of every instance during classification of each new instance. A node in the concept hierarchy contains knowledge that summarizes the instances included below it and allows the top-down classification of new instances. Our view takes a position in line with pattern composition views that regularities should be represented separately. In addition, we also want to allow for direct access of instances (without going through the hierarchy) if new instances are very similar or contain similar attributes that are highly distinctive. For example, a repetition of an instance may lead to the retrieval of its earlier presentation without being passed down through the hierarchy. As a different example, an instance that contains a very unusual value on an attribute that matches an earlier instance might lead to this earlier instance being retrieved without going through the top-down classification. Thus, although it seems quite reasonable that important features should be extracted and used in a top-down fashion, it may also be important to have alternative means of classification (see also Silber & Fisher, 1989; Porter, Bareiss, & Holte, 1990).

7. However, some models (e.g., Silber & Fisher, 1989) do not require this strict top-down classification.

Second, as advocated by the hierarchical concept formation view, it seems likely that some instances (e.g., ones sharing many features with each other and not with other instances) are grouped together. The basic argument of this view (see Anderson & Matessa, this volume; Fisher, 1987; Yoo & Fisher, this volume) is that classification is the goal and that inference falls out as a by-product of classification. By having classified instances together, one has a reasonable basis for using the class features to make predictions about some feature(s) of a new instance. The approach taken in this chapter argues that it is the access and use of earlier instances that permits them to be considered as members of a class. This difference is important because the task may affect which instances are accessed and used and, hence, which classes are formed. Although the experimental studies have used categorization, the same idea can be applied to any inference task. Once an instance is used to infer something about another instance, it is as if they are in the same class (for the current task). The punch line, then, is that the reminding-based view is arguing the *classification is a by-product of inference*. We have the classes we do because we have (retrieved and) used instances within a class to make inferences about other instances in the class.

The point is put more strongly than seems reasonable in order to argue that it should not be overlooked in current theories. Rather, it seems that the views are somewhat complementary. On the one hand, it seems likely that we have knowledge about concept hierarchies that we use in our general default mode (i.e., unsupervised) of learning and comprehending the world. On the other hand, it also seems likely that there are a variety of circumstances in which some other knowledge than this hierarchy is used for accessing a relevant instance directly and using it to accomplish some task. In addition, whatever is learned from this use might also be stored and used at some later time, even if it does not fit neatly into the concept hierarchy.

These differences between the reminding-based view and the hierarchical concept formation models should not obscure some important similarities (Fisher, personal communication; Langley, personal communication). In both approaches, the incremental processing of instances is crucial. In addition, both views build abstractions during the processing of new instances and this processing often involves the retrieval of earlier instances.

3.3 Some Commonalities and Differences Across the Views

It is useful to go back through the four types of models that we have briefly discussed to ask what commonalities they contain and in what crucial ways are they different. We mention three commonalities here. First, the models are all hybrids, using both specific knowledge and more generalized knowledge, allowing the representations to be at varying levels of generality. In addition, for each model, the same processes access and use the varying levels of knowledge. Second, and closely related, the knowledge used to make predictions or categorize may vary with the instances under consideration. That is, there may be multiple bases for categorizing an instance as a member of a category. Third, each model has some reason(s) for showing sensitivity to order of the instances, a property not true of many abstraction and exemplar models.

What are the important differences? To answer this question, we return to the idea that some of these views may be complementary. More generally, one needs to account for both the stability and the flexibility of human cognition (e.g., see Barsalou, 1987). Partitionings provide highly structured stable knowledge that can be used across a variety of tasks. However, it is also true that people show incredible flexibility in their ability to use knowledge. For instance, categories may be created dynamically to meet some new goal, such as "things to save from your house during a fire" (Barsalou, 1983). It is very unlikely that some predefined partition has clustered such objects together. Norm theory (Kahneman & Miller, 1986) claims that all uses of categories may involve classes that are constructed dynamically in this way. Whether this stronger claim is true or not, it does seem that the access and use of concepts needs to allow for a great deal of flexibility.

4. Conclusions

In this chapter, we have examined a general view of reminding-based learning. The discussion has centered on the implications of such a view, focussing on which generalizations are formed, the nature and focus of the generalizations, and the effects of knowledge. Some preliminary empirical evidence was described that supported these implications. We then addressed some more speculative implications of this idea related to the representation of categories and schemas. Finally, we discussed the relation of this view to some current theories of concept formation.

The use of concepts appears to involve both considerable stability and flexibility. A challenge for all the views is to remain flexible, but, at the same time, to be able to make use of knowledge about regularities. We believe that part of the solution may require more attention to the learning process and may lie in the ways in which instances are compared and used to learn more general knowledge. In this chapter, we have presented a preliminary proposal for how such instance comparison may influence the formation of concepts.

Acknowledgements

Preparation of this chapter was supported by Grant AFOSR 89-0447 from the Air Force Office of Scientific Research. We thank Larry Barsalou, Dorritt Billman, Doug Fisher, Pat Langley, Doug Medin, Greg Murphy, and Mike Pazzani for their comments on an earlier version.

References

- Allen, S. W., & Brooks, L. R. (1991). Specializing the operation of an explicit rule. *Journal of Experimental Psychology: General*, 120, 3-19.
- Anderson, J. R. (1986). Knowledge compilation: The general learning mechanism. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science*, 8, 87-129.
- Anderson, J. R., Kline, P. G., & Beasley, C. M. (1979). A general learning theory and its applications to schema abstraction. In G. H. Bower (Ed.), *The psychology of learning and motivation* (Vol. 13). New York: Academic Press.
- Barsalou, L. W. (1982). Context-independent and context-dependent information in concepts. *Memory and Cognition*, 10, 82-93.
- Barsalou, L. W. (1983). Ad hoc categories. *Memory and Cognition*, 11, 211-227.
- Barsalou, L. W. (1987). The instability of graded structure: Implication for the nature of concepts. In U. Neisser (Ed.), *Concepts and conceptual development: Ecological and intellectual factors in categorization*. Cambridge: Cambridge University Press.

- Barsalou, L. W., & Ross, B. H. (1986). The roles of automatic and strategic processing in sensitivity to superordinate and property frequency. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 12*, 116-134.
- Billman, D., & Heit, E. (1988). Observational learning from internal feedback: A simulation of an adaptive learning method. *Cognitive Science, 12*, 587-625.
- Brooks, L. (1978). Nonanalytic concept formation and memory for instances. In E. Rosch & B. B. Lloyd (Eds.), *Cognition and categorization*. Hillsdale, NJ: Lawrence Erlbaum.
- Brooks, L. (1987). Decentralized control of categorization: The role of prior processing episodes. In U. Neisser (Ed.), *Concepts and conceptual development: Ecological and intellectual factors in categorization*. London: Cambridge University Press.
- Carbonell, J. G. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Chalnick, A., & Billman, D. (1988). Unsupervised learning of correlational structure. *Proceedings of the Tenth Annual Meeting of the Cognitive Science Society* (pp. 510-516). Montreal, Canada: Lawrence Erlbaum.
- Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5*, 121-152.
- DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1*, 145-176.
- Elio, R., & Anderson, J. R. (1981). The effects of category generalizations and instance similarity on schema abstraction. *Journal of Experimental Psychology: Human Learning and Memory, 7*, 397-417.
- Elio, R., & Anderson, J. R. (1984). The effects of information order and learning mode on schema abstraction. *Memory and Cognition, 12*, 20-30.
- Faries, J. M., & Reiser, B. J. (1988). Access and use of previous solutions in a problem solving situation. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 433-439). Montreal, Canada: Lawrence Erlbaum.

- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139-172.
- Fisher, D. H., & Langley, P. (1990). The structure and formation of natural categories. In G. H. Bower (Ed.), *The psychology of learning and motivation* (Vol. 26). New York: Academic Press.
- Gennari, J. H., Langley, P., & Fisher, D. H. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11-61.
- Gentner, D. (1989). The mechanisms of analogical learning. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. Cambridge: Cambridge University Press.
- Gentner, D., & Landers, R. (1985). Analogical reminding: A good match is hard to find. *Proceedings of the International Conference on Systems, Man, and Cybernetics* (pp. 607-613). New York: IEEE.
- Gilovich, T. (1981). Seeing the past in the present: The effect of associations to familiar events on judgements and decisions. *Journal of Personality and Social Psychology*, 40, 797-808.
- Gluck, M. A., & Bower, G. H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General*, 117, 227-247.
- Hammond, K. J. (1989). *Case-based planning: Viewing planning as a memory task*. Boston: Academic Press.
- Hanson, S. J., & Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3, 343-372.
- Hinsley, D. A., Hayes, J. R., & Simon, H. A. (1977). From words to equations: Meaning and representation in algebra word problems. In M. A. Just & P. A. Carpenter (Eds.), *Cognitive processes in comprehension*. Hillsdale, NJ: Lawrence Erlbaum.
- Hintzman, D. L. (1986). "Schema abstraction" in a multiple-trace model. *Psychological Review*, 93, 411-428.
- Holyoak, K. J., & Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory and Cognition*, 15, 332-440.
- Jacoby, L. L., & Brooks, L. R. (1984). Non-analytic cognition: Memory, perception, and concept learning. In G. H. Bower (Ed.), *The psychology of learning and motivation* (Vol. 18). New York: Academic Press.

- Kahneman, D., & Miller, D. T. (1986). Norm theory: Comparing reality to its alternative. *Psychological Review*, 93, 136-153.
- Knapp, A. G., & Anderson, J. R. (1984). Theory of categorization based on distributed memory storage. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 10, 616-637.
- Kolodner, J. L., & Simpson, R. L. (1989). The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science*, 13, 507-549.
- Kolodner, J. L., & Thau, R. (1988). *Design and implementation of a case memory* (Tech. Rep. No. GIT-ICS-88/34). Atlanta: Georgia Institute of Technology, School of Information & Computer Science.
- Lebowitz, M. (1986). Integrated learning: Controlling explanation. *Cognitive Science*, 10, 219-240.
- Lewis, M. W., & Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology*, 17, 26-65.
- Malt, B. C., & Smith, E. E. (1984). Correlated properties in natural categories. *Journal of Verbal Learning and Verbal Behavior*, 23, 250-269.
- McClelland, J. L., & Rumelhart, D. E. (1985). Distributed memory and the representation of general and specific information. *Journal of Experimental Psychology: General*, 114, 159-188.
- Medin, D. L., & Edelson, S. (1988). Problem structure and the use of base rate information from experience. *Journal of Experimental Psychology: General*, 117, 68-85.
- Medin, D. L., & Ortony, A. (1989). Psychological essentialism. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. Cambridge: Cambridge University Press.
- Medin, D. L., & Ross, B. H. (1989). The specific character of abstract thought: Categorization, problem-solving, and induction. In R. J. Sternberg (Ed.), *Advances in the psychology of human intelligence* (Vol. 5). Hillsdale, NJ: Lawrence Erlbaum.
- Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review*, 85, 207-238.
- Medin, D. L., Dewey, G. I., & Murphy, T. D. (1983). Relationships between item and category learning: Evidence that abstraction is not automatic. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 9, 607-625.

- Mooney, R., Ourston, D., & Wu, S. (1989). *Induction over the unexplained: A new approach to combining empirical and explanation-based learning* (Tech. Rep. No. AI89-110). Austin: The University of Texas, Artificial Intelligence Laboratory.
- Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review*, 92, 289-316.
- Murphy, G. L., & Wisniewski, E. J. (1989). Feature correlations in conceptual representations. In G. Tiberghien (Ed.), *Advances in cognitive science: Theory and applications* (Vol. 2). Chichester: Ellis Horwood.
- Neumann, P. G. (1974). An attribute frequency model for the abstraction of prototypes. *Memory and Cognition*, 2, 241-248.
- Nosofsky, R. (1988). Similarity, frequency, and category representations. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14, 54-65.
- Pazzani, M. J. (1990). *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods*. Hillsdale, NJ: Lawrence Erlbaum.
- Pazzani, M. J., & Schulenburg, D. (1989). The influence of prior theories on the ease of concept acquisition. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 812-819). Ann Arbor, MI: Lawrence Erlbaum.
- Pirolli, P. L., & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39, 240-272.
- Porter, B. W., Bareiss, R., & Holte, R. C. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45, 229-263.
- Posner, M. I., & Keele, S. W. (1970). Retention of abstract ideas. *Journal of Experimental Psychology*, 83, 304-308.
- Read, S. J. (1983). Once is enough: Causal reasoning from a single instance. *Journal of Personality and Social Psychology*, 45, 323-334.
- Reed, S. K., Dempster, A., & Ettinger, M. (1985). Usefulness of analogous solutions for solving algebra word problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11, 106-125.

- Rosch, E. (1975). Cognitive representations of semantic categories. *Journal of Experimental Psychology: General, 104*, 192-233.
- Ross, B. H. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology, 16*, 371-416.
- Ross, B. H. (1987). This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 13*, 629-639.
- Ross, B. H. (1989). Remindings in learning and instruction. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. Cambridge: Cambridge University Press.
- Ross, B. H., & Kennedy, P. T. (1990). Generalizing from the use of earlier examples in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 16*, 42-55.
- Ross, B. H., Perkins, S. J., & Tenpenny, P. L. (1990). Reminding-based category learning. *Cognitive Psychology, 22*, 460-492.
- Roth, E. M., & Shoben, E. J. (1983). The effect of context on the structure of categories. *Cognitive Psychology, 15*, 346-378.
- Schank, R. C. (1982). *Dynamic memory*. Cambridge: Cambridge University Press.
- Shavlik, J. W., DeJong, G. F., & Ross, B. H. (1987). Acquiring special case schemata in explanation-based learning. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 851-860). Seattle, WA: Lawrence Erlbaum.
- Silber, J., & Fisher, D. (1989). A model of natural category structure and its behavioral implications. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 884-891). Ann Arbor, MI: Lawrence Erlbaum.
- Spalding, T. L., & Ross, B. H. (1990, May). *Reminding-based generalizations in categorization*. Paper presented at the meeting of the Midwest Psychological Association, Chicago, IL.
- Spalding, T. L., & Ross, B. H. (1991, May). *Focussing effects of remindings during category learning*. Paper presented at the meeting of the Midwest Psychological Association, Chicago, IL.
- Whittlesea, B. W. A. (1987). Preservation of specific experiences in the representation of knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 13*, 3-17.

CHAPTER 9

Harpoons and Long Sticks: The Interaction of Theory and Similarity in Rule Induction

EDWARD J. WISNIEWSKI

DOUGLAS L. MEDIN

1. Introduction

Concept learning is a fundamental aspect of intelligent behavior. Concepts let organisms relate new experiences to old ones, treat nonidentical experiences as equivalent, and make predictions about new experiences. All of these roles are crucial to an organism's survival. For example, consider the predictive role of a concept. By recognizing that a large, brown, furry creature seen from a distance is a bear, a person can predict that such a creature will be dangerous and avoid it. Given the significance of concepts, it is important to understand how intelligent systems form, update, and use them.

Understanding concept learning is a challenging problem. One can view concept formation as an *unsupervised* task of partitioning n items into a set of categories (i.e., equivalence classes) and forming a concept for each category (e.g., Anderson & Matessa, this volume; Fisher, 1987). In this context, a concept is a classification rule that divides items into those that satisfy the rule (i.e., those belonging to the category) and those that do not. However, there are an exponential number of ways in which n entities can be partitioned into categories; for any n of reasonable size, the learner cannot possibly consider all of them (Anderson & Matessa, this volume).

Even in more constrained, *supervised* tasks (Fisher & Pazzani, Chapter 1, this volume), in which the partition is known and when the learner is presented with preclassified entities, there may be many possible classification rules that apply to each category. Therefore, machine learning has been greatly concerned with discovering useful biases or constraints on concept learning (e.g., Mitchell, 1980; Utgoff, 1986). Psychological studies of concept learning also have attempted to determine the biases that make some categorization tasks easy and natural and others difficult and unnatural (e.g., Medin, 1983; Medin & Schwanenflugel, 1981).

More generally, cognitive psychologists and researchers in machine learning have recently discovered a common agenda with respect to understanding concept learning. Cognitive psychologists have become interested in treating machine learning programs as candidates for psychological process models, and machine learning researchers have turned to psychological research to identify useful constraints (e.g., Fisher, 1988; Fisher & Langley, 1990; Medin, Wattenmaker, & Michalski, 1987; Pazzani, Dyer, & Flowers, 1987; Pazzani & Schulenburg, 1989). In both disciplines, a prominent approach to concept learning relies on empirical or data-driven learning (e.g., Anderson & Matessa, this volume; Dietterich, London, Clarkson, & Dromey, 1982; Fisher, 1987; Hintzman, 1986; Medin & Schaffer, 1978; Michalski, 1983a, 1983b; Mitchell, 1982; Nosofsky, 1986; Quinlan, 1983, 1986; Smith & Medin, 1981). Typically, this strategy involves acquiring concepts based on patterns of similarities and differences that are observed across a number of training items. However, researchers in psychology have increasingly emphasized the limitations of similarity as the basis for concept learning.

A major problem is that empirical learning methods can be misled by irrelevant information (Schank, Collins, & Hunter, 1986). For example, suppose that by coincidence, all the apartment dogs that a learner has been exposed to were brown. Many empirical models would include this irrelevant feature in the concept "apartment dog". This problem is accentuated if there are not a significant number of examples available, or if examples are "costly" to obtain. Consider one example (Scott, 1987) of a system that places a can of Coca Cola in the freezer to cool and returns hours later to discover that it has shattered. There are many features (and interactions among these features) associated with this situation (e.g., the shape, color, composition, and contents of the can, the shape and color of the freezer). To empirically learn why the can shattered could require that the system observe many examples,

noting which features vary and which remain constant. Such a scenario would be impractical and expensive.

In response, many researchers argue that human concepts are organized around people's theories about objects and events in their world (e.g., Carey, 1985; Keil, 1981; Murphy & Medin, 1985; Rips, 1989). In other words, learning is *theory driven*. For example, a theory-driven system could learn why the Coca Cola can shattered from a single example, by reasoning from its prior knowledge that water expands when it freezes, that Coca Cola is composed almost entirely of water, and so on. It also might generalize this explanation to other beverages, containers, and situations involving low temperatures. Similar motivations for theory-driven processing have led machine learning researchers to focus on *explanation-based learning* (Mitchell, Keller, & Kedar-Cabelli, 1986; DeJong & Mooney, 1986; Ellman, 1989; Mooney, this volume). Typically, an explanation-based system uses its background knowledge (in the form of a theory) to explain or prove why a training example is a member of a given category. It then generalizes the explanation so that it will apply to future examples. In both the human and machine case, theories provide biases or constraints on learning.

However, theory-based approaches also have their problems. Typically, explanation-based learning systems learn by restructuring their existing knowledge. If a learner's domain theory is incorrect, incomplete, or inconsistent, then it may incorrectly explain or fail to explain a given phenomenon (Mitchell, Keller, & Kedar-Cabelli, 1986; Rajamoney, 1986). In addition, many concepts have important non-explanatory components that are often conventional in nature (Ahn & Brewer, 1988; Mooney & Ourston, 1989). For example, features like has a white color and has a circular shape typically would not enter into an explanation for why a light bulb gives off illumination. An explanation-based learning system would fail to incorporate these features into its light bulb concept, but they are nonetheless important for identifying light bulbs.

As implied above, some of the strengths of each approach complement the weaknesses of the other. As a result, cognitive psychologists have argued for the integration of empirical and theory-driven learning (Medin & Ortony, 1989; Wattenmaker, Nakamura, & Medin, 1987) and researchers in machine learning have developed a number of systems that combine both empirical and explanation-based learning (e.g., Flann &

Dietterich, 1989; Kedar-Cabelli, 1985; Lebowitz, 1986a, 1986b; Mooney & Ourston, 1989; Pazzani, 1987, 1988; Rajamoney, 1986; Shavlik & Towell, 1989; Wisniewski, Winston, Smith, & Kleyn, 1987).

Despite the concern with combining these two learning paradigms, most current approaches do not tightly couple their interaction. In this chapter we argue that, at least in some domains, this loose coupling is inadequate. We begin our analysis by describing a standard concept learning task (rule induction) found in many psychological experiments. Next, we present several recent studies that have examined the relation between theories and data in rule formation. We then explore the ways that researchers have combined explanation-based and empirical learning. The plausibility of these approaches can be evaluated in light of our own experimental findings, which expose the close interaction between theory and similarity in concept learning. We conclude the chapter by briefly sketching a model of learning in which theory and data are more tightly coupled.

2. Rule Induction Paradigms

In a typical rule induction task, the experimenter selects a rule or concept, and participants must learn the rule based on feedback that they receive from classifying examples (e.g., Bruner, Goodnow, & Austin, 1956; Haygood & Bourne, 1965). Furthermore, participants may be told what types of rules are involved (e.g., conjunctive rules, single feature rules), so that learning involves a selection among a small set of rules. In the tasks that we will describe, however, the experimenter does not select a particular rule for the subject to learn. Rather, there are many possible rules that describe the categories and interest lies in which rules people find natural to form. The rules that people create should reveal constraints on induction that can be used to evaluate alternative models of rule formation.

2.1 Simultaneous and Sequential Induction

We will describe studies that involve two types of rule induction tasks, which closely correspond to nonincremental and incremental models discussed by Fisher and Pazzani (Chapter 1, this volume). In the *simultaneous* rule induction task, participants determine a single rule by examining a number of preclassified items that are presented at the same

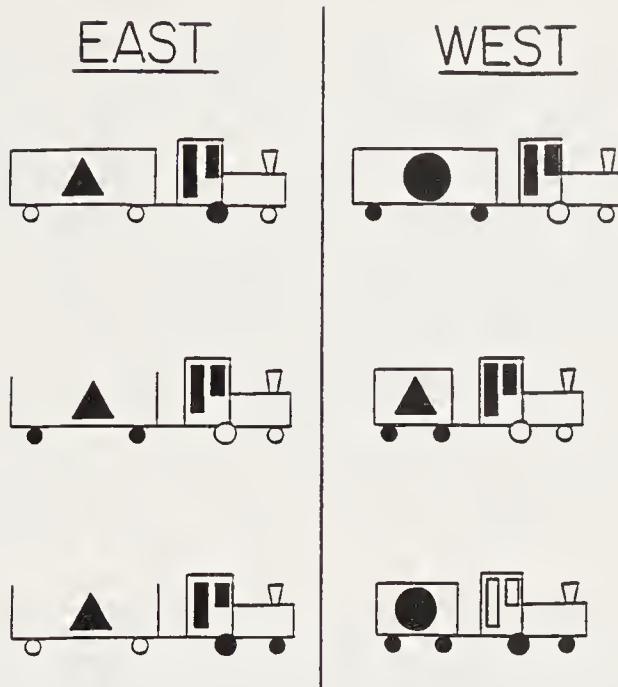


Figure 1. Stimuli used in concept induction tasks by Medin, Wattenmaker, and Michalski (1987).

time. In the *sequential* rule induction task, training items are presented sequentially rather than simultaneously and participants determine a rule after seeing each item. Furthermore, the items are not preclassified. Instead, participants determine a rule that can be used to classify items into a category, and then receive feedback on their classification. Thus, rules may change over the course of the task.

Figure 1 shows stimuli used in one simple rule induction task, taken from Medin, Wattenmaker, and Michalski (1987). Here, a set of trains have been preclassified as Eastbound or Westbound. There are many possible rules that apply to these categories. Rules for the Eastbound trains include "long car and triangle load", "white car wheels or open car top", "long car and not circular load", "open car top or engine with one white wheel", and so on.

One reason that the rule induction task is of interest is that alternative learning systems will induce different rules and one can compare these rules to those that people develop. For example, Quinlan's (1986) ID3 would favor simple disjunctive rules, such as white wheels or open car, for the examples shown in Figure 1; in contrast, Michalski's (1983a,

1983b) INDUCE would favor the rule long car and triangle load, though for other category structures INDUCE might yield a disjunctive rule. Medin et al. (1987) found that people were far more likely to develop conjunctive rules than disjunctive rules for the trains shown in Figure 1. The major difference between the rules formed by INDUCE and the participants was that people were somewhat more likely to begin with a simple rule that was complete (i.e., applied to all positive examples) but inconsistent (i.e., also applied to nonmembers). They would then refine the rule to make it both complete and consistent (i.e., apply to all members and only to members of the category). For example, people might start with a rule like Eastbound trains have triangle loads, then notice the counterexample and amend their rule to Eastbound trains have a triangle load and not a short car (see Figure 1).

2.2 Using Theories to Determine the Feature Space

In both cognitive psychology and machine learning, researchers often investigate learning by providing intelligent systems with a space of well-defined features that describe one or more training items. Learning is viewed as selecting an appropriate combination of features from this space. This view is especially common in traditional empirical learning systems (see Fisher & Pazzani, Chapter 1, this volume). However, a crucial problem in learning involves deciding the units of analysis or constituents upon which to invoke learning (Medin, Wattenmaker, & Michalski, 1987). That is, learning involves not only selecting features from a feature space but also determining that feature space. This problem typically is solved for the learning system by the programmer, who presents the system with predefined, well-specified, unambiguous constituents. How does one determine the feature space upon which learning operates?

Explanation-based learning provides a starting point for exploring how theories determine the feature space (Mooney, this volume). Typically, an explanation-based system is given a training item, described as a set of features (e.g., a particular item with features such as weight of item is light and item has a handle) and a functional specification (e.g., one can drink from the item). Using a theory, it deductively proves that or explains why the particular item meets this functional specification. The explanation is actually a tree whose root is the functional specification and whose leaves are a subset of the features of the

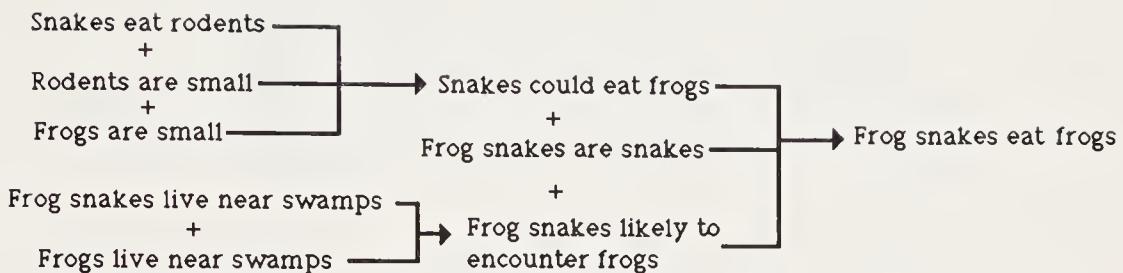


Figure 2. A plausible theory for why frog snakes eat frogs.

training example. The intermediate nodes of the tree are additional features that explanation-based learning has deduced about the item.

As we will suggest in this chapter, people's theories and how they use them to determine features is quite different from the approach taken in explanation-based learning. For one thing, an explanation-based system typically views a theory as a well-defined set of rules that are used to construct a deductive explanation (e.g., Mitchell et al., 1986; Yoo & Fisher, this volume). In contrast, people's everyday theories contain a mixture of fact and fiction. A person's theories may combine scientific principles, stereotypes, and informal observations of experiences. Furthermore, people use these theories to construct plausible rather than deductive explanations (Collins & Michalski, 1989). For example, suppose you read about a novel reptile called a *frog snake*, which was found near swamps in South America. You might reasonably conclude that frog snakes eat frogs. Figure 2 illustrates a simple theory and explanation for why this might be the case. Some of the reasoning may follow from informal observations. For example, you may believe that snakes generally eat rodents simply because you saw a TV program in which a snake ate a mouse. Thus, the explanation is plausible rather than deductive, and it could well turn out that frog snakes do not eat frogs. Rather, they might be called frog snakes because they croak like a frog or because they have bulging, frog-like eyes.

How might one study theories in categorization tasks? One approach involves using complex rather than simple stimuli. The relatively simple train stimuli of Figure 1 have the virtue that they can be used to directly compare the rules formed by induction programs with those formed by people. The constituents (features) of the trains that a programmer would present to induction programs are probably the same ones that people would consider in forming their rules. The stimuli have relatively few, unambiguous constituents (e.g., wheels, color of wheels, and type of load).

However, these virtues are also limitations: a small, unambiguous feature set is often an unrealistic presumption about a domain. Additionally, people are very unlikely to have elaborate theories about the behavior and appearance of Eastbound and Westbound trains. Therefore, we have shifted our attention to more complex stimuli that are likely to involve people's theories and to highlight the problem of determining a feature space.

In particular, the category items for our experiments are children's drawings of people, shown in Figures 3, 4, and 5. The drawings, taken from Koppitz (1984) and Harris (1963), were produced by children who were administered the "draw-a-person test". In this task, children are instructed to draw "one whole person". This test is one tool used in psychodiagnostics and IQ assessment, and it is a fairly reliable shorthand indicator of emotional problems and of intelligence in young children (Goodenough & Harris, 1950; Harris, 1963; Koppitz, 1984).

Given these more complex stimuli, one simple way of teasing out theoretical preconceptions is by providing subjects with meaningful labels for the categories (e.g., Adelman, 1981; Muchinsky & Dudycha, 1974; Wattenmaker, Dewey, Murphy, & Medin, 1986; Wisniewski & Medin, 1991; Wright & Murphy, 1984). The purpose of providing such labels is to activate theories or prior expectations that may guide rule induction, in much the same way that functional specifications do in explanation-based learning. We can contrast *theory-guided* rule induction to the situation described above, in which the category labels have little meaningful content and therefore theories are less likely to be activated.¹ We

1. Prior expectations also could be triggered by information in the training instances. For example, even without the label done by creative children, a person might believe that a particularly realistic, detailed drawing was done by a creative child. As a result, that person might hypothesize that drawings of a category were done by such children. However, this was seldom the case.

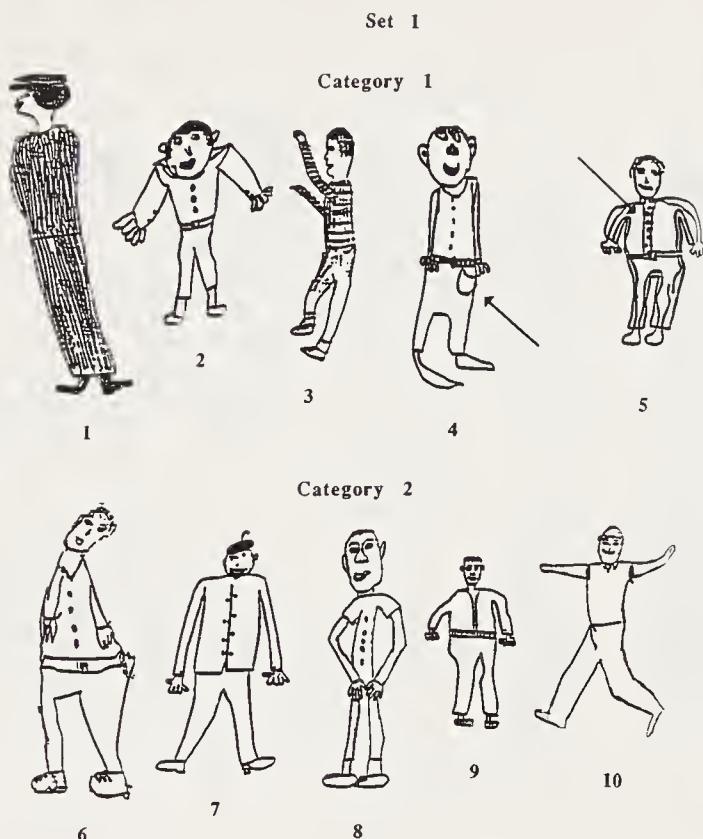


Figure 3. Drawings by "high" and "low" IQ children in Study 1, and by "farm/city" and "creative/noncreative" children in Study 2.

will call this situation *empirical* rule induction. The latter is more typical of categorization studies in psychology (e.g., Hintzman, 1986; Medin, Wattenmaker, & Michalski, 1987; Nosofsky, 1986) and a number of machine learning programs (Fisher, 1987; Michalski, 1983b; Mitchell, 1982; Quinlan, 1986; Winston, 1975).

2.3 Supervised Versus Unsupervised Learning

The rule induction tasks that we have described are supervised. That is, the experimenter indicates to the participant the category in which an item belongs. In contrast, *unsupervised* tasks do not involve explicit feedback from an experimenter (or programmer). For example, conceptual clustering systems (e.g., Anderson & Matessa, this volume; Fisher, 1987; Hanson & Bauer, 1989) partition items into categories based on some evaluation function, forming a concept or rule for each category. Learning is unsupervised in the sense that a programmer does not give the system feedback about whether or not an item belongs in a category.

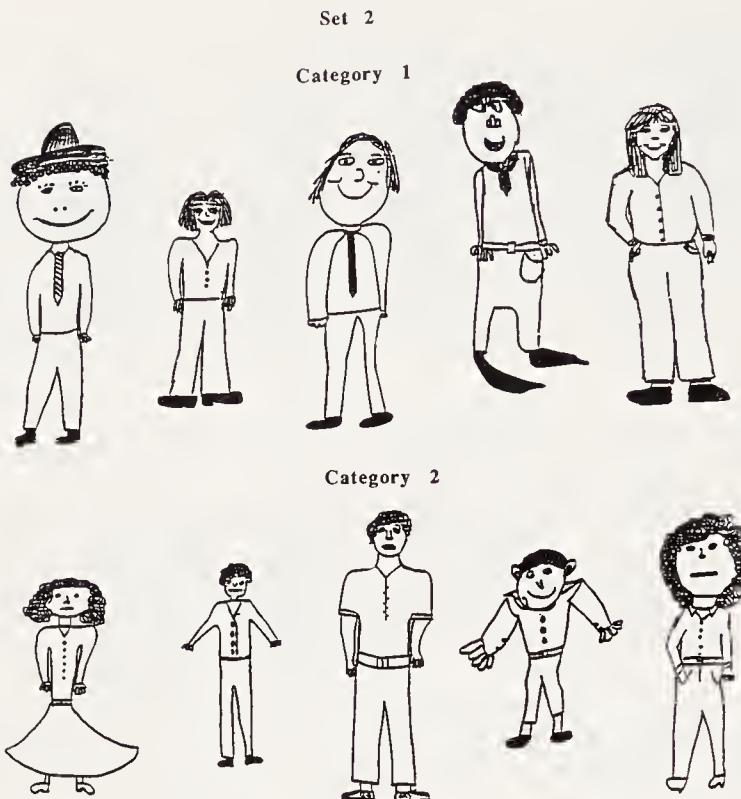


Figure 4. Drawings by “creative” and “noncreative” children.

In general, how important is the distinction between supervised and unsupervised learning? Langley (1987) suggests that unsupervised learning characterizes much of human learning. He notes that before they can understand language, children form useful concepts primarily by direct interaction with their environment, rather than by advice given to them by other humans. On the other hand, it is clear that in many domains, human learning is supervised. Learning medical diagnosis, nuclear reactor monitoring, mathematical problem solving, and guitar playing are just a few of the domains in which people are explicitly informed of the categories in which items belong.

We believe that the distinction between supervised and unsupervised learning is really addressing a deeper issue: the nature of feedback and credit assignment. All systems eventually must use their concepts; otherwise, there would be little point in learning them. Importantly, some type of success or failure (i.e., feedback) will be associated with that use. Thus, it seems reasonable that intelligent systems should take advantage of feedback and assign credit to concepts that lead to success

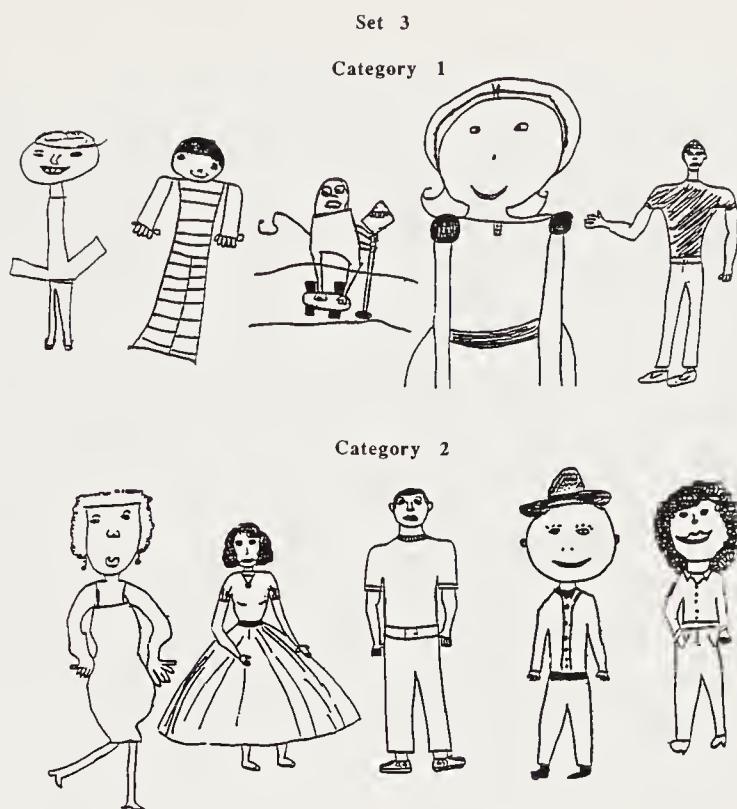


Figure 5. Drawings by “well-adjusted” and “emotionally-disturbed” children.

and blame to concepts that result in failure. In any event, we believe that the distinction between supervised and unsupervised learning is not crucial to the studies that we will describe. In these studies, even with explicit feedback, credit assignment was not at all straightforward. As we describe later, our stimuli were more complex than those typically found in psychological studies and in computational simulations. Determining the reason a group of items belonged to the same category was not an easy task. This was especially true since the validity of a person’s hypothesized rule could vary with its level of abstraction. For example, consider a person who believes that a drawing is detailed because it has hair, eyes, and shoes. At a high level of abstraction, the person might correctly assume that the drawing belongs to a particular category because it is detailed. At the same time, the person’s more specific reasons for why the drawing is detailed (i.e., hairs, eyes, and shoes) might be incorrect. (The drawing might be detailed because it has curly hair, eyebrows, and shoelaces.) In addition, a goal of our second study was to examine people’s rule modification strategies. In this case, explicit negative feedback was desirable.

3. Study 1: Simultaneous Rule Induction

The first study explored the effects of theoretical knowledge, empirical knowledge, and their interaction in a simultaneous rule induction task. In this task, participants determined a classification rule by examining a number of the members of a category that were presented at the same time. The categories consisted of the three sets of children's drawings shown in Figures 3, 4, and 5. Two groups of participants, the THEORY 1 and THEORY 2 groups, learned about pairs of categories with meaningful labels (indicated in Figures 3, 4, and 5). In contrast, the STANDARD group learned the identical pairs of categories, but were given irrelevant labels. The basic task was straightforward. Participants examined each set of two categories and wrote down a rule that distinguished the members of one category from those of the other. In addition, the THEORY 1 and THEORY 2 groups differed in that the labeling of their categories was counterbalanced. For example, in Figure 3, the THEORY 1 group was told that Category 1 was drawn by low IQ children and that Category 2 was drawn by high IQ children. For the THEORY 2 group, this labeling was reversed.

3.1 Goals of the Study

This study had two purposes. First, it should provide a simple demonstration of whether people use integrated or nonintegrated learning. Because both empirical and theoretical knowledge are available, it should tell whether learning is primarily empirical, theory guided, or some combination of the two. A number of findings are consistent with each of these possibilities. For example, if the THEORY groups use only empirical learning and ignore theoretical knowledge (activated by the meaningful labels), then their rules should be similar to those of the STANDARD group. This would occur because all groups learn about *identical* categories and therefore are provided with the *same* empirical knowledge. There are a number of rules that can be found empirically (i.e., without recourse to theoretical knowledge) that distinguish the categories in each set. For example, in Figure 4, the rule "smiling and hands at side" distinguishes the members of Category 1 from those of Category 2.

On the other hand, if people use only their theories and ignore empirical evidence, then the THEORY 1 and THEORY 2 groups should produce similar rules for identically labeled but different categories. For

example, the rules of the THEORY 1 group for Category 1 in Figure 3 should be very similar to those of the THEORY 2 group for Category 2. This would occur because both groups were told that these different categories were drawn by low IQ children. In this case, *identical* theoretical knowledge should have been activated in these groups, even though it is present in the context of *different* empirical knowledge.

Although the two situations described above are logical possibilities, they represent rather extreme views of how people utilize theoretical and empirical knowledge. Each view suggests that with both sources of knowledge available, people ignore one source and only use the other. More than likely, however, people use some combination of theoretical and empirical knowledge. There would be good evidence for such integrated learning if the rules of the STANDARD and THEORY groups were different and if the rules for the THEORY 1 and THEORY 2 groups were different for identical but differently labeled categories.

A second purpose of the study was more exploratory in nature. We were interested in the effects of theoretical expectations on rules. Specifically, how might the rules of people with expectations differ from those of people without such expectations? To explore this issue, we compared the rules of the THEORY groups to those of the STANDARD group.

3.2 Method

The subjects were 40 undergraduates (male and female), attending the University of Michigan, who received course credit for participating in the study. A participant was assigned randomly to a THEORY group or the STANDARD group. Each participant saw three sets of children's drawings, with each set divided into two equal-sized categories of five drawings (as shown in Figures 3, 4, and 5).

Participants in the study were instructed to write down a rule for each set that someone else could use to accurately place the drawings in their respective categories. They also were told that someone else should be able to use the rule to decide whether a new (not yet seen) drawing belonged to one or the other category. For the STANDARD group, the two categories in each set were simply labeled Category 1 and Category 2. For the THEORY groups, the two categories in each set were given meaningful labels, corresponding to the functional specifications typically given to explanation-based learning systems. We used different meaningful labels to explore a variety of theories that might

be activated. Specifically, the two categories in the first set (Figure 3) were labeled drawn by children with high IQ/low IQ, those in the second (Figure 4) were labeled drawn by creative/noncreative children, and those in the third set (Figure 5) were labeled as drawn by emotionally disturbed/well-adjusted children. As noted, the labeling of categories was counterbalanced in the THEORY groups. That is, for the THEORY 1 group, a category was given one of the two meaningful labels, and for the THEORY 2 group, the same category was given the other meaningful label.

3.3 Results

To explore possible learning strategies (i.e., empirical only, theory only, or some combination), we compared the rules of the STANDARD group to those of THEORY groups and the rules of the THEORY 1 group to those of the THEORY 2 group. In general, there were differences between the THEORY groups and the STANDARD group and between the two THEORY groups. These differences suggest that people use some combination of theoretical and empirical knowledge to formulate rules. We describe these results below.

3.3.1 EFFECTS OF THEORETICAL KNOWLEDGE ON RULE LEARNING

To examine the impact of theoretical knowledge, we compared the similarity *between* the rules of the THEORY groups and those of the STANDARD group to the similarity of the rules *within* the STANDARD group for the same category. If theoretical knowledge affects rule formation, then on the average, a rule from the THEORY group and one from the STANDARD group should be more different than two rules from the STANDARD group. A rule from the THEORY group should be based on both theoretical and empirical knowledge, whereas a rule from the STANDARD group will be based on only empirical knowledge. In contrast, any two rules from the STANDARD group will be based on the same empirical knowledge (if they apply to the same category).

We measured the similarity of rules by calculating their feature overlap. More specifically, for each rule in the THEORY groups, we calculated the percentage of rules in the STANDARD group that matched at least one constituent in the THEORY-group rule. We then averaged these percentages. Across categories, the average percentage overlap between

the rules of the THEORY and STANDARD groups was 15%. To measure the similarity of the rules in the STANDARD group, we randomly divided the rules of a category into two equal-sized groups. For each rule in one group, we calculated its average percentage overlap with the rules of the other group. Across categories, the average percentage overlap of rules within the STANDARD group was 26%. The fact that the similarity of the rules of the THEORY groups and STANDARD group (15%) was less than the similarity within the STANDARD group (26%) suggests that theoretical knowledge affected the rules of the THEORY groups.

One curious issue is why the rules of the STANDARD group differed from each other so much for the same category. Although the features of these rules overlapped more with each other than with the rules of the THEORY group, the overlap was only 25%. There are probably several reasons for this result. First, the stimuli are relatively complex and there was a wide range of features that people could use in their rules. Second, as detailed later, very few features were complete and consistent, in that they applied to all members of a category and to none of the contrast category. In a sense then, many features may have been "equally good" for determining category membership and participants were not biased to use any particular one in forming their rules.

3.3.2 EFFECTS OF EMPIRICAL KNOWLEDGE ON RULE INDUCTION

To study the effect of empirical knowledge, we examined the rules of the THEORY 1 and THEORY 2 groups for different categories that were identically labeled. In these cases, identical theoretical knowledge but different empirical knowledge should be available to these groups. In general, the rules of the groups were very different, suggesting that people were not just using theoretical knowledge. In fact, the average percentage overlap between the rules of the THEORY 1 and THEORY 2 groups for identically labeled categories was only 2.4% using the measure described above.

To get some appreciation of these differences, consider the set of categories shown in Figure 5. For Category 1, the THEORY 1 group mentioned lack of detail or simplicity in four of their eight rules, when the category was labeled drawn by emotionally disturbed children. In contrast, the THEORY 2 group failed to mention lack of detail in any of their rules for the identically labeled Category 2. Instead, the THEORY 2 group mentioned that the drawings of Category 2 were

accurate or precise (three of nine rules) or that they depicted normal, conventional, or regular looking people (five of nine rules). None of these characteristics was mentioned in any of the rules of the THEORY 1 group for the identically labeled Category 1. In short, empirical knowledge also appeared to affect the rules of the THEORY groups. Based on these analyses, it appears that people in the THEORY groups used some combination of theoretical and empirical knowledge in formulating rules. We now take a closer look at how people's theories affected rule formation.

3.3.3 THEORY GROUP VERSUS STANDARD GROUP RULES

There were some striking differences between the rules of the THEORY groups and those of the STANDARD group. First, we noticed that the two groups differed in the kinds of features that they used in their rules. We classified the rules into three major types, based on the types of features that they included. *Concrete* rules consisted of simple features that were easily observable in the drawings (e.g., eyebrows, white shoes, buttons, ears, striped clothes). *Abstract* rules consisted of features that were more complex, higher level, or less perceptual (e.g., normal, bizarre, detailed, well-proportioned, hastily drawn). *Linked* rules consisted of both concrete and abstract features (e.g., cross-eyes, abnormalities, proportion).

Two examples of each rule type for categories 1 and 2 of Figure 3 illustrate the differences:

- *Concrete*: “buttons or stripes on their shirts and dark, thick hair” (for Category 1) and “facing forward and showing ears and not showing teeth” (for Category 2);
- *Abstract*: “relaxed and free flowing” (for Category 1) and “look more normal” (for Category 2);
- *Linked*: “added more details such as teeth, extensive shading, and drawing the body underneath the clothes” (for Category 1) and “limbs proportional to rest of the body, some drawings demonstrate dimensional representation (e.g., one leg crossing over the other), more detailed dress (e.g., shoelaces)” (for Category 2).

Table 1 presents the percentages of each type for the two groups. In using linked rules, people often listed several different features as exam-

Table 1. Frequency of rule types among the STANDARD and THEORY groups.

	STANDARD GROUP	THEORY GROUP
% CONCRETE RULES	81	35
% ABSTRACT RULES	16	37
% LINKED RULES	3	28

ples of a more abstract feature. The linked rules provide evidence that the THEORY group *operationalized* abstract information (e.g., Mostow, 1983) by linking it to concrete features in the data.

If this account of operationalization is correct, then one might ask why the THEORY groups produced fewer linked rules than abstract rules. One reason is that there may have been many different ways to operationalize an abstract feature among the examples. People may have written their rules at the level of description that was common to all examples. A similar strategy is used by some systems that combine explanation-based and empirical learning, such as Flann and Dietterich's (1989) IOE and Yoo and Fisher's (this volume) EXOR system. Second, the rules may have been less detailed because of such subtle factors as the amount of space provided for writing and the time spent writing. In a previous study involving meaningful category labels (done with Glenn Nakamura), almost all the rules produced were linked. In that study, participants had more space and time to write down rules.

A second observed difference was that the rules of the STANDARD group generally were more syntactically complex than those of the THEORY group. To measure syntactic complexity, we counted the number of properties and logical operators (conjunction, disjunction, or negation) used in each rule. For the STANDARD group, 59% of the rules contained more than two features, compared to 44% of the rules for the THEORY group. The STANDARD group used an average of 2.42 logical operators per rule, compared to 1.60 for the THEORY group.

To summarize, there were some clear differences between the kinds of features occurring in rules of the THEORY groups and those of the STANDARD group. The latter primarily used concrete rules, seldom used abstract rules, and very rarely used linked rules. In contrast, the THEORY groups used many fewer concrete rules, and many more abstract

and linked rules, than did the STANDARD group. The groups also differed in terms of the syntactic complexity of their rules. In general, the STANDARD group used rules that contained more features and more disjunctions, conjunctions, and negations of properties.

3.4 Discussion

Our results suggest that the THEORY group combined both theoretical and empirical knowledge in constructing their rules. In particular, it appears that theoretical knowledge provides certain biases to the learner. One way that theoretical knowledge strongly affects rule formation is by activating a space of abstract features and hypotheses. People are then biased to search for evidence in the data that supports these features and hypotheses. Informal examination of the linked rules suggests that this was the case. For example, consider the following linked rule:

High IQ children are capable of drawing people from a profile angle; their drawings show detail such as pockets and collars in their figures.

Figure 6 illustrates that the label, drawn by high IQ children, may have activated the hypothesis that intelligence is required to draw well and, therefore, more intelligent children will draw better. This hypothesis, in turn, might suggest that better drawings will capture abstract features such as details (e.g., pockets and collars) and difficult aspects of drawing people (e.g., profile angles).

By activating high-level, abstract features, theories also may bias people to treat lower-level features as equivalent, when in other contexts they would be considered different. In other words, theories allow people to generalize across different features. For example, in a neutral context, the features pockets and collars appear quite different. However, as illustrated in Figure 6, the theory-activated expectation that drawings will show detail lets one generalize across these features and consider them similar.

Most empirical learning systems do not generalize across features and do not represent features at multiple levels of abstraction, but there are exceptions. Some inductive systems use various heuristics to treat features equivalently. For example, INDUCE (Michalski, 1983a) uses an IS-A hierarchy to transform the features Chicago, De Kalb, and Peoria into the feature Cities in Illinois. However, note that such a hier-

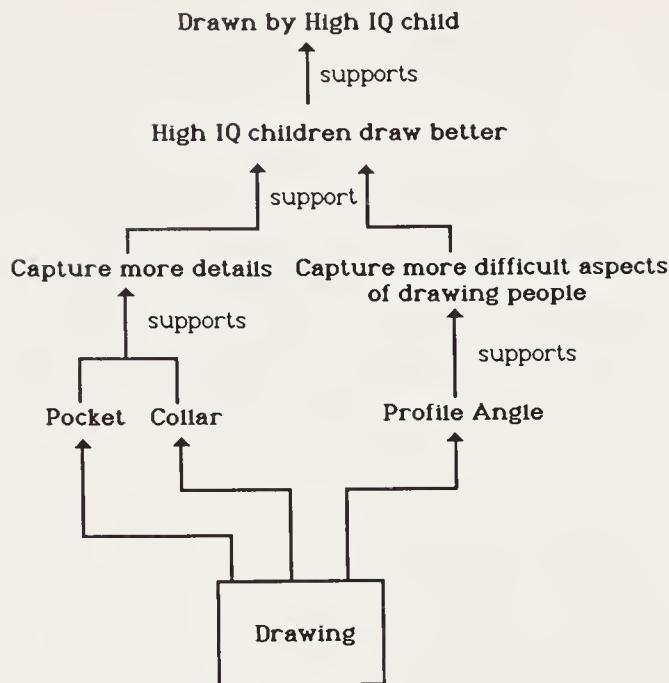


Figure 6. Plausible explanation for a drawing by a high IQ child.

archy is built into the system. In our studies, people probably created such hierarchies dynamically, through an interaction of empirical and theoretical knowledge. It seems unlikely that our subjects (typical undergraduates) had prestored, hierarchical knowledge about drawings of people done by emotionally disturbed children.

Whether or not a system generalizes across features has important effects on how it carries out induction. To take one simple example, consider the feature descriptions of four children's drawings shown in Figure 7. These descriptions correspond to actual drawings done by well-adjusted and emotionally disturbed children.² Consider two induction systems whose task is to cluster the descriptions into categories. The first system groups descriptions based on a measure of surface similarity like *category utility* (Gluck & Corter, 1985; Fisher & Pazzani, Chapter 1, this volume) and does not generalize across features. The second system groups descriptions based on theoretical expectations

2. In children's drawings, one indicator of emotional problems is the omission of body parts (Koppitz, 1984). In the drawings done by children with emotional problems, note that one description includes the features **hands missing** and **nose missing**, whereas the other description includes the features **feet missing** and **neck missing**.

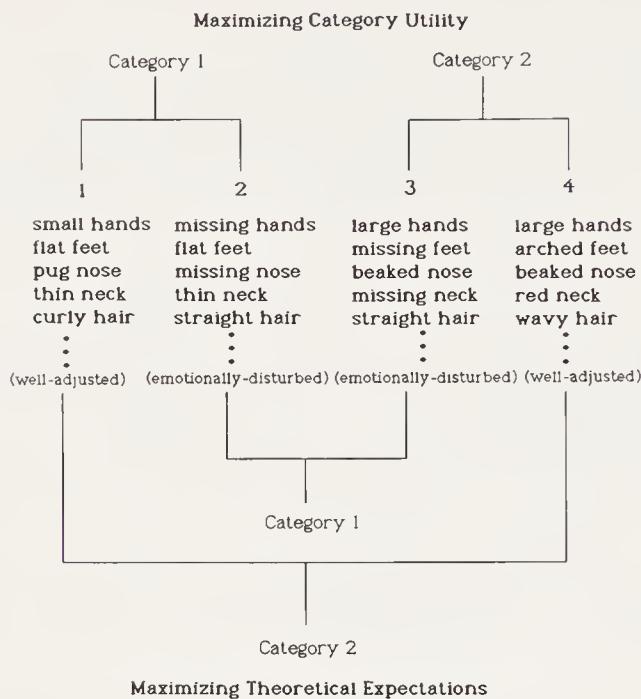


Figure 7. Alternative classifications of drawings by well-adjusted and by emotionally disturbed children.

about well-adjusted and emotionally disturbed children and generalizes across features. These systems will cluster the descriptions in different ways, as shown in Figure 7. Specifically, by grouping drawings 1 and 2 together and drawings 3 and 4 together, the first system maximizes surface similarity in terms of category utility. In contrast, the second system will group descriptions 2 and 3 together and 1 and 4 together. Such a system can treat the features hands missing, nose missing, feet missing, and neck missing as equivalent, because they are all examples of the same high-level feature, missing body parts. This feature, in turn, is an example of deviation from normal, which in turn provides evidence for the category emotionally disturbed. On the other hand, the second system will group descriptions 1 and 4 together, as these examples do not deviate from the normal, thus implying that both were done by well-adjusted children. Interestingly, at the concrete feature level, these descriptions have few features in common, yet were placed in the same category.³

3. Of course, one could include missing body part as a feature in the drawings by the emotionally disturbed children, and then highly weight this feature during categorization. In this case, a model that maximizes category validity could derive the alternative, theory-based grouping, but there would be no clear

In contrast to the THEORY groups, the STANDARD group did not operate with the biases described above. Its members seldom used abstract rules or linked rules involving operationalization, suggesting that abstract features and hypotheses were absent. They did not generalize across features. Instead, this group appeared to focus on finding any set of simple, observable features that could discriminate the categories.

The STANDARD group constructed rules that were considerably more complex than those typically found in other studies of empirical learning. For example, in the study involving the train stimuli shown in Figure 1, Medin et al. (1987) found that 75% of the rules were based on a single property, a conjunction of two properties, or a disjunction of two properties. Only 41% of the rules of the STANDARD groups could be classified as one of these three types. Instead, many rules involved complicated combinations of conjunctions and disjunctions, with three or more properties. Most likely, the rules of the STANDARD group were more syntactically complex because they relied on concrete features that were neither consistent nor complete. A feature is consistent if it is present in (at least) some members of a category but absent in all members of the contrasting category. A feature is complete if it is present in all members of the category. Simple rules, like the three types noted in the Medin et al. study above, typically are constructed out of features that have these characteristics. A correct rule can be based on a single feature if that feature is both complete and consistent. It can be based on conjunction of two features if each feature is complete and their conjunction is consistent. Finally, a rule can be based on a disjunction of two features if each feature is consistent and their disjunction is complete. Looking at the train stimuli shown in Figure 1, one can see that there are many features that have one or both of these characteristics. Not surprisingly then, a large majority of rules in the Medin et al. (1987) study were simple rules.

For the categories used in our drawings, however, there were few features with one or both of these characteristics. Specifically, for each pair of categories, we examined the completeness and consistency of 40 concrete features that were selected arbitrarily from the rules given by subjects. For the three category pairs, none of the features was

justification for including the feature and weighting it highly. Our approach includes the feature because different, low-level features are viewed as similar, high-level features. Their importance comes from prior expectations about the category.

complete and consistent, 32% were consistent, and 11% were complete. In addition, for those features that were consistent, almost all of them (87%) were true of only one or two (out of five) category members. Also, the most syntactically complex rules were those for the categories shown in Figure 3, which had fewer complete or consistent features than other category sets. Specifically, none of the features was complete for these categories, and all those that were consistent were true of only one or two category members. As a result, a large majority of our participants' rules were more complex than those found with the train stimuli.

This study has shown that by activating an abstract feature space, theoretical knowledge provides certain biases that are used in rule induction. Because of these biases, the THEORY group appeared able to consider fewer rules that potentially describe a category than the STANDARD group. This might be the case for two reasons. First, some rules involving low-level features may be eliminated, since only a subset of low-level features will be relevant to an abstract hypothesis. Thus, a person in the THEORY group might not consider the features hands at side and smiling in a rule, since they would not be relevant to the abstract feature detail. Second, abstract hypotheses let people consider fewer competing rules during learning, since they can treat different features as equivalent. For example, a person in the THEORY group might view the features has shoelaces and has buttons on jacket as support for the single rule drawings will be detailed. In contrast, after examining a subset of category members that contain both features, someone in the STANDARD group might consider both features as distinct, potential rules for the category.

4. Study 2: Sequential Rule Induction

Although the first study demonstrated how theoretical knowledge guides rule induction, its methodology was limited in several important ways. Notably, people seldom acquire rules or concepts by examining a number of preclassified examples at the same time, as they did in the first study. In more realistic learning situations, rule induction is typically incremental and items are not preclassified. Rules are acquired gradually and modified in response to new examples. Second, although the previous methodology lets one assess whether or not theoretical and empirical knowledge interact, it does not clearly assess *how* they interact.

Our second study addresses this interaction by examining the development and modification of rules as training items are presented sequentially. We begin by giving people meaningful labels for the categories, then ask them to give us a "first-impression" rule of the category before they have seen any items. We assume that this rule primarily is based on a person's theories. We can then examine how the empirical evidence (i.e., training items and feedback), in conjunction with people's theories, causes people to modify their rules and affects the interpretation of succeeding training items. In this way, we can gain some insight into how empirical knowledge interacts with theoretical knowledge and vice versa. In this study we also examine in more detail the kinds of hypotheses that people formulate. We will show how different hypotheses (activated by different theories) dramatically affect the interpretation and selection of the particular features that people use in their rules.

4.1 Method

The subjects were 40 undergraduate students (male and female), attending the University of Illinois, who received course credit for participating in the study. A participant was randomly assigned to one of two groups of 20 subjects. Both groups learned about the same two categories (those shown in Figure 3). However, the groups were given different labels for the categories. One group (the FARM/CITY group) was told that the categories were drawings done by farm kids and drawings done by city kids. The other group (the CREATIVE/NONCREATIVE group) was told that the categories were drawings done by creative kids and drawings done by noncreative kids.

Students learned about the categories in the following manner. First, they were given the names of the categories. Then, before seeing any of the training items, they wrote down their initial rules for classifying the drawings. Next, students were shown the drawings one at a time, presented in a random order. A given drawing was not preclassified, and students had to decide to which category it belonged. After this decision, students wrote down the rule that they used to determine category membership. Then, they were given feedback on the correctness of their decision. After this feedback, students were given the option of modifying their rule. This process was repeated for each of the ten drawings. Afterward, students wrote down their overall rule for classifying drawings into the two categories.

4.2 Results

We discuss the results of this study in terms of two broad questions. First, how did theoretical knowledge affect the use and interpretation of the training data? Second, how did the training instances, in turn, affect the use of theoretical knowledge? Examining these questions provides some insight into how empirical and theoretical knowledge interact. Later, we sketch a model of this interaction and compare it to existing machine learning systems that have combined empirical and theory-based learning.

4.2.1 EFFECTS OF THEORETICAL KNOWLEDGE ON TRAINING DATA

Our results indicate that the two groups, given different labels, used different theories in learning about the same sets of drawings. In turn, these different theories had three important effects on how people processed the data. First, different theories caused people to selectively attend to different features in the training items. Second, and more striking, different theories caused people to interpret the same data differently. For example, people with different theories sometimes interpreted the same part of a drawing as a different feature. Third, like the first study, theories allowed people to view different features as similar at a higher level of abstraction.

Different labels activate different theories. To assess whether or not different labels activated different theories, we compared the initial rules of the two groups before any training items were presented. Differences in theoretical knowledge should be most apparent in these rules, since they were formulated in the absence of empirical evidence. In general, the initial rules of the groups were quite different, suggesting that they operated with different theories during the task. An examination of the protocols revealed that the initial rules of the CREATIVE/NONCREATIVE group were of two major types. Seventy percent of this group stated that creative kids would draw pictures with more detail and add more things to their drawings, whereas noncreative kids would draw pictures that were more basic and simple. For example, one subject stated that “creative kids will draw more detail — like eyelashes, teeth, curly hair, shading and coloring in. Noncreative kids draw more stick-figurish people.” Twenty-five percent of this group stated that creative kids would draw pictures that were more imaginative and unusual than the pic-

tures drawn by noncreative kids. For example, one subject stated that "creative kids would stray from basics, show imagination — noncreative kids will draw regular, plain clothes, straight hair, basic facial features."

On the other hand, most of the FARM/CITY group stated that drawings would reflect the kinds of people that farm and city kids encountered in their environments and that these people could be identified by their clothing. Seventy-five percent of this group mentioned differences in the kind of clothing that the two types of drawing would have. For example, one subject stated that "farm kids will draw people with overalls, straw or farm hats. City kids will draw people with ties, suits."

Selective attention to features. It was clear from the protocols that the two groups selectively attended to different features in the input. Furthermore, we can relate these differences to the different theories of the groups. To show this, we examined two large classes of features that appeared in the rules. These classes were body terms like arms, body, dancing, eyes, face, figure, hair, proportion, teeth, and running, and clothing terms such as belt, button, clothing, dress, jacket, pants, shoes, sleeves, suit, tie, and wearing. For the two groups, we counted the number of different features (i.e., types) that people used in their rules as well as the number of occurrences (i.e., tokens) of each feature.

Table 2 shows the number of occurrences and different types of body terms and clothing terms listed by the two groups. The rules of the CREATIVE/NONCREATIVE group had almost three times as many occurrences of body terms (487 versus 173) and about twice as many types (60 versus 29) as the rules of the FARM/CITY group. In contrast, the rules of the FARM/CITY group had almost three times as many occurrences of clothing terms (295 versus 116) and slightly more types (25 versus 19) than the rules of the CREATIVE/NONCREATIVE group.

Why did the different theories cause the CREATIVE/NONCREATIVE group to consider mostly body features and the FARM/CITY group to consider mostly clothing features? One apparent reason for this difference is related to the different hypotheses that were initially activated by people's simple theories.⁴ Recall that the FARM/CITY group hypothesized that farm and city children would draw people that they typically encountered in their different environments. Differences in the type of

4. The difference also is related to other hypotheses that the two groups formed over the course of category learning and not just to the initial hypotheses.

Table 2. Number of occurrences of body terms and clothing terms in Study 2.

	FARM/CITY	CREATIVE/NONCREATIVE
	Body Terms	
DIFFERENT TYPES	29	60
OCCURRENCES	173	487
	Clothing Terms	
DIFFERENT TYPES	25	19
OCCURRENCES	295	116

clothing (as opposed to body parts) would indicate whether the person was from a farm or a city. As a result, the FARM/CITY group attended more to clothing features than to body features. On the other hand, the use of body features provides plausible evidence for the two initial hypotheses entertained by the CREATIVE/NONCREATIVE group. Smaller-sized body parts (e.g., eyebrows, teeth, fingers) indicate that a drawing is detailed as opposed to simple (at least for a child's drawing). Similarly, references to body movement (e.g., dancing, climbing) provide evidence that a child's drawing is unusual as opposed to ordinary.

Interestingly, when the CREATIVE/NONCREATIVE rules included clothing items, they tended to refer to relatively small items of clothing (e.g., shoelaces, buttons, belts, pockets). In fact, the average size of clothing items mentioned by the CREATIVE/NONCREATIVE group was considerably less than that of the FARM/CITY group. Small clothing items provide evidence for detail in the drawings, one of the two initial hypotheses of the CREATIVE/NONCREATIVE group.

Interpreting the same data differently. We also examined the features that people mentioned about a given drawing. The most striking finding was that subjects sometimes interpreted the same part of a drawing as a different feature. Furthermore, it appeared that people's theoretical expectations determined the nature of these features. We can illustrate this process using examples from Figure 3. For drawing 5, a person in the CREATIVE/NONCREATIVE group referred to the part indicated by the arrow as buttons. The person mentioned this feature as evidence

of detail, which implied that the drawing was done by a creative child. On the other hand, a person in the FARM/CITY group interpreted the same part of the drawing as a tie. The person mentioned this feature as evidence that the drawing depicted a business-person, which implied that the drawing was done by a child from the city. As a second example, a person in the CREATIVE/NONCREATIVE group stated that drawing 3 of Figure 3 depicted someone dancing. This feature, in turn, showed imagination and implied that the drawing was done by a creative child. In contrast, a person in the FARM/CITY group interpreted the drawing as someone climbing in a playground. This feature, in turn, implied that the person in the drawing was from the city and therefore that the drawing was done by a child from the city. As a final example, a person in the CREATIVE/NONCREATIVE group interpreted the part of drawing 4 indicated by an arrow as a pocket. The person mentioned this feature as evidence of detail, which implied that the drawing was done by a creative child. In contrast, a person in the FARM/CITY group interpreted the same part of the drawing as a purse. This feature implied that the drawing depicted a city person and therefore was drawn by a child from the city.

Besides interpreting the same part of a drawing as a different feature, participants often interpreted a drawing as depicting different, stereotypical people. For example, in Figure 3, five people in the CREATIVE/NONCREATIVE group interpreted drawing 7 as a person from a culture different from that of America, such as a Frenchman, which implied that it was done by a creative child. In contrast, six people in the FARM/CITY group interpreted this drawing as a person from a city, such as a bellboy, which implied that it was done by a city child. No one in the CREATIVE/NONCREATIVE group mentioned that the drawing was done by a city person, and only one FARM/CITY subject mentioned that the drawing was done by a person from a different culture. As another example, in Figure 3, two subjects in the CREATIVE/NONCREATIVE group mentioned that drawing 9 was a regular, ordinary person and therefore was done by a noncreative child. In contrast, three subjects in the FARM/CITY group mentioned that the drawing was a farmer or cowboy and therefore was done by a farm child.

This type of theory-based processing was much more common in the FARM/CITY group (58 examples) than in the CREATIVE/NONCREATIVE group (21 examples). Recall that the major initial expectation of the FARM/CITY group was that children from the farm or city would draw

people from those environments. Not surprisingly, this group interpreted a number of drawings as depicting people typically found in such settings. People from the city included a cool guy from a gang, a bellboy, a child climbing on a playground, a bus driver, and a professional. People from the farm included a farmer, a cowboy, a carpenter, and a blue-collar craftsman.

4.2.2 THE EFFECT OF DATA ON THEORETICAL KNOWLEDGE

The next issue that we address is how training data caused people to revise their theories about the drawings of creative and noncreative children. Specifically, we will examine how rules are affected by training items that have been incorrectly classified. Recall that after classifying a drawing and writing down their rule, participants were given feedback on their decisions. At this point, participants had the option of modifying their rule. In general, when people were given positive feedback, they did not modify their rules. However, when given negative feedback, subjects used a number of strategies to modify their rules. Below, we describe three of these strategies and provide examples of each.

Using theory-based features in context-specific ways. At times people altered their use of a theory-based feature (e.g., detail), depending on the context. In particular, they mentioned that a feature was true of a drawing and constituted evidence for one of the categories. However, given feedback that the drawing belonged to the contrasting category, they *changed their criteria* for applying that feature. For example, one person mentioned that a drawing depicted detailed clothing and therefore that it was drawn by a creative child. But when told that the drawing was done by a noncreative child, the person changed his or her rule to “drawings done by creative children would be *more* detailed.” In this example, the person adjusted the theory-based belief that creative children will show detail. Specifically, for a drawing to be classified as creative, it must have an increased amount of detail relative to the noncreative drawing just presented.

As another example, one person stated that a drawing was done by a city child because “it looks very detailed, has colored-in places.” However, upon being told that the drawing was done by a farm child, the person reexamined the details of the drawing and stated that “drawings with detail in specific clothing is more of a rule for city kids — not

detail in body movement as this one had." As in the first example, the person has altered his or her use of the theory-based feature detail. Specifically, for a drawing to be classified as done by a city child, it must now show detailed clothing as opposed to detail in body movement.

Reinterpreting features to preserve theory-based beliefs. Given feedback that they had incorrectly classified a drawing, people sometimes reinterpreted the features that they had used in making that incorrect decision. Then, they used these reinterpreted features as evidence for the correct category. This strategy might allow people to preserve the validity of their theory-based beliefs. For example, one person mentioned that a drawing was done by a city child because it depicted a television character, and city children watch more television. However, when told that the drawing was done by a farm child, the person reinterpreted the character as one created from a farm child's imagination. Such a strategy would let the person's theory-based belief (namely, that city children watch more television) remain intact.

As another example, one person thought that the clothing in a drawing was a city uniform and was drawn by a city child. Given feedback that the drawing was done by a farm child, the person reinterpreted the clothing as a farm uniform. As before, such a strategy allows the person's theory-based beliefs to remain intact. That is, after modifying the rule, the person's belief that city children would draw people with city uniforms remains a valid hypothesis.

Shifting to evidence that supports alternative theory-based beliefs. Given feedback that they had incorrectly classified a drawing, people often considered alternative evidence for the correct category. For example, one person mentioned "stiff arms, misproportioned legs, and feet stemming out in both directions" as evidence that a drawing was noncreative. Told that the drawing was creative, the person stated that "avoiding those attributes, there is some eye for detail in the face." As a second example, one person classified a drawing as done by a city child because it depicted a "child climbing in a playground." Told that the drawing was done by a farm child, the person noted that the drawing also showed "plaid clothing." In fact, some participants in the FARM/CITY group believed that plaid clothing was typical of people from rural areas. As a third example, one person categorized a drawing as noncreative because it was not "really real in the way the head and arms don't look like they

belong to the legs." Given negative feedback, the person suggested that creative drawings have "a lot of effort put into them" and that this drawing included "facial features and extra things (belt, purse)."

4.3 Discussion

These results suggest some of the ways that theoretical expectations and empirical evidence interact. They also suggest that the problem of induction should be viewed from a somewhat different perspective than that typically seen in disciplines of cognitive psychology and machine learning. In particular, our results imply that theoretical expectations and empirical evidence can interact closely to determine the features that induction operates upon. The major evidence for this claim comes from the finding that people with different theoretical expectations sometimes interpreted the same parts of drawings as different features, or the same drawing as representing a different kind of person. Thus, theoretical knowledge affected feature interpretation. On the other hand, people clearly did not just "see what they wanted to see." The data provided constraints that allowed some interpretations and not others. For example, consider the part indicated by the arrow in drawing 4 of Figure 3. Although one subject interpreted the configuration as a pocket, therefore providing evidence of detail, the person did not interpret the configuration as shoelaces, eyebrows, or buttons, even though such features also provided evidence for detail. In the following section, we outline a speculative model of how theory-based beliefs and perceptual input interact to determine features.

Besides interacting to constrain features, theoretical expectations and empirical evidence can interact to produce the explanations that participants gave as rules. Often it was clear that an explanation for one drawing's membership in a category was affected by previous items and feedback. There were a number of ways that such information influenced the current explanation. Although not described in the results, people often constructed explanations that were similar to ones used in explaining correct classifications. They also avoided explanations that had been used in explaining incorrect classifications. These findings simply suggest that people were sensitive to previous explanations which had been successful or unsuccessful. As described in the results, people also adjusted their explanations when given negative feedback. They sometimes readjusted the criteria for their applicability or reinterpreted the features upon which they based those explanations.

5. Models of Theoretical and Empirical Learning

We have suggested that people learn about categories by combining theory-based beliefs with empirical evidence. As mentioned in the introduction, researchers in machine learning have developed a number of systems that combine data-driven or empirical learning with theory-driven or explanation-based learning. These models can be divided into two broad classes. Below, we describe these classes and discuss their plausibility, given the current findings.

5.1 Previous Models

Many models learn by noting similarities among the training examples and passing the output to a theory-driven component. Some of these systems, like OCCAM (Pazzani, 1987), use an empirical learning component to acquire domain theory knowledge, which can then be used by the explanation-based learning component to construct explanations. Basically, these methods improve the domain theories used by explanation-based learning systems. For example, Lebowitz (1986b) describes a system that employs UNIMEM to detect similarities across a number of training examples and then uses explanation-based learning to explain them. This approach apparently improves the efficiency of the explanation process.

In contrast, some models carry out explanation-based learning followed by empirical learning. These systems first proceed by processing training examples in a theory-driven manner and then sending output to a data-driven component (e.g., Cohen, 1988; Flann & Dietterich, 1989; Kedar-Cabelli, 1985; Mooney & Ourston, 1989; Pazzani, 1988; Shavlik & Towell, 1989; Yoo & Fisher, this volume). For example, Mooney and Ourston's (1989) IOU (induction over the unexplained) first applies explanation-based learning to training items. Features of the items that do not enter into the explanations are input to an empirical learning component, which detects features that tend to occur across the items. As previously mentioned, this approach is important because many concepts have both explanatory and nonexplanatory components, whereas the theoretical knowledge of explanation-based systems is typically explanatory in nature. Other systems use explanation-based learning to construct explanation trees for a number of training items and employ empirical methods to find the largest subtree that is common to all

the explanations (e.g., Flann & Dietterich, 1989; Yoo & Fisher, this volume).

In both types of models, the interaction between empirical and explanation-based learning is indirect and typically operates in one direction. In a number of these systems, the first module acts as a *filter* for the input to the second module by reducing the number of features that the second processes. For example, the explanation-based module in IOU first processes the training items and then passes a subset of their features (i.e., the unexplained features) to the empirical learning module. In Lebowitz's system, the empirical component processes training items and passes the common features to the explanation-based module. In these examples, one component influences the other but not vice versa.

However, our results suggest that the interaction between theory-driven and empirical learning can be much more direct. In this case, the learning modules are tightly coupled, with one module's processing "interwoven" with that of the other. Both modules influence each other directly. Now we turn to a model that incorporates this idea.

5.2 A Tightly Coupled Approach

We propose an alternative model, in which an explanation or rule is a function of three sources of knowledge: prior theories, explanations and data associated with previous training items, and the data provided by the current training item. In contrast, current models that integrate explanation-based and empirical learning formulate explanations that are constructed from *two* sources of knowledge: prior theories and the data provided by the current item. Below we speculate on the characteristics of this model and compare them to those current systems that combine explanation-based learning and empirical learning.

5.2.1 CONSTRUCTING DOMAIN RULES

In learning about categories of children's drawings, it is unlikely that people have a domain theory that can be applied directly to such drawings. For example, although many people might believe that creative drawings will show detail, what counts as evidence for detail might vary widely with the drawer. For a creative child, *shoelaces*, *buttons*, and *eyebrows* might provide good evidence. However, these features probably are not good indicators of detail in the drawings of creative adults.

Instead, people probably use general knowledge and functional specifications of categories to *construct hypothetical domain rules*, which they attempt to verify in the course of learning about the category. In the case of children's drawings, people probably access general knowledge structures corresponding to **creativity**, **children**, **drawing**, and what a typical person looks like. These structures are called *schemata* or *frames* (e.g., Minsky, 1975; Palmer, 1978; Norman & Rumelhart, 1975). Schemata capture (among other things) people's stereotypical beliefs and prior probabilities about events and objects, the visual appearance of objects, the temporal sequence of actions within an event, and so on.

Given the functional specification of a category (e.g., the meaningful label **drawings done by creative children**) and these basic knowledge structures, a person may plausibly construct an initial hypothesis about the category. In contrast to this rule-construction process, an explanation-based learning system typically has a domain theory containing preexisting rules. Using these rules, it directly links the features of a training item to the functional specification of a category. An exception to this requirement is Pazzani's (1987) **OCCAM**, which learns new domain theory rules from very high-level knowledge (e.g., about causality) much like our studies indicate that humans learn.

5.2.2 FEATURE INTERPRETATION

Our results also suggest that part of learning involves feature interpretation. This process can occur through a complex interaction of top-down knowledge (i.e., the hypothetical rules) and bottom-up knowledge (i.e., data from training items). There are several ways in which these types of knowledge might mutually influence each other during feature interpretation. First, theoretical expectations may initially activate feature schemata, and data from the current training item may constrain which schemata are plausible. These data include low-level shape descriptors and the relations among them, which are delivered by the perceptual system (Biederman, 1985; Marr & Nishihara, 1978).

Feature interpretation is a matter of finding a good match between feature schemata and the low-level data. As a simple example, consider again the part indicated by the arrow in drawing 4 of Figure 3. The hypothesis "creative children will show detail" will activate a wide range of schemata (e.g., **button**, **shoelace**, **pocket**, **eyebrows**, **teeth**). How-

ever, given this set of schemata, only the pocket schema is consistent with the shape and relative location of the indicated part. As a result, our participant recognizes the part as a pocket. Interestingly, the low-level constraints also are consistent with a purse schema. However, as we have suggested, this schema is unlikely to have been activated by top-down knowledge in this context, but it would be if a person thought that the drawing was done by a city child.

Second, the presence of features from *previous* training items may alter the strength of one's top-down knowledge. This change in strength may affect how one interprets the data of the *current* training item. For example, told that drawings were done by *eskimo children*, a person may have an initial, moderate expectation of seeing harpoons. After being shown a number of training items depicting eskimos holding harpoons, the person may increase his or her expectation that such drawings will contain harpoons. This increased expectation may, in turn, affect how the data of the current drawing are interpreted. Suppose the current drawing depicts a person holding a long *stick*, without the barbed, pointy end that is typical of harpoons. The person may interpret the *stick* as a harpoon, even though the low-level data are more consistent with a *stick* schema. If this had been the first drawing seen, then the person may have interpreted the harpoon as a stick, since initially he or she had only moderate expectations of seeing harpoons.

Given the possible interactions between theory and data, it may be best to view the presence of features as *probabilistic*. In contrast, most learning systems assume that all features are unambiguous and are present with certainty. In our model, all features can vary in their degree of certainty, from abstract features contained in hypothetical domain rules (such as *detail*) to the concrete features of training items (such as *buttons*). As people experience more training items and receive feedback, they may adjust these certainties. Sometimes the certainty of a feature becomes very low, it is effectively disregarded, and a new feature takes its place. Evidence for this claim comes from the finding that people sometimes reinterpreted data as new features that supported the correct category and abandoned previous interpretations that supported the incorrect category.

Of course, low-level perceptual constraints and top-down theory constraints need not interact in order for people to interpret features. In our first study, people in both the **THEORY** and **STANDARD** conditions

perceived many of the same (concrete) features in the drawings. Clearly, low-level perceptual information can activate features in the absence of prior expectations. Furthermore, one type of constraint may dominate the other. For example, no matter how strong your top-down expectations were for seeing elephants, you would still perceive drawing 1 of Figure 3 as a person.

5.2.3 CONTEXT-SENSITIVE EXPLANATIONS

In our model, people's explanations are sensitive to empirical evidence involved in the learning task. In particular, feedback and hypothesized features associated with previous examples influence the construction of later explanations. Whereas people construct context-sensitive explanations, systems that have used explanation-based learning followed by empirical learning have typically constructed context-independent explanations. That is, an explanation of an example E_i does not affect an explanation of the next example E_{i+1} .

We can think of several reasons why context sensitivity of explanations is a desirable quality, especially in the case where a learner's theoretical expectations or domain theory is weak. First, if an explanation of an example is incorrect, it would be sensible to avoid constructing a similar explanation for another item (i.e., "don't make the same mistake twice"). The incorrect explanation may indicate that the learner's theoretical expectations or domain theory needs modification. Likewise, if the same explanation is constructed for a number of examples, it might be sensible to prefer constructing that explanation for a new item. The successful explanation may indicate that aspects of the learner's theoretical expectations or domain theory are especially accurate.

Another important reason to prefer context-sensitive explanations is that some aspects of knowledge are *relative*. When applied to an item, the meanings of features such as *detailed*, *tall*, *heavy*, and *small* are defined with respect to other objects (e.g., Rips & Turnbull, 1980). For example, a small horse is small relative to other horses, but it is not small relative to many other objects (e.g., frogs and pencils). The fact that knowledge can be relative means that some explanations crucially interact with training data. For example, as noted in the second study, an explanation involving relative features may essentially be correct, but need to be "adjusted" based on the empirical evidence. Thus, one person believed that the drawings done by creative children would be

detailed, but after examining a drawing done by a noncreative child, the person adjusted the criteria for calling a drawing detailed. Here, the explanation remained basically the same, but was adjusted based on the empirical evidence.

One dramatic example of this interaction occurs with the use of the feature *detailed*. Consider three (hypothetical) sets of faces, ordered by increasing amount of detail. The first set (and least *detailed*) was drawn by noncreative children. The second set (and the second most *detailed*) was done by creative children. The third (and most *detailed*) was drawn by creative adults. The explanation "drawings will show *more detail*" accounts for why drawings are done by creative children when compared to noncreative children, but the same explanation will not account for why drawings are done by creative children when compared to creative adults. In fact, the opposite explanation that "drawings will show *less detail*" might be more appropriate.

6. Summary

In this chapter we presented two studies that examined the roles of theoretical expectations and empirical evidence in rule induction. These studies produced several major findings.

First, theoretical expectations strongly affect the kinds of rules that people construct for a given category. These rules are qualitatively different from those constructed by people without such expectations. Theoretical expectations appear to activate hypotheses about abstract features that might be true of a category. People who had these expectations tended to form rules with abstract features and to operationalize these abstract features by linking them to more concrete features in the training examples. In contrast, people without such expectations seldom used abstract features or operationalization in their rules. Instead, they focused on finding a set of simple, observable features that described a category. It also appears that theoretical expectations promote generalization across features. People operating with such expectations were able to treat different features equivalently. Therefore, they were able to find commonalities among category members that people without such expectations were unable to find.

Second, theoretical knowledge closely interacts with empirical evidence, and these types of knowledge mutually influence each other. In particular, the two sources of knowledge interact to determine hypo-

theoretical features upon which induction operates. They also mutually influence the types of explanations that people construct for a category. We briefly described how a learning system might closely integrate these sources of knowledge.

The view that people's theoretical expectations closely interact with experience generally has not been emphasized in machine learning models that integrate these two sources of knowledge. Furthermore, the notion that theoretical and empirical knowledge jointly determine the features for the induction process has been largely ignored in both cognitive psychology and machine learning. We hope that this paper contributes to a clearer understanding of these very complex problems.

Acknowledgements

We gratefully acknowledge the comments of Woo-kyoung Ahn, Douglas Fisher, Evan Heit, Pat Langley, Colleen Seifert, Edward Smith, and Rick Riolo on a previous version of this chapter. We also thank Bob Dylan for inspiration.

References

- Adelman, L. (1981). The influence of formal, substantive, and contextual task properties on the relative effectiveness of different forms of feedback in multiple-cue probability learning tasks. *Organizational Behavior and Human Performance*, 27, 423-442.
- Ahn, W., & Brewer, W. F. (1988). Similarity-based and explanation-based learning of explanatory and nonexplanatory information. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 50-57). Montreal, Quebec: Lawrence Erlbaum.
- Biederman, I. (1985). Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32, 29-73.
- Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking*. New York: John Wiley & Sons.
- Carey, S. (1985). *Conceptual change in childhood*. Cambridge, MA: MIT Press.
- Collins, A. (1978). *Human plausible reasoning* (Tech. Rep. No. 3810). Cambridge, MA: Bolt, Beranek, & Newmann.

- Collins, A., & Michalski, R. S. (1989). The logic of plausible reasoning: A core theory. *Cognitive Science*, 13, 1-50.
- DeJong, G. (1988). An introduction to explanation-based learning. In H. E. Shrobe (Ed.), *Exploring artificial intelligence*. San Mateo, CA: Morgan Kaufmann.
- DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1, 145-176.
- Dietterich, T. G., London, B., Clarkson, K., & Dromey, G. (1982). Learning and inductive inference. In P. R. Cohen & E. A. Feigenbaum (Eds.), *The handbook of artificial intelligence*. San Mateo, CA: Morgan Kaufmann.
- Ellman, T. (1989). Explanation-based learning: A survey of programs and perspectives. *Computing Surveys*, 21, 163-221.
- Flann, N. S., & Dietterich, T. G. (1989). A study of explanation-based methods for inductive learning. *Machine Learning*, 4, 187-226.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139-172.
- Fisher, D. H. (1988). A computational account of basic level and typicality effects. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 233-238). Saint Paul, MN: Morgan Kaufmann.
- Fisher, D. H., & Langley, P. (1985). Approaches to conceptual clustering. *Proceedings of the Ninth International Conference on Artificial Intelligence* (pp. 691-697). Los Angeles, CA: Morgan Kaufmann.
- Goodenough, F. L., & Harris, D. B. (1950). Studies in the psychology of children's drawings II, 1928-1949. *Psychological Bulletin*, 47, 363-433.
- Hanson, S. J., & Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3, 343-372.
- Harris, D. B. (1963). *Children's drawings as measures of intellectual maturity*. New York: Harcourt Brace & World.
- Haygood, R. C., & Bourne, L. E. (1965). Attribute and rule-learning aspects of conceptual behavior. *Psychological Review*, 72, 175-195.
- Hintzman, D. L. (1986). Schema abstraction in a multiple-trace memory model. *Psychological Review*, 93, 411-428.

- Kedar-Cabelli, S. (1985). Purpose-directed analogy. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 150–159). Irvine, CA: Lawrence Erlbaum.
- Keil, F. C. (1981). Constraints on knowledge and cognitive development. *Psychological Review*, 88, 197–227.
- Koppitz, E. M. (1984). *Psychological evaluation of human figure drawings by middle school pupils*. Orlando, FL: Grune and Stratton.
- Langley, P. (1987). Machine learning and concept formation. *Machine Learning*, 2, 99–102.
- Lebowitz, M. (1986a). Not the path to perdition: The utility of similarity-based learning. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 533–538). Philadelphia, PA: Morgan Kaufmann.
- Lebowitz, M. (1986b). Integrated learning: Controlling explanation. *Cognitive Science*, 10, 219–240.
- Marr, D., & Nishihara, H. K. (1978). Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London B* 200 (pp. 269–294).
- Medin, D. L. (1983). Structural principles of categorization. In B. Shepp & T. Tighe (Eds.), *Interaction: Perception, development, and cognition*. Hillsdale, NJ: Lawrence Erlbaum.
- Medin, D. L., & Ortony, A. (1989). Psychological essentialism. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. Cambridge: Cambridge University Press.
- Medin, D. L., & Shaffer, M. M. (1978). A context theory of classification learning. *Psychological Review*, 85, 207–238.
- Medin, D. L., & Schwanenflugel, P. L. (1981). Linear separability in classification learning. *Journal of Experimental Psychology: Human Learning and Memory*, 7, 355–368.
- Medin, D. L., Wattenmaker, W. D., & Michalski, R. S. (1987). Constraints and preferences in inductive learning: An experimental study of human and machine performance. *Cognitive Science*, 11, 299–339.
- Michalski, R. S. (1983a). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1). San Mateo, CA: Morgan Kaufmann.

- Michalski, R. S. (1983b). A theory and methodology of inductive learning. *Artificial Intelligence*, 20, 111-161.
- Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.
- Mitchell, T. M. (1980). *The need for biases in learning generalizations* (Tech. Rep. No. CBM-TR-117). New Brunswick, NJ: Rutgers University, Department of Computer Science.
- Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18, 203-226.
- Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1, 47-80.
- Mooney, R. J., & Ourston, D. (1989). Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 5-7). Ithaca, NY: Morgan Kaufmann.
- Mostow, J. (1983). Machine transformation of advice into a heuristic search procedure. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1). San Mateo, CA: Morgan Kaufmann.
- Muchinsky, P. M., & Dudycha, A. L. (1974). The influence of a suppressor variable and labeled stimuli on multiple cue probability learning. *Organizational Behavior and Human Performance*, 12, 429-444.
- Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review*, 92, 289-316.
- Norman, D. A., & Rumelhart, D. E. (1975). Memory and knowledge. In D. A. Norman & D. E. Rumelhart (Eds.), *Explorations in cognition*. San Francisco: Freeman.
- Nosofsky, R. M. (1986). Attention and learning processes in the identification and categorization of integral stimuli. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13, 87-108.
- Palmer, S. E. (1975). Visual perception and world knowledge: Notes on a model of sensory-cognitive interaction. In D. A. Norman & D. E. Rumelhart (Eds.), *Explorations in cognition*. San Francisco: Freeman.

- Pazzani, M. J. (1987). Explanation and generalization based memory. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 323-328). Irvine, CA: Lawrence Erlbaum.
- Pazzani, M. J. (1988). Integrated learning with incorrect and incomplete theories. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 291-297). Ann Arbor, MI: Morgan Kaufmann.
- Pazzani, M. J., & Schulenburg, D. (1989). The influence of prior theories on the ease of concept acquisition. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 812-819). Ann Arbor, MI: Lawrence Erlbaum.
- Pazzani, M. J., Dyer, M., & Flowers, M. (1987). Using prior theories to facilitate the learning of new causal theories. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 277-279). Milano, Italy: Morgan Kaufmann.
- Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1). San Mateo, CA: Morgan Kaufmann.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81-106.
- Rajamoney, S. A. (1986). *Automated design of experiments for refining theories* (Tech. Rep. No. UILU-ENG-86-2213). Urbana: University of Illinois, Department of Computer Science.
- Rendell, L. (1985). Substantial constructive induction using layered information compression: Tractable feature formation in search. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 650-658). Los Angeles: Morgan Kaufmann.
- Rips, L. (1989). Similarity, typicality, and categorization. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. Cambridge: Cambridge University Press.
- Rips, L. J., & Turnbull, W. (1980). How big is big? Relative and absolute properties in memory. *Cognition*, 8, 145-174.
- Scott, P. D. (1987). *On the systematics of machine learning*. Unpublished manuscript, Department of Computer Science, University of Michigan, Ann Arbor.

- Schank, R. C., Collins, G. C., & Hunter, L. E. (1986). Transcending inductive category formation in learning. *Behavioral and Brain Sciences*, 9, 639-686.
- Shavlik, J. W., & Towell, G. G. (1989). *Combining explanation-based and neural learning: An algorithm and empirical results* (Tech. Rep. No. 859). Madison: University of Wisconsin, Department of Computer Sciences.
- Smith, E., & Medin, D. L. (1981). *Categories and concepts*. Cambridge, MA: Harvard University Press.
- Utgoff, P. E. (1986). A shift in bias for inductive concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Wattenmaker, W. D., Dewey, G. I., Murphy, T. D., & Medin, D. L. (1986). Linear separability and concept learning: Context, relational properties, and concept naturalness. *Cognitive Psychology*, 18, 158-194.
- Wattenmaker, W. D., Nakamura, G. V., & Medin, D. L. (1987). Categorization processes and causal explanation. In D. Hilton (Ed.), *Contemporary science and natural explanation: Commonsense concepts of causality*. Sussex, England: Harvester Press.
- Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.
- Wisniewski, E. J., & Medin, D. L. (1991). *Is it a mornek or a plapet? Prior expectations about functionally-relevant features in category learning*. Unpublished manuscript, Department of Psychology, University of Michigan, Ann Arbor.
- Wisniewski, E. J., Winston, H. A., Smith, R. G., & Kleyn, M. (1987). A conceptual clustering program for rule generation. *International Journal of Man-Machine Studies*, 27, 295-313.
- Wright, J. C., & Murphy, G. L. (1984). The utility of theories in intuitive statistics: The robustness of theory-based judgments. *Journal of Experimental Psychology: General*, 113, 301-322.

CHAPTER 10

Concept Formation over Problem-Solving Experience

JUNGSOON YOO

DOUG FISHER

1. Introduction

Problem solving is often viewed as a classification task. This is nicely illustrated in chess, where both human experts (DeGroot, 1966) and some AI systems (e.g., Wilkins, 1980) categorize board patterns to find strategies that benefit the current situation. In this framework, learning identifies patterns that usefully discriminate between problem-solving choices. Production-system models of learning and problem solving (Anderson & Kline, 1979; Langley, 1985) capture this idea, as do more recent case-based approaches (e.g., Kolodner, 1987).

With this view in mind, it is natural that efforts to improve problem-solving efficiency look to traditional models of categorization. For example, concept formation systems like Fisher's (1987) COBWEB attempt to maximize the similarity of observations within the same categories and minimize the similarity of observations between categories. In principle, this strategy can efficiently guide problem solving by discovering classes of similar problems and solutions, so that categorization of new problems suggests plausible solutions. In fact, COBWEB has already been used to efficiently guide certain types of search-based reasoning (Carlson, Weinberg, & Fisher, 1990).

However, to make the full leap to problem solving, COBWEB and related systems must be extended in several ways. Notably, problem and solution characteristics are linked by interrelationships (Medin, 1990) that are not well expressed by independent features. In addition, these relations are often not immediately perceivable in a problem situation, but are still vital for purposes of classification. For example, a 'fork' in chess is an abstract relationship that one must infer from a particular board position before taking appropriate action.¹

The importance of inferred relationships for categorization has been formally studied by a number of researchers. For example, Chi, Felтовich, and Glaser (1981) had subjects construct categories of physics problems. Experts (graduate students in physics) defined categories based on principles that played an important role in the problems' solutions. For example, problems with solutions that depend on Newton's second law were categorized together. In contrast, novices grouped problems that shared features that were directly perceived in the problem statement (e.g., two problems involving an inclined plane).

These and other data (Medin, 1990; Ross, 1987; Ross & Spalding, this volume; Wisniewski & Medin, this volume) have caused many to question surface similarity as a sole basis for categorization. A growing view is that human categorization involves a rich set of underlying relationships or 'deep' features. This view also dominates explanation-based (Mooney, this volume) and case-based (Kolodner, 1987) research, which are intimately concerned with explicating feature interrelationships, and identifying deep and surface features that guide problem solving.

This chapter describes a model of learning and problem solving that builds on approaches to clustering and object concept formation, but that also incorporates lessons from psychological, explanation-based, and case-based research. In particular, the EXOR system clusters problems and their solutions into a classification hierarchy; this is the source of its name, which is an acronym for 'EXplanation ORganizer'. Problem solving initially requires an unconstrained search of a 'domain theory' of relations. With experience, however, problem solving becomes a matter of categorizing problem descriptions based on surface and deep features, and reusing solutions in the process.

1. A 'fork' occurs when one player's pieces threaten another player's pieces in two or more ways. The defensive player can only guard against one of these threats, leaving her or him vulnerable from the other direction.

The remainder of the chapter presents this system and the ideas on which it is based. Section 2 opens by reviewing general principles that are common to inductive concept formation and explanation-based learning. Section 3 describes how these principles are partially implemented in the EXOR system, which is evaluated in Section 4. Finally, Section 5 extends the basic strategy in several ways, particularly by exploiting deep features during categorization.

2. Principles of Memory Organization

We have noted that expert problem solving can be viewed as an efficient process of classification. Machine learning researchers have modeled the acquisition of classification rules for problem solving in a variety of ways (Langley, 1985; Mitchell, Utgoff, & Banerji, 1983). The most recent of these is explanation-based learning. One particular model in this paradigm is *explanation-based generalization* (Mitchell, Keller, & Kedar-Cabelli, 1986) or EBG, which assumes that an agent's theory of a domain is a set of primitive implication (IF-THEN) rules. Initially, problems are solved by searching the domain theory for chains of rules that link the problem's statement to a solution. Problem solutions are then generalized so that they can be efficiently reused.

This process can be illustrated in a domain of *algebra story problems* (Mayer, 1981). Consider the following problem:

A train leaves a station and travels east at 72 km/h. Three hours later a second train leaves and travels east at 120 km/h. How long will it take to overtake the first train?

Figure 1 presents a solution to this 'overtake' problem using a set of primitive rules.² To summarize, the time until overtake (Time = Distance/Rate) is obtained from the distance that must be made up by the faster train ($D = D_1 + \Delta d$), and the relative rate of travel of the second train ($R = R_2 - R_1$). The solution trace forms an AND tree in which the root is the variable whose value we want to find, and the leaves are conditions that are known to be true from the problem's

2. For brevity, we have abbreviated several domain theory rules. For example, $T = D/R$ is short for $\text{Time}(X) \Leftarrow \text{Dist}(Y) \wedge \text{Rate}(Z) \wedge (X \leftarrow Y/Z)$. Thus, the depths of the solution traces are between six and seven inference rules deep, rather than three or four, as suggested by the figure. In addition, the figure omits the rule that two vehicles going **east** implies they are moving in the **same direction**.

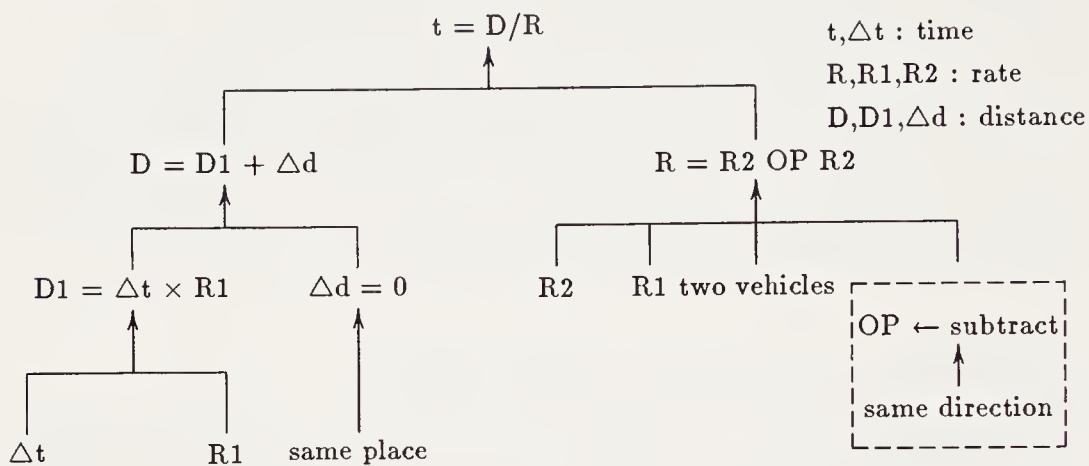


Figure 1. A generalized problem solution trace (explanation).

statement. In the context of our introductory discussion, conditions in the problem statement correspond to surface features that are networked together via a search of the learner's background knowledge or theory about algebraic relationships and domain-specific predicates like *distance*, *time*, and *rate*. Since rules of the domain theory can be combined in a wide variety of ways, finding the configuration that correctly solves a problem requires search.

One goal of explanation-based learning is to improve the efficiency of problem solving. For example, EBG will generalize the arguments (i.e., by turning them to variables in a controlled way) of the 'overtake' solution so that it can be reused on future problems. The generalized solution trace that results can then be used for an 'overtake' problem that involves cars instead of trains that are traveling at 55 mph and 75 mph. Presumably, this generalization process mitigates the need for extensive search when new 'overtake' problems are encountered. However, recent research (e.g., Minton, 1990) indicates that learned knowledge differs in *utility*; naive storage and reuse strategies can have a detrimental effect on efficiency. In these cases, the search for applicable explanations may become more expensive than the search of the original domain theory.

In part, inefficiencies stem from redundancies introduced by learned knowledge (Markovitch & Scott, 1989; Minton, 1990). We can see this by considering an 'opposite-direction' algebra problem:

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

This problem has a solution structure that is almost identical in form to the solution of Figure 1; it differs only in the structure of the boxed subtree, which calculates the relative rates of two trains moving in the same direction instead of opposite directions. Unfortunately, even after variabilization, the applicability of the original ‘overtake’ solution trace will be highly limited; it cannot contribute to the solution of the ‘opposite-direction’ problem. Nonetheless, an EBG-like system may expend considerable effort on matching the new problem’s constraints against the old solution before a contradiction (e.g., *opposite-direction* versus *same-direction*) is finally found. Those portions of the old solution that are shared with the new problem will have to be ‘rediscovered’ in a domain theory search. If there are many solutions that differ in small ways from the new problem, then efficiency will degrade because redundant matching of common substructure must occur before previous solutions can be rejected.

Recently, Flann and Dietterich (1989) identified a class of inductive strategies (Hirsh, 1989; Pazzani, 1990) that abstract redundancies out of multiple explanations. These systems excise substructures that are not shared among explanations. Flann and Dietterich illustrate their particular approach to ‘induction over explanations’ (IOE) with the task of explaining how the *operational* (i.e., surface, perceivable) properties of an observation satisfy the functional requirements of a cup. The left-most portion of Figure 2 illustrates two (abbreviated) explanations of cup membership in terms of operational predicates. These explanations differ in the ways that they enable grasping (Gr): by virtue of having a handle (Hh) or having suitable insulation (In). These surface or operational predicates are networked via domain theory rules which indicate that a cup holds liquid (Hl), is liftable (Li), and is stable (St); it is liftable if it is lightweight (Lw) and graspable (Gr).

IOE superimposes these explanations, and severs subtrees that differ at their roots among the explanations.³ For example, the boxed abstrac-

3. Actually, IOE also instantiates predicate arguments (i.e., with variables and constants) so that the generalized explanation is maximally specific, but still consistent with all the explanations. We use the complete IOE procedure, but its full description here is somewhat tangential to our purposes.

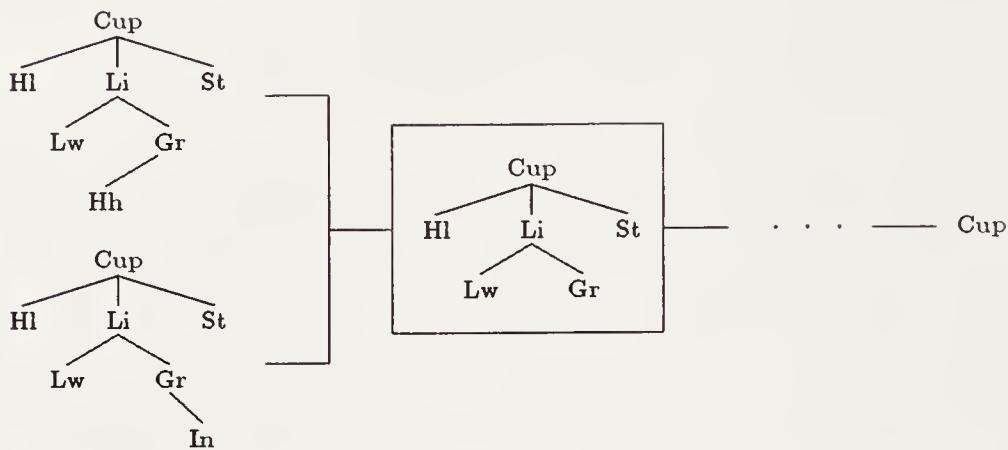


Figure 2. Maximally specific to maximally general abstractions.

tion gives the result of applying this process. Notice that the abstracted explanation excludes a commitment as to how the *graspable* predicate might be operationalized. The abstraction can still be reused to determine cup membership of new objects, but the graspability of the object must be verified by a (presumably trivial) search of the original domain theory rules.

Induction over explanations can abstract redundant substructures out of different explanations. Flann and Dietterich were concerned with extending the flexibility of representing target concepts (e.g., *cup*), but the approach also promises to improve the efficiency of problem solving. For example, in the domain of algebra story problems, the generalized solution structure with the boxed subtree removed (see Figure 1) would provide a large step toward the solution of the ‘opposite-direction’ problem, with only the relative rate left for domain-theory search to compute.

However, IOE is simply a generalization operator; it does not identify when induction will be beneficial and when it will be detrimental. For example, consider that there may be radically different explanations of why oat-bran cereal and baked fish are *health-food*. Abstracting out the common substructure might yield an ‘explanatory’ substructure, *health-food(x)*, which is a trivial statement of the target concept; this is one lesson that Fisher and Pazzani (Chapter 6, this volume) draw from experiments with Pazzani’s (1990) OCCAM. Figure 2 illustrates that maximally specific explanations provide a complete explanation for new situations, but that it is difficult to find applicable explanations

since redundant structures must be searched multiple times. Conversely, maximally general structures (e.g., $\text{cup}(x)$, $\text{health-food}(x)$, $\text{time}(?)$) make it easy to find applicable ‘experience’, but nothing useful is gleaned as a result.

The tradeoffs of explanation-based memory are closely related to tradeoffs traditionally found in inductive concept formation (Fisher, 1987; Lebowitz, 1982; Kolodner, 1983; Medin, 1983). In concept formation, one desires contrast categories that are easily distinguished and that maximize inter-category dissimilarity. If members of different categories are as distinct as possible, then it will be easy to categorize an observation. However, a simple-minded attempt to maximize inter-category dissimilarity results in the construction of one, all-inclusive category, since there are no contrast categories at all. Rather, one wants to balance inter-category dissimilarity against intra-category similarity; if observations within the same categories are very similar, this lets one make accurate predictions about the properties of category members. An attempt to unilaterally maximize intra-category similarity favors single-member categories, since each observation is maximally similar to itself. Unfortunately, this complicates an initial categorization, since observations will tend to overlap with one another. Thus, a suitable tradeoff between inter-category dissimilarity and intra-category similarity tends to optimize the potential of a categorization scheme for purposes of prediction.

Our implicit equation of inductive concerns about ‘prediction’ accuracy (Quinlan, 1986; Fisher, 1987) with traditional explanation-based concerns about *efficiency* may strike some readers as unintuitive. However, note that choices in a domain theory search constitute predictions; informed and accurate decisions at choice points result in an efficient search, while erroneous decisions cause backtracking and inefficiency. The hypothetical curve of Figure 3 illustrates performance trends that might occur under different conditions. Maximally general descriptions will underfit the data, necessitating uninformed prediction (i.e., uninformed search through a domain theory). Conversely, maximally specific explanations overfit the data and introduce redundancies into the search for applicable experience.⁴ An ideal abstraction (boxed in Figure 2) should be relatively unique and easily identified; it can then pre-

4. Figure 3 implies that the cost of overfitting asymptotes at the level of uninformed prediction. In fact, overfitting may result in performance that is worse than the uninformed case (Fisher, 1987; Mooney, 1989).

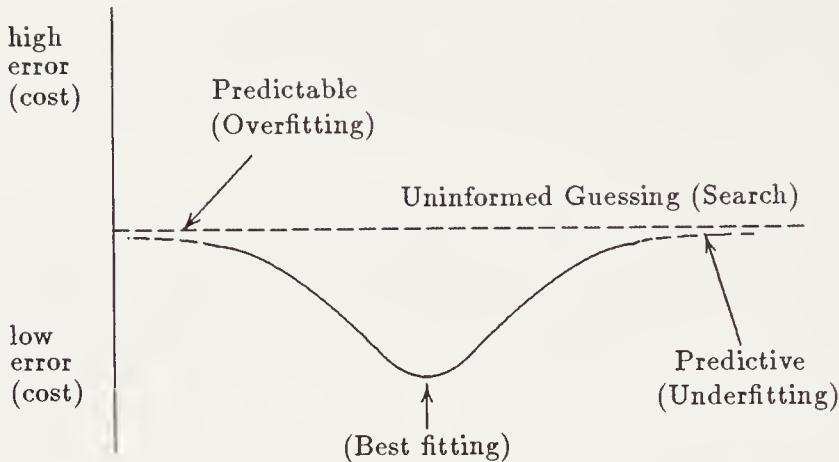


Figure 3. Prediction error (and search efficiency) as a function of abstraction.

dict a sizable portion of the complete explanation structure; the domain theory must only complete the proof (e.g., of *graspability*).

In summary, induction over explanations promises to reduce redundant search, but this process must be regulated to be useful. We suggest that appropriate regulatory mechanisms are provided by well-studied principles of inductive learning. The following section describes our approach to this task. In particular, we use a simple concept formation strategy to abstract and exploit problem-solving experience.

3. Concept Formation over Explanations

Our system, EXOR, carries out concept formation over explanations. In particular, Flann and Dietterich's IOE is embedded within a control structure for building abstraction hierarchies inspired by Lebowitz's (1982) UNIMEM and Fisher's (1987) COBWEB. Figure 4 gives an example of an abstraction hierarchy over algebra story problems (e.g., overtaking, opposite direction, round trip) taken from Mayer (1981). Solutions to these problems range from very similar to quite different. The domain theory includes formulae that represent various ways for solving for quantities like Distance, Time, and Rate. Within each node of the abstraction hierarchy is a generalized explanation subtree that is common to *all* descendants of the node. If there is no common substructure over the entire set of observed explanations, then the root of the hierarchy will be empty.

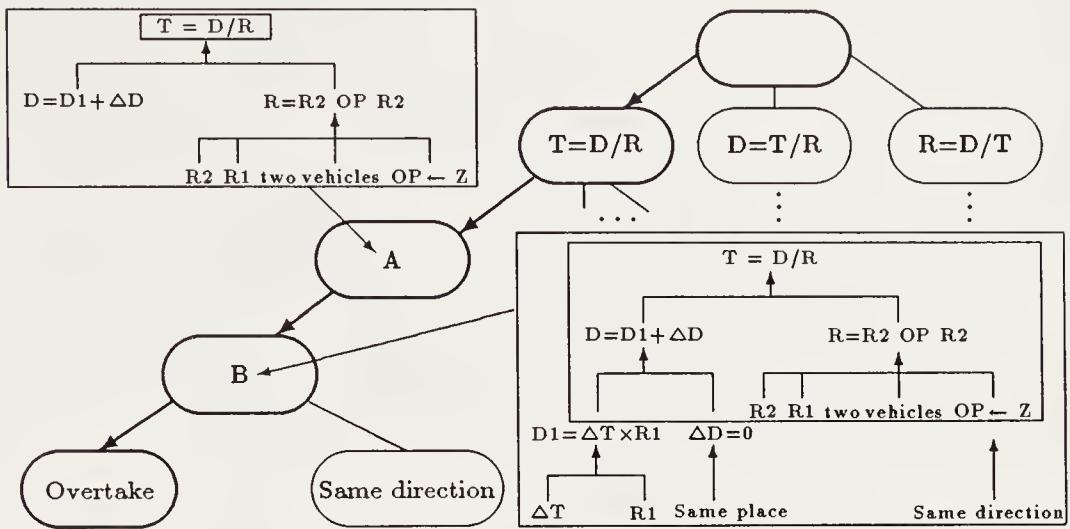


Figure 4. A classification hierarchy over explanations (problem-solving traces).

EXOR incrementally incorporates a new problem's solution into an existing hierarchy. To do this, the explanation is compared to the explanation substructure of a node in the abstraction hierarchy (initially the root), and the IOE procedure generalizes the new explanation and the node's substructure. If this results in a generalization that is equivalent to the node's generalized explanation, then the new explanation must be more specific than the node's partial explanation; in this case classification of the explanation proceeds to the children of the node. If IOE yields a structure that is more general than the current node, then the node's explanation structure is not more general than the new explanation; in this case, the new explanation is made the node's sibling.

Of course, to be useful the hierarchy must not simply be used to store explanations, but it should facilitate explanation construction (i.e., problem solving). In particular, EXOR can be given an existing classification tree, a target variable (e.g., distance), and a problem statement of known predicates. The general strategy is to classify the problem conditions and target variable down the hierarchy, 'accumulating' the solution for the new problem as classification proceeds. In particular, this process begins at the root of the classification tree. If a node's partial solution does not contradict known conditions of the problem statement, then the solution structure stored there is asserted as a partial solution for the new problem. After this, EXOR investigates the node's children in an attempt to complete the solution.

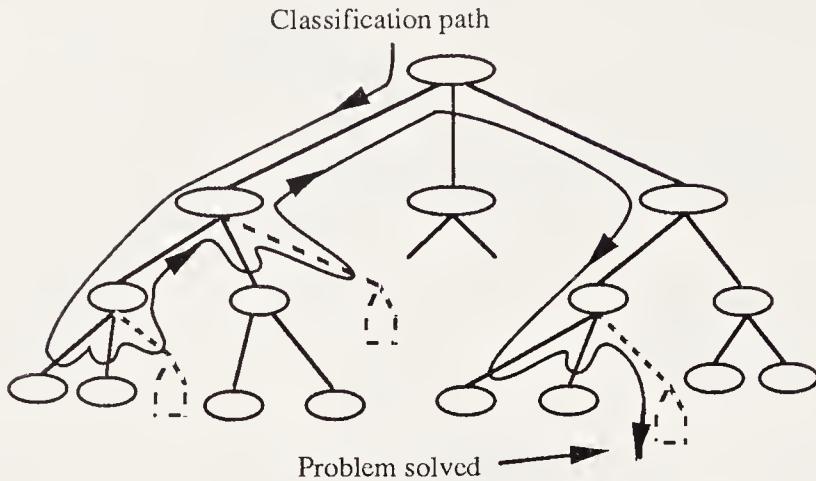


Figure 5. Paths taken during problem-solving classification.

In contrast, if a part of a node's partial explanation contradicts a known condition of the problem (e.g., traveling east instead of west), then the node is abandoned from further consideration and search control looks to a sibling of the node. That is, control returns to the node's parent and another child is explored. If all of a node's children result in contradictions, then an attempt is made to complete the partial explanation accumulated thus far with the *domain theory*. If this fails, then the node is abandoned as above, the partial solution associated with that node is retracted from the new problem's solution, and backtracking returns control to the node's parent. Figure 5 shows a high-level trace of the classification process, including backtracking, that occurs when solving a new problem. 'House'-shaped 'nodes' represent searches of the domain theory that occur after all of a node's children are examined unsuccessfully, but before the node is abandoned.

To understand this process more deeply, consider that explanation-based classification requires an AND/OR search of a domain theory. For example, a domain theory rule which states that *distance = rate × time* requires the system to solve for *rate* and *time*. In turn, there may be several ways to solve each of these quantities; this choice constitutes the OR part of the search. EXOR still conducts an AND/OR search, but transforms it. Collections of ANDed rules are stored as generalized explanations *within* nodes of the abstraction hierarchy, while OR conditions are implicit in the choice *between* nodes (siblings) that

can be made during classification. If the space of alternative explanations defined by the classification hierarchy is sparser than the space defined by the original domain theory (i.e., if only certain rules occur in conjunction), then efficiency will likely be improved. However, even if the ‘explanation space’ cannot be strictly reduced, the average search required for arbitrary problems can be reduced by exploiting differences in explanation utility (Minton, 1988). Trivially, explanation substructures (e.g., classification tree nodes) can be ranked by the proportion of past cases in which they were applicable. The inductive assumption is that this proportion reflects the probability that they will be applicable in the future and that examining nodes in order of likelihood will reduce the space of explanations one must search on average, just as subgoal ordering can reduce search in logic programming (Warren, 1981).

However, it is possible to use a more sophisticated approach to guide search. In particular, the system uses a measure of *category utility* (Gluck & Corter, 1985) to rank the promise of children under a node. In addition to storing a partial explanation at a node that is true of all of the node’s descendants, EXOR also stores statistics on the distribution of surface predicates that are found in problem statements. These predicates need not occur in all descendants; rather, their probability of occurrence in descendants is stored at the node. Statistical trends in these predicates can be used to heuristically guide the selection of nodes from which EXOR builds an explanation for the current problem.⁵

Intuitively, category utility measures the predictability and predictiveness of a new problem’s known predicates relative to a category — i.e., a classification tree node. The predictability of a predicate F_k relative to a category (node) N_i is given as $P(F_k|N_i)$, the probability that F_k will participate in an explanation stored under N_i . The predictiveness of a predicate is given by $P(N_i|F_k)$, the probability that an explanation with F_k will be stored under N_i . The main term in category utility, which specifies a tradeoff between these two factors, is $\sum_k P(F_k)P(N_i|F_k)P(F_k|N_i)$, where $P(F_k)$ weights the importance of the tradeoff for the most frequently observed predicates. Fisher and Pazzani (Chapter 1, this volume) give the complete definition for this function.

5. Probabilistic matching overcomes some of the problems associated with OCCAM (see Fisher & Pazzani, Chapter 6, this volume), which stem from its use of an all-or-none matching scheme to classify ‘events’.

When a problem statement is presented to EXOR, candidate nodes are ranked by their category utility scores over the predicates of the problem statement. The highest scoring nodes are investigated first. The system follows this process recursively as it descends the hierarchy. Contradictions may still arise, causing EXOR to abandon a proposed node, but the inductive assumption is that the distribution of surface predicates provides some heuristic guidance.

4. Experimental Results

We tested EXOR's ability to improve problem-solving efficiency in the algebra domain, using 48 problems. We selected a subset of 32 problems for training, and retained 16 problems (one of each problem type) for testing. At intermittent points in training (every four problems), we evaluated the performance of the EXOR classification tree. In particular, we compared the total number of predicates instantiated using a domain theory search (no learning), an EBG-like system, and the heuristically guided EXOR tree search over the test problems.⁶

Figure 6 illustrates the total number of predicates instantiated during a domain theory search and an EXOR classification tree search as training proceeds over the 32 training problems. The experiment was averaged over ten random orderings of the tasks. The solid horizontal line reflects the total work performed by domain theory search alone (without learning) over the 16 test problems. The figure also graphs the amount of work performed using EXOR trees, but recall from the description of the explanation-construction procedure that search using an EXOR classification tree stems from two sources. First, EXOR searches paths in the classification tree to find a maximally specific node that seems applicable to the new problem. The partial explanation at such a node does not contradict the operational (observed) predicates of the new problem, but the partial explanation may not be successfully extended by any of the node's children. However, before a node is abandoned, the system makes a final effort to extend the partial explanation using the domain theory; these (nested) domain theory searches are the second source of search. The shaded (lower, increasing)

6. There are many dimensions over which one can compare performance, including the total number of rules examined and the total number of backtracks. We report predicate instantiations, as this is the most granular dimension. Segre, Elkan, and Russell (1991) provide a good summary of possible performance dimensions.

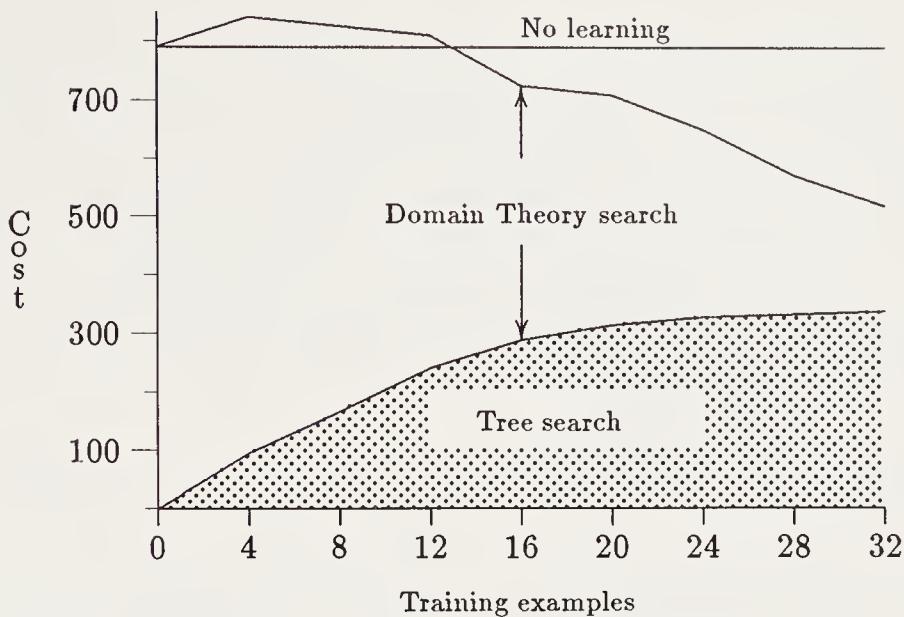


Figure 6. The search efficiency of EXOR as a function of training in the algebra domain.

area reflects search in tree nodes; the upper (decreasing) curve gives the total amount of search required to solve all 16 test problems. The amount of domain theory search to complete partial solutions is the difference between the upper and lower curves. Thus, EXOR reduces the overall effort required to solve problems, as shown by the decreasing curve; the required effort is increasingly borne by the EXOR classification tree, while the domain theory plays a correspondingly smaller part as training proceeds.

We have not graphed the efficiency of EBG, but we did test it in the same domain. After 32 training problems, EBG required 1352.6 predicate instantiations to solve 16 test problems. This is considerably more than either EXOR or the domain theory alone. EXOR's relative success stems from its ability to exploit shared partial solutions. For example, the system can exploit the generalized solution from an 'opposite direction' and 'overtake' problem to partially solve a 'closure' problem, something that EBG cannot do. This limitation of EBG is magnified when there are many explanations that differ in very minor ways. For example, suppose that an opposite-direction problem describes a car traveling east and one traveling west, and domain theory inference rules tell us that east and west are opposite-directions,

as are north and south. EBG cannot exploit the solution to the first problem in its attempt to solve a new problem, which is identical to the first, except that the cars are traveling north and south. In fact, the great similarity between problems may lead to a considerable amount of redundant search until a contradiction is found. This was the case with many of the problems in our domain, which explains the poor performance of EBG.⁷ In the next section we discuss this and other issues of explanation-based learning, and the manner in which EXOR addresses these limitations by drawing on lessons adapted from inductive learning.

5. Issues in Human and Explanation-Based Learning

EXOR casts problem solving as a classification process that descends from earlier approaches to hierarchical concept formation. Beyond this general framework, the system's classification hierarchies also suggest natural approaches to more specific issues of explanation-based and human learning. We summarize some of these issues here and describe extensions to EXOR that are motivated by them.

5.1 The Utility Problem Revisited

The utility problem (Minton, 1990) suggests that learned knowledge differs in its contribution to problem-solving efficiency. In fact, some acquired rules have a detrimental effect on performance. Markovitch and Scott (1989) identify three general strategies for mitigating the utility problem. *Selective acquisition* methods forecast the probable utility of a rule in advance of its actual use, and retain it only if it is likely to be helpful (Iba, 1989; Minton, 1990). In contrast, *selective retention* evaluates rule utility in actual practice, and only retains rules that prove helpful (Iba, 1989; Minton, 1990).⁸ However, the most general strategy

7. Of course, other representations of the domain theory would yield different, possibly better, results. The point is that IOE (Flann & Dietterich, 1989) and thus EXOR are less sensitive to the precise representation of the domain theory.

8. Of these two alternatives, several authors seem to prefer selective retention to selective acquisition, since the objective utility ratings supplied by actual experience are more likely to be reliable than subjective predictions of utility (Etzioni, 1988; Fisher & Chan, 1990; Scott & Markovitch, this volume). This is reminiscent of earlier debates on the use of objective and subjective probabilities in statistics and AI. However, the merits of objective assessments are accentuated in explanation-based settings where *a priori* expectations may be minimal.

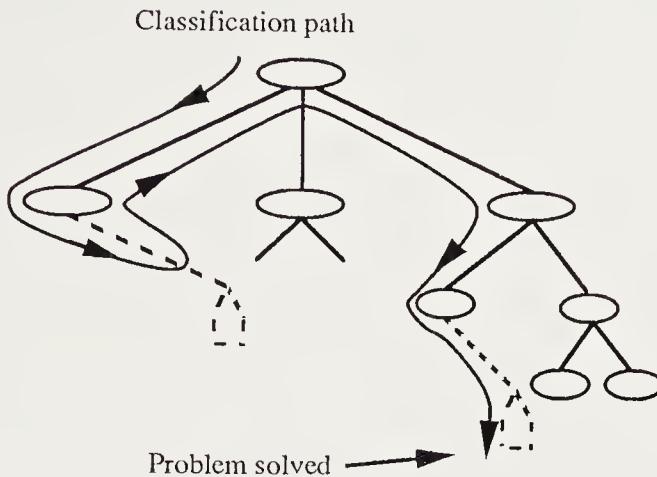


Figure 7. A classification tree after pruning.

is one of *selective utilization*, which assumes that learned rules are not necessarily 'good' or 'bad,' but that their merit varies with context.

The problem in selective utilization is to determine when learned knowledge should be exploited and when it should not. For example, Markovitch and Scott's (1989) LASSY constructs AND tree 'explanations' or proofs from a search of IF-THEN or implication rules. The system accumulates statistics on how often an antecedent (or subgoal) in the left-hand side of an implication fails to be achieved; learned rules are only used in attempts to prove antecedents that have succeeded sufficiently often (e.g., at least 50% of the time). The justification for this strategy is that, if a subgoal is likely to fail, then one should not search for a subproof in vain *twice* — once with learned rules and once with the domain theory from which the learned rules were constructed. Mooney's (1989) EGGS uses a technique that is similarly motivated. These approaches mitigate the utility problem and improve efficiency, but nonetheless suffer from two limitations. First, the likelihood of subgoal failures is only estimated within the context of a single (domain theory) rule; intuitively, one might expect that the likelihood of a subgoal failure would be dependent on a more complete problem-solving context. Second, LASSY and EGGS decide to make all learned rules available for examination or none. In contrast, we believe the relevance of learned rules will vary with problem-solving context, and one should ignore rules that are not relevant in a particular context.

An EXOR classification hierarchy addresses both limitations. Each node represents the problem-solving context as a partially expanded explanation. We have extended EXOR beyond the basic strategy described in Section 3 to maintain statistics like LASSY's on the number of backtracks (failures) at each node. Nodes with an unacceptable number of backtracks (e.g., greater than 50% of the time) are *pruned* (Quinlan, 1986). Figure 7 illustrates how pruning might alter the classification tree of Figure 5. If all of a node's children are pruned, this effectively identifies the problem-solving contexts in which the system should rely exclusively on the domain theory. Those children that remain serve to identify learned extensions that have been previously applicable; learned rules that are not present in these children are not considered when attempting to extend the explanation from the current node.

In sum, EXOR's approach offers several advantages for problem solving. These stem in part from the hierarchical partitioning of problem types and solutions. Each node represents a problem-solving context that is refined as one descends the hierarchy; the tree structure supports context-dependent decisions about the relevance and utility of learned knowledge. In addition, advantages stem from EXOR's inductive concept formation strategy. As we noted in Section 2, specialized rules may be idiosyncratic and degrade performance. In problem-solving domains and others, these specializations are best dealt with through well-studied inductive mechanisms such as pruning. On this particular dimension, some researchers (Carlson, Weinberg, & Fisher, 1990; Fisher & Chan, 1990; Holder, 1990) have argued that inductive issues of noise and search-based issues of utility are closely linked and are best dealt with in a uniform manner.

5.2 Boundaries of Operability

In addition to pruning and selective utilization, we have also investigated a second extension to EXOR. Thus far, we have described a search procedure guided solely by category utility computed over surface predicates of the problem statement. These correspond to early notions of operational predicates in explanation-based learning, which were defined as predicates that are easily observable or testable (Mitchell, Keller, & Kedar-Cabelli, 1986). However, these are not necessarily optimal for purposes of efficiency. For example, in algebra story problems, predicates such as east and west convey little information per se; it is the in-

furred predicate, *opposite-direction*, that identifies almost uniquely the solution type to which a problem corresponds. Thus, we hope to improve performance by inferring appropriate predicates from the problem statement during classification.

The success of this strategy depends on identifying predicates like *opposite-direction* that differentiate categories of problem-solving experiences, thus better focusing the search for solutions to new problems. However, proving or disproving the truth of a predicate requires effort as well. Thus, an ideal ‘boundary of operationality’ (Braverman & Russell, 1988) includes predicates with greater efficiency benefits than costs. We can formalize this notion in terms of the *expected number* of problem-solving steps (or predicates instantiated, or any of several other measures of *cost*) required to solve a problem with and without knowledge of a predicate’s truth. Let $E(\text{cost}|N)$ be the expected cost of solving an arbitrary problem beginning at node N of an EXOR classification tree. Assume that we investigate the children, C_i^N , of N in order of probability. Our inductive assumption is that children will successfully extend the current problem with roughly the same probability that they successfully classified earlier problems. Thus,

$$\begin{aligned} E(\text{cost}|N) = \\ P(C_{\max}^N)[E(\text{cost}|C_{\max}^N)] + \\ P(C_{\max-1}^N)[E(\text{cost}|C_{\max-1}^N) + U(\text{cost}|C_{\max}^N)] + \\ P(C_{\max-2}^N)[E(\text{cost}|C_{\max-2}^N) + U(\text{cost}|C_{\max}^N) + U(\text{cost}|C_{\max-1}^N)] + \\ \dots \\ P(C_{\max-m}^N)[E(\text{cost}|C_{\max-m}^N) + U(\text{cost}|C_{\max}^N) \\ + \dots + U(\text{cost}|C_{\max-m+1}^N)] , \end{aligned}$$

where $P(C_{\max}^N) \geq P(C_{\max-1}^N) \geq \dots \geq P(C_{\max-m}^N)$, $E(\text{cost}|C_j^N)$ is the expected cost (e.g., number of steps) of successfully finding the problem’s solution under a child, C_j^N , and $U(\text{cost}|C_j^N)$ is the expected cost of an unsuccessful search of C_j^N for a solution to the problem. Thus, the cost of finding a solution in the second most probable subtree of N , $C_{\max-1}^N$, includes the cost of having first searched the most probable node unsuccessfully. These quantities can be computed or at least approximated from the statistics that are maintained in the tree, such as $P(C_j^N)$, and from the structure of the tree itself, such as $E(\text{cost}|C_j^N)$.

Using similar constructions, we can approximate the expected cost when the truth of a predicate F_k is known, $E(\text{cost}|N, F_k)$, and the expectation in the case of $\neg F_k$, $E(\text{cost}|N, \neg F_k)$.

Putting these quantities together, an attempt to verify a predicate's truth is made if

$$\begin{aligned} E(\text{cost}|N) > \\ P(F_k|N)[E(\text{cost}|N, F_k) + E(\text{cost to prove } F_k)] + \\ [1 - P(F_k|N)][E(\text{cost}|N, \neg F_k) + E(\text{cost to prove } \neg F_k)] \quad , \end{aligned}$$

where $P(F_k|N)$ is the probability that one will be able to prove the truth of F_k (e.g., **opposite-direction**) and $1 - P(F_k|N)$ is the probability that the proof will fail. In general, knowing the complement (e.g., **same-direction**) to be true can also be predictive of a particular course of action. $P(F_k|N)$ can be approximated by the proportion of explanations stored under N that contain F_k . EXOR currently approximates $E(\text{cost to prove } F_k)$ and $E(\text{cost to prove } \neg F_k)$ from the domain theory, based on the number of rules that contain F_k as a consequent and the number of rule combinations that can conclude F_k . Assuming the combinations that are responsible for concluding F_k are equiprobable, the expected cost is proportional to approximately $\frac{1}{2}$ the cardinality of this set. Intuitively, the greater the branching factor and distance of F_k from the initial operational (problem statement) boundary, the greater the cost of inferring it.⁹

EXOR exploits predicates that are good candidates for inference during classification.¹⁰ Notice that the utility of inferring a particular predicate F_k varies with the problem-solving context specified by a node N ; EXOR only expends work to infer F_k when it helps distinguish N 's descendants. For example, within a certain context one might distinguish **opposite-direction** from **same-direction** problems, since this helps

-
- 9. Notice that $E(\text{cost to prove } F_k)$ is not conditioned on N , though this would be preferable. Explanations under N may exhibit a relatively small number of combinations to prove F_k ; thus, we should approximate $E(\text{cost to prove } F_k|N)$ in future work.
 - 10. Currently, we identify these predicates manually, using an approximation of the inequality above. However, we are in the process of automating this procedure. Once identified, the current system automatically maintains statistics and inferences about predicates during classification.

Table 1. Comparison of different versions of EXOR in terms of search.

	BASIC EXOR	PRUNE/INFER	IMPROVEMENT
CONCEPT TREE	337.0	316.4	6.1%
DOMAIN THEORY	180.9	140.6	22.3%
TOTAL	517.9	457.0	11.8%

define the relative rate between vehicles. However, in subcontexts (i.e., subtrees) of the hierarchy these predicates may have limited value. Instead, it may be useful to infer whether a *current* is present from problem-statement references to boats on a river, planes in the sky, or cars on a highway, since this affects the magnitude of relative rates.¹¹ More generally, EXOR's tree-structured classification scheme naturally supports context-dependent boundaries of operability.

Once identified, statistics on inferred predicates are used in the category utility calculation to bias search in the most promising directions. Table 1 shows the result of predicate inference and pruning on EXOR's performance, using the same dependent measure and the same ten orderings as in Figure 1, after the 32 training examples. These extensions reduce the search within the EXOR tree by 6.1% and lower the domain theory search by 22.3%. The total savings from inference and pruning is 11.8%.

Our approach is related to research by Braverman and Russell (1988) and by Keller (1988), who have directly linked operability to performance. For example, Braverman and Russell argue that to improve the efficiency of a problem solver, a rule should be as general as possible (in the IOE sense) to increase its applicability, while still retaining a set of easily verified predicates as part of the definition. Thus, these authors link operability to utility. We have continued along this path, but have made the tradeoff between the costs and benefits of inference more explicit. A suitable boundary of operability is a set of predicates whose inference is *cost effective*.

11. Although Mayer (1981) identifies 'current' problems as one problem class, we have not included these in our study. Rather, we use it here as an intuitive example of context-dependent operability.

5.3 Deep and Surface Features

Section 1 highlighted characteristics of human problem solving and classification that parallel aspects of EXOR's processing. In particular, there is growing evidence that 'surface' similarity alone is an inadequate basis for effective problem solving. Rather, 'deep', solution-related features are often exploited by problem solvers (Faries & Reiser, 1988). This is not to say that surface features are unimportant. In particular, Chi et al. (1981) found that novices sort problems based on surface similarities, since they have little else on which to base decisions. In fact, this is not a bad heuristic strategy, since surface similarity and 'deep' similarity are often correlated (e.g., the mention of boats usually signifies a current problem); under these conditions experts use surface similarity as well (Ross & Spalding, this volume).

We believe that the methods of Section 5.2, and explanation-based research on boundaries of operability generally, speak to these psychological concerns. In particular, EXOR does not distinguish between deep and surface features *per se*. Rather, cost effective features are exploited as a classification scheme evolves. Very early in this evolutionary process, surface features are regarded as cost effective because they cost nothing to infer, and they are likely to discriminate the few data points that are available. Features that continue to discriminate as training increases are still used, but those that cease to be discriminating due to low cue validity, $P(C|F)$, have a negligible impact on categorization.¹² In addition, experience may dictate that the benefits of certain 'deep' features outweigh their costs. These features are inferred and influence the categorization of new problems. This inference is context dependent and occurs as a problem is classified (Seifert, 1989).

At this stage in our research, it appears that EXOR is *roughly* consistent with human behavior. Our approach shares much in common with a rational analysis (Anderson & Matessa, this volume): the construction of EXOR is largely theory driven and assumes that human behavior is guided by principles of bounded rationality (Simon, 1969). That is, humans tend to optimize a *tradeoff* of cost and correctness. Thus, a reasonable approach to cognitive modeling is to propose bounded-rational models and verify their consistency by fitting them to the human data, usually by an iterative process of refinement. Importantly, this chap-

12. Fisher and Pazzani (Chapter 1, this volume) detail the way in which the evaluation metric used by EXOR diminishes the influence of nondiscriminating features.

ter does not include the latter verification step and thus we will stop short of characterizing it as a rational analysis. Rather, our work so far can be best characterized as a *speculative* cognitive model (Hall & Kibler, 1985), since we have proposed a mechanism that satisfies the rough constraints of a high-level human behavior.

6. Concluding Remarks

In summary, EXOR abstracts redundant explanation substructures and organizes them hierarchically for reuse. This yields demonstrable advantages in terms of problem-solving efficiency. More generally, our research seeks to unify principles of inductive and explanation-based learning. For instance, explanation-based concerns with efficiency can be cast in terms of the inductive performance dimension of prediction accuracy: accurate prediction along search choice points results in a more efficient search (Carlson, Weinberg, & Fisher, 1990; Fisher & Chan, 1990). This relationship was recognized in earlier work that treated search-control learning as a problem of concept induction (Mitchell, Utgoff, & Banerji, 1983; Langley, 1985), but we have strengthened the connection in several ways. Notably, inductive principles of similarity and inductive mechanisms of clustering and pruning are equally applicable to problem-solving issues of selective utilization and operability.

In addition, we have been concerned with common principles of AI and human problem solving. EXOR's mechanisms were motivated by psychological data, as well as by computational concerns of cost effectiveness. We believe that using one system as the starting point for distinct but related lines of research in computational and cognitive directions illustrates the utility of rational analyses for those at the interface of AI and cognitive psychology.

Of course, there are many areas that we have not addressed. Perhaps the most critical assumption that we have made is that the domain theory is *complete*: EXOR relies on the domain theory early in learning, but even after extensive training the system relies on the domain theory to complete solutions to idiosyncratic problems. Work in the area of incomplete domain theories (Sleeman, Hirsh, Ellery, & Kim, 1990; Bergadano & Giordana, 1988) builds inductive associations if an explanatory gap cannot otherwise be bridged with known domain rules. Incomplete domain theories will require that EXOR be able to match trees across several levels, since gaps may exist in the explanatory structure of each.

Another important research topic concerns the problem of *inconsistent* domain theories. We have not explored EXOR's tolerance of inconsistency, but we believe that the system's inductive mechanisms will render it robust on this dimension. To test this conjecture, we are investigating the use of EXOR for purposes of fault diagnosis. In particular, one part of many engineering projects (e.g., designing a purifier/pump system) is the construction of a *fault tree* (Malasky, 1982). This is an AND/OR tree that describes the possible events, singly and in combination, that lead to a top-level fault (e.g., loss of pump flow). Search for causes in this AND/OR space is analogous to search through a domain theory for explanations, and is thus amenable to speedup by EXOR. However, as a human-engineered artifact, there are often inconsistencies in the fault tree. Thus, we plan to use the fault tree as an initial domain theory, but to use EXOR to organize experience and more efficiently guide diagnosis. Logical inconsistencies may remain or they may be 'pruned' out, but in either case explanation patterns that better reflect the system's true behavior should come to statistically dominate EXOR's reasoning. Thus, this approach to inconsistent domain theories is similar in intent to systems like Towell, Shavlik, and Noordewier's (1989) knowledge-based neural network, albeit with very different approaches to the inductive learning component.

Acknowledgements

This research was supported by Grant NCC 2-645 from NASA Ames Research Center. We thank Mike Pazzani and Pat Langley for helpful comments on an earlier draft. A shorter version of this chapter appears in the *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*.

References

- Anderson, J. R., & Kline, P. J. (1979). A learning system and its psychological implications. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (pp. 16-21). Vancouver, BC, Canada: Morgan Kaufmann.
- Bergadano, F., & Giordana, A. (1988). A knowledge intensive approach to concept induction. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 305-317). Ann Arbor, MI: Morgan Kaufmann.

- Braverman, M., & Russell, S. (1988). Boundaries of operability. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 221–234). Ann Arbor, MI: Morgan Kaufmann.
- Carlson, B., Weinberg, J., & Fisher, D. (1990). Search control, utility, and concept induction. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 85–92). Austin, NY: Morgan Kaufmann.
- Cheeseman, P. (1988). An inquiry into computer understanding. *Computational Intelligence*, 4, 58–66.
- Chi, M. T., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5, 121–152.
- DeGroot, A. D. (1966). Perception and memory versus thought: Some ideas and recent findings. In B. Kleinmuntz (Ed.), *Problem solving: Research, methods, and theory*. New York: John Wiley.
- Etzioni, O. (1988). Hypothesis filtering: A practical approach to reliable learning. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 416–429). Ann Arbor, MI: Morgan Kaufmann.
- Faries, J. M., & Reiser, B. J. (1988). Access and use of previous solutions in a problem solving situation. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 433–439). Montreal, Canada: Lawrence Erlbaum.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139–172.
- Fisher, D. H., & Chan, P. (1990). Statistical guidance in symbolic learning. *Annals of Mathematics and Artificial Intelligence*, 2, 135–148.
- Flann, N. S., & Dietterich, T. G. (1989). A study of explanation-based methods for inductive learning. *Machine Learning*, 4, 187–226.
- Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11–62.
- Gluck, M., & Corter, J. (1985). Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum.

- Hall, R., & Kibler, D. (1985). Differing methodological perspectives in artificial intelligence. *AI Magazine*, 6, 166–178.
- Hirsh, H. (1989). Combining empirical and analytical learning with version spaces. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 29–33). Ithaca, NY: Morgan Kaufmann.
- Holder, L. (1990). The general utility problem in machine learning. *Proceedings of the Seventh International Machine Learning Conference* (pp. 402–411). Austin, TX: Morgan Kaufmann.
- Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3, 285–317.
- Keller, R. M. (1988). Defining operability for explanation-based learning. *Artificial Intelligence*, 35, 227–241.
- Kolodner, J. L. (1983). Reconstructive memory: A computer model. *Cognitive Science*, 7, 281–328.
- Kolodner, J. L. (1987). Extending problem solver capabilities through case-based inference. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 167–178). Irvine, CA: Morgan Kaufmann.
- Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science*, 9, 217–260.
- Lebowitz, M. (1982). Correcting erroneous generalizations. *Cognition and Brain Theory*, 5, 367–381.
- Malasky, S. (1982). *System safety: Technology and application*. New York: Garland STPM Press.
- Markovitch, S., & Scott, P. D. (1989). Information filters and their implementation in the SYLLOG system. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 404–407). Ithaca, NY: Morgan Kaufmann.
- Mayer, R. (1981). Frequency norms and structural analysis of algebra story problems into families, categories, and templates. *Instructional Science*, 10, 135–175.
- Medin, D. L. (1989). Concepts and conceptual structure. *American Psychologist*, 44, 1469–1481.
- Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42, 363–392.

- Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based learning: A unifying view. *Machine Learning*, 1, 47–80.
- Mitchell, T. M., Utgoff, P. E., & Banerji, R. (1983). Learning problem solving heuristics by experimentation. In R. S. Michalski, T. M. Mitchell, & J. G. Carbonell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Mooney, R. (1989). The effect of rule use on the utility of explanation-based learning. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 725–730). Detroit, MI: Morgan Kaufmann.
- Pazzani, M. J. (1990). *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods*. Hillsdale, NJ: Lawrence Erlbaum.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.
- Ross, B. (1987). This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13, 629–639.
- Segre, A., Elkan, C., & Russell, A. (1991). A critical look at experimental evaluations of EBL. *Machine Learning*, 6, 183–195.
- Seifert, C. M. (1989). A retrieval model using feature selection. *Proceedings of the Sixth International Machine Learning Workshop* (pp. 52–54). Ithaca, NY: Morgan Kaufmann.
- Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.
- Sleeman, D., Hirsh, H., Ellery, I., & Kim, I. (1990). Extending domain theories: Two case studies in student modeling. *Machine Learning*, 5, 11–37.
- Towell, G., Shavlik, J., & Noordewier, M. (1990). Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 861–866). Boston, MA: Morgan Kaufmann.
- Warren, D. (1981). Efficient processing of interactive relational database queries expressed in logic. *Proceedings of the Seventh International Conference on Very Large Data Bases*.
- Wilkins, D. (1980). Using patterns and plans in chess. *Artificial Intelligence*, 14, 165–203.

Part III

The Utility of Concept Formation in Intelligent Behavior

CHAPTER 11

Concept Formation in Context

DOUG FISHER

MICHAEL PAZZANI

1. Introduction

Thus far, this volume has focused on theoretical issues of concept formation, including representation languages, search strategies, measures of similarity, and the utility of background knowledge. However, to be useful, concept formation cannot occur in a vacuum. Rather, it should facilitate performance in some context (e.g., problem solving).

This chapter surveys applications of concept formation. We also include related learning tasks in cases where they are relevant to directions in concept formation research. For example, nonincremental, unsupervised systems have long been used as tools in exploratory data analysis (Everitt, 1980; Jain & Dubes, 1987). In this setting, clustering is a tool for discovering patterns in data that must be interpreted and exploited by a human data analyst. In part, some concept formation research can be viewed as partially automating the analyst's interpretation task, thus motivating discussion of the earlier data analysis work.

Our survey divides research into five major areas — discovery and exploration, problem solving and planning, engineering applications, natural language processing, and information retrieval. Inevitably, any such division is unsatisfying, since there are principles that are common across areas, but we point these out as appropriate. In addition, our survey includes research in areas like scientific discovery that are not traditionally regarded as concept formation processes, but which we nonetheless feel can be described as such.

2. Discovery and Exploration

As we have noted, clustering has traditionally been used as a tool in data analysis. For example, clustering facilitates both hypothesis generation and testing during scientific inquiry. Romesburg (1984) provides an excellent survey of many such applications, but let us consider only two. Sigleo (1975) clustered over descriptions of prehistoric native-American artifacts from various sites. This clustering revealed that in some cases the most similar artifacts were from relatively distant sites, suggesting that the *trading patterns of the participating cultures were not localized*. This is an example of hypothesis generation, a process of working backward from the data in search of an explanatory hypothesis. In artificial intelligence, this is referred to as *abduction* (O'Rorke, Morris, & Schulenburg, 1989). In data analysis, it has been referred to as *retroduction*. A study that is similar to, but independent from, Sigleo's illustrates hypothesis testing. In this case the researchers (Charlton, Grove, & Hopke, 1978) tested a hypothesis suggested by others that prehistoric craftsmen at one site did not have access to obsidian mined at another site. If true, (irradiated) obsidian samples from the first site should cluster to different parts of a hierarchical classification than should samples from the mining site. In fact, this did not occur consistently, thereby disconfirming the hypothesis and suggesting that the retroductively suggested hypothesis be reexamined.

The applications of traditional clustering methods are extensive and suggest that unsupervised methods from machine learning can be applied in a similar way (Biswas, Weinberg, Yang, & Koller, 1991). For example, Michalski and Stepp's (1983) CLUSTER/2 was used to discover categories of Spanish folk songs based on various musicological attributes (e.g., harmonic structure, tonal range). They did not detail hypotheses that were generated or tested after the clustering process, but such an analysis could clearly be performed. In fact, the concepts associated with discovered categories were 'helpful' to a participating domain expert, presumably because they facilitated the generation and testing of hypotheses. Similarly, Cheeseman, Kelly, Self, Stutz, Taylor, and Freeman (1988) have used their AUTOCLASS system to discover classes of astronomical objects from their infrared spectra. The system found classes that were previously unexplored by astronomers, though these classes appear to suggest useful hypotheses about the objects under study (Cheeseman et al., 1989).

In addition to clustering systems for numerical taxonomy and machine learning, researchers have developed a number of systems that are generally described as pursuing scientific discovery. These systems tend to make the interface between hypothesis generation and testing more explicit. For example, we have noted elsewhere (Fisher & Pazzani, Chapters 1 and 6, this volume) that the GLAUBER system (Langley, Zytkow, Simon, & Bradshaw, 1986) generates qualitative laws (e.g., acids and alkalis react to form salts) through a process of conceptual clustering. The acquisition of rules from data corresponds to hypothesis generation. Jones' (1986) NGLAUBER adds a prediction component to the earlier system, which lets it anticipate and verify the behavior of future data based on existing rules. Thus, Jones' system performs a primitive type of hypothesis testing as well.

Hypothesis testing plays an even greater role in several discovery systems that autonomously perform experiments. Perhaps the earliest such system was Lenat's (1977) AM, which used heuristics to generate 'interesting' domain-specific hypotheses. For example, one of AM's heuristics can be paraphrased as 'if a relation exists, then explore instances of it with extreme values'. In the realm of number theory, this heuristic causes the system to investigate numbers with very few divisors (e.g., zero through three divisors, which are 'extreme' values in a 'divisors-of' relation). Of these, numbers with exactly two divisors (i.e., prime numbers like 7, which only have 1 and itself as divisors) are found to hold interest, as are numbers with three exactly divisors (e.g., 49 with 1, 7, and 49 as divisors). In addition, AM notices relationships between discovered classes. Thus, numbers with exactly three divisors are perfect squares (e.g., 49), and include a prime number (e.g., 7) as a divisor. This hypothesis can be empirically confirmed or disconfirmed by searching for more examples of primes and perfect squares, then noting whether the relationship persists.

Other systems also actively explore their environment. For example, Scott and Markovitch's (this volume) DIDO learns the effects of operators by applying them, making initial hypotheses about operator effects, and confirming these hypotheses through continued experimentation. The system includes a concept formation component that organizes discovered concepts in a way that facilitates hypothesis generation and testing. Of course, many other discovery systems possess some degree of autonomy, but only some rely explicitly on the incremental clustering abilities that define concept formation systems. Systems

like NGLAUBER, AM, and DIDO meet these criteria, as do a few others, such as LIVE (Shen, 1989) and IDS (Nordhausen, 1989).

3. Planning and Problem Solving

Data analysis and discovery are primarily concerned with characterizing an environment. These processes often leave implicit the tasks that are facilitated through discovery. In contrast, planning and problem solving are explicit about the goals to be achieved. They are concerned with learning and exploiting a picture of the environment with specific ends in mind. In these contexts, concept formation and other forms of concept learning also play an important role.

Romesburg (1984) describes several applications in which clustering methods from numerical taxonomy are subservient to planning and problem-solving tasks. An application by McCormick, Schweitzer, and White (1972) illustrates the basic strategy. They clustered transport aircraft based on a large number of 'functional' attributes, including "ability to function as an aerial refueler, as an assault transport, as an aeromedical transport, as a personnel transport, as a utility transport, and so forth" (Romesburg, pp. 69-70, 1984). The intent of this study was to discover categories of aircraft that were roughly interchangeable. These categories would be useful in making dynamic decisions about aircraft assignments and scheduling, as well as recommending longer-term maintenance schedules and the best 'mix' of aircraft at an airbase.

As in exploratory data analysis, a human user is responsible for a considerable portion of the overall clustering task in planning contexts. In the above example, a suitable language of 'functional' features must be devised that is based on the user's goals. A similar but somewhat different responsibility is imposed on the user of an AI system like CLUSTER/CA (Stepp & Michalski, 1986). Recall from Fisher and Pazzani's discussion (Chapter 6, this volume) that this system uses a *goal dependency network* to link system goals (e.g., maintain environmental integrity) with descriptive features (e.g., a congressperson's vote on off-shore oil and automobile emission standards) of objects to be classified. Discovered classes (of members of congress) are presumably useful in decision making (voting). Functional descriptors are not explicitly used in clustering, but they form an integral part of the dependency network that is constructed by a user.

These examples suggest the relevance of ‘functional clustering’ (Fisher & Pazzani, Chapter 6, this volume) in cases where it seems most useful to cluster objects based on the roles they share during problem solving and planning. Consider early work on learning from solution traces (Mitchell, Utgoff, & Banerji, 1983; Langley, 1985). For each problem-solving operator, such as ‘integrating by parts’ in the domain of integral calculus, states were ‘clustered’ into positive and negative classes based on whether the operator was recommended in the presence of that state. An inductive (Mitchell et al., 1983; Langley, 1985) or analytic (Mitchell, Keller, & Kedar-Cabelli, 1986) method for supervised learning was then used to find a general rule that guided future operator applications.

This early work used supervised techniques because the applicability conditions of each operator were learned independently — states fell into one of two categories. However, McCormick et al.’s example indicates that there are many cases where one wants to simultaneously consider many of the roles in which an object or state can participate. In these situations one does not know *a priori* what categories might best distinguish objects or states for purposes of problem solving. Thus, unsupervised methods are most relevant.

Let us consider Allen and Langley’s (1990) DAEDALUS and Yang and Fisher’s PLOT (1989) as examples of concept formation in planning. In each case, instances of STRIPS-like operators are clustered to improve the efficiency of means-ends planning. Operators are represented by *preconditions* that must be true of the world immediately before an operator can be applied, *add* properties that are true immediately after application, and *delete* properties that are *not* true immediately after an operator is applied. Operators are chosen by their ability to reduce differences between a current state and a goal state. Abstraction in these systems improves efficiency for some of the same reasons that abstraction in other systems does (Sacerdoti, 1974; Knoblock, 1990): they remove details that can be initially overlooked during planning. However, DAEDALUS and PLOT are distinguished by their reliance on probabilistic concepts, since the anticipated effects of abstract operators may not cooccur with certainty. This type of representation is better suited to planning and acting in uncertain environments (Fisher & Pazzani, Chapter 1, this volume).

DAEDALUS and PLOT borrow from earlier systems that can also be viewed as clustering operator descriptions, notably Anderson and Far-

ley's (1988) PLANEREUS and Vere's (1978) THOTH. The former system learns operator classes based exclusively on shared add properties, whereas THOTH induces a set of primitive operator descriptions (i.e., preconditions, adds, and deletes) from examples of before-after state transitions. In addition, many other systems learn operator effects through clustering and concept formation (e.g., Carlson, Weinberg, & Fisher, 1990). As we have noted, Scott and Markovitch's (this volume) DIDO actively explores operator effects in terms of preconditions and results. Iba and Gennari's (this volume) OXBOW learns motor schemas by observing physical movements. In contrast to the discrete sequence of states assumed by THOTH, OXBOW's task is complicated by the continuous nature of motion. Simon, Newell, and Klahr's (this volume) Q-SOAR models the development of operator effects in children. In particular, they are concerned with a child's acquisition of 'number conservation'. Stacking blocks does not change the number of blocks, though it does change their collective height, and children must learn to distinguish these effects.

In addition to learning about operator effects, there are other contexts in which clustering is used to improve problem solving. A notable example is Rendell, Seshu, and Tcheng's (1987) PLS system, which clusters problem descriptions based on their characteristics (e.g., scores on various intrinsic measures) and functional aspects. PLS' task is somewhat like that described earlier: the system must cluster objects (i.e., 'problem states') based on intrinsic and functional properties, where functionality is summarized by 'performance' scores, as measured by distance to a desired state.

Before closing this section, it is worth touching upon an apparent anomaly. Wisniewski and Medin (this volume) point out that natural and artificial learning systems, including those that are 'unsupervised', should be considered in some context (automated or otherwise). However, context implies environmental feedback that should help direct or 'supervise' learning. In this light we believe that the 'unsupervised' label is a misnomer, since these systems do not disallow feedback or 'supervision'. Rather, 'unsupervised' systems assume that data must be compressed relative to their values along multiple dimensions, whether these be 'feedback' dimensions, intrinsic dimensions, or some combination of both. In contrast, supervised learning systems dictate categorization by values along a single dimension. In PLS, intrinsic properties correspond to problem characteristics, and 'feedback' dimensions correspond to val-

ues obtained by multiple learning algorithms. In DAEDALUS and PLOT, feedback dimensions correspond to the differences that determine the roles played by each operator in the final structure of a plan.

Reich and Fenves (this volume) address similar issues. In their view, supervised systems assume a many-to-one mapping between intrinsic properties of observations and 'feedback'. That is, objects may be described along many attributes, but system performance is only tested against a single 'class' attribute. In contrast, unsupervised systems must manage a many-to-many mapping. Even systems not traditionally linked to context-dependent processing assume a many-to-many mapping between the descriptive features that guide clustering and performance tasks applied to resultant classes. For example, recall the attribute-prediction task (Fisher & Pazzani, Chapter 1, this volume) used in conjunction with systems like COBWEB (Fisher, 1987). In this case, the 'language' of descriptive and 'performance' attributes are the same, but the many-to-many characteristic remains.

In sum, feedback alone does not imply 'supervision' in the traditional, technical sense. Rather, 'unsupervised' learning compresses data based on information along many dimensions, which is typically some combination of intrinsic object properties and externally supplied feedback.

4. Engineering Applications

There has been much recent interest in the use of concept learning systems, supervised and unsupervised, in engineering applications. In most cases, these applications are goal directed and might reasonably be viewed as planning or problem-solving tasks. However, many engineering tasks lack clear-cut criteria for success and the problem space that defines the possible engineering solutions may be only partially defined. Reich and Fenves (this volume) argue that this is the case in many problems of engineering design. In particular, they focus on the initial stages of bridge design. Their BRIDGER system carries out concept formation over known bridge specifications and designs. When presented with a new set of specifications, the system retrieves candidate designs that best match the known specifications. The same idea underlies the use of concept formation to promote software reuse. Arango (1988) clusters over descriptions of software components in an attempt to facilitate their reuse in new software designs.

Clustering has also been applied to the problem of diagnosis. Wu (1990) uses a form of clustering to improve the efficiency of diagnosis in model-based reasoning by discovering correlated symptoms and using these to constrain the search for possible causes. Although Wu and others (e.g., Cheng & Fu, 1985) have investigated the applicability of the approach to medical diagnosis, it appears useful for fault diagnosis in engineering applications as well. Along similar lines, Yoo and Fisher (this volume) have proposed using their EXOR system to improve diagnosis with fault trees.

Another notable concern in many engineering problems is the need to handle numeric data (Whitehall, Lu, & Stepp, 1990). This issue is also addressed by Reich and Fenves (this volume), but it is problematic in many engineering tasks. For example, Chen and Shiavi (1990) use concept formation and traditional clustering procedures in a biomedical engineering application involving the classification of human gaits. Presumably, the discovery of appropriate classification schemes would facilitate the identification of appropriate remediation strategies for individuals with different kinds and degrees of handicaps or abnormalities. In this case, signals from different muscle groups of a human subject are recorded during walking. The signal from each muscle group is partitioned and becomes a component in an 'object' description. The engineering of an appropriate representation language consumes considerable effort in this research, much of it concerned with analyzing and representing the analog signals from muscle groups in motion. To carry out clustering in this domain, Chen and Shiavi employ a method and representation much like that described by Thompson and Langley (this volume).

In addition to concerns with numeric data, many engineering tasks require an ability to handle relational descriptions (see Fisher & Pazzani, Chapter 1, this volume). For example, if one is interested in diagnosing faults in physical devices, then one must be concerned with relationships among components. Many early machine learning approaches to supervised learning were concerned with relational representations (Dietterich & Michalski, 1983). Recently, there has been a resurgence of research in this area, with special attention given to issues of efficiency and noise tolerance by employing principled statistical guidance. Thus, systems such as Quinlan's (1990) FOIL appear to hold considerable promise in various engineering applications. In concept formation there has also been some interest in relational descriptions for engineering applications. For ex-

ample, Lebowitz' (1986) RESEARCHER clusters descriptions of physical devices such as disk drives. In addition, Yoo and Fisher's (this volume) EXOR has limited relational capabilities, thus making it suitable for some fault diagnosis tasks. However, many engineering applications must await methods that can handle relational and numeric data.

5. Natural Language Processing

Language is another domain in which clustering methods have traditionally been applied in various ways. A notable example is Wolff's (1982) SNPR system, which clusters over a continuous stream of letters. The system determines boundaries between letter strings that identify 'words', abstracts over these strings to identify parts of speech (e.g., nouns and verbs), and groups patterns of word classes into phrases, thereby identifying a grammar consistent with the 'training' text. The system uses a strategy much like Langley et al.'s (1986) GLAUBER, which was significantly influenced by SNPR. The system's behavior is impressive, particularly given the semantic-free strategies that it uses. In fact, one of Wolff's goals was to explore the extent to which these methods could account for human language learning. This research illustrates that natural language sentences have surface regularities which facilitate the acquisition of syntactic structures.

Another system that deals with natural language input is Lebowitz' (1983) IPP, which reads newspaper stories about international terrorism and makes generalizations from them. Its major contribution concerns the role of episodic memory in text understanding. IPP constructs a number of clusters that represent specializations of terrorism stories it has read. These specializations provide internally generated expectations that guide the process of interpreting and disambiguating new stories. For example, after reading a number of stories, IPP created a generalization describing the fact that shootings in Italy often result in leg wounds. Thus, the system could predict that there might be a leg wound when reading about an Italian shooting. This prediction could enable the parser to select the proper sense of the word 'calf', deciding on 'part of the leg' rather than 'young cow'.

Along similar lines, Lebowitz' (1985) UNIVERSE, an extension of IPP, has been used to generate stories. UNIVERSE finds commonalities among the plots of soap-opera stories. The stereotypic structure of these tele-

vision programs yields clear-cut story lines through a process of concept formation. The discovered classifications and 'rules' that describe each class are then used to generate soap opera plots.

6. Information Retrieval

Finally, let us consider the application of concept formation and other clustering methods to information retrieval. This has long been an area of application for traditional clustering methods (Van Rijsbergen, 1979), where documents are clustered based on keywords that have been explicitly provided by the authors or automatically extracted from the title, abstract, and text. The assumption is that documents which are deemed similar in this way will be relevant to the same queries on the part of users (e.g., in a library). Ideally, clustering should increase the efficiency and effectiveness of retrieval, where effectiveness is generally regarded as a function of two quantities. *Precision* is the proportion of retrieved documents that are deemed by the user as relevant to a query, and *recall* is the proportion of relevant documents that are actually retrieved. One wishes to maximize both precision and recall to the extent possible.

Information retrieval has also been of great concern in machine learning, particularly in work on concept formation. Research in this area generally focuses on issues of representation language and/or human-machine interface. For example, graphical and relational representations considerably expand the descriptive flexibility of queries and knowledge base entries relative to attribute-value or keyword representations. Graph clustering has been used to store molecular structures that could be retrieved efficiently by chemists during experimental design and analysis (Wilcox & Levinson, 1986). Lebowitz (1983) describes IPP and RESEARCHER as information retrieval systems with particular attention to the natural language interface of IPP and the relational representations that lets RESEARCHER store and recall patent abstracts of disk drives.

A number of case-based systems have also been used for information retrieval, though these can also be viewed as carrying out concept formation. A highly influential system in this regard is Kolodner's (1983) CYRUS, which incrementally clusters descriptions of events (e.g., from newspaper stories) in a manner that facilitates question answering. In

particular, CYRUS' memory encodes idealized experiences of former Secretary of State Cyrus Vance, such as diplomatic trips and negotiations. The organization of memory supports many types of queries, ranging from questions about which countries have been visited to the purpose of specific trips. Many systems have built on CYRUS's general information storage and retrieval strategies. A particularly novel application is Swaminathan's (1990) RA, which is an automated research advisor for graduate study; based on an initial interaction with a user on research interests, RA points to related work and suggests a research program based on the methodologies followed by this earlier work.

In general, machine learning approaches to information retrieval support flexible question answering by exploiting domain-independent inference strategies such as inheritance, as well as domain-dependent inference rules. For example, in the congressional voting example used to illustrate CLUSTER/CA, we might infer that a member of congress votes along 'party lines' unless we explicitly know otherwise. In addition, machine learning methods are designed to improve precision and recall. However, concern with these measures is generally expressed in other terms like feature *predictability* and *predictiveness*, which are summarized by Fisher and Pazzani (Chapter 1, this volume) in their description of CYRUS, UNIMEM, and related systems.

The primary focus of current research is on extending representations and inference strategies that support flexible response to users' queries. In some sense, all of the applications considered in this chapter exploit concept formation as a mechanism for intelligent information storage and retrieval, though historically this area of study has been segregated from others. This is an unfortunate precedent that we cannot hope to correct here, but there are encouraging signs that research in information retrieval, machine learning, and concept formation will benefit from scientific exchange (e.g., Owens et al., 1991).

7. Conclusion

This chapter has briefly reviewed some contexts in which concept formation can play a role. These included discovery and data analysis, planning and problem solving, engineering design and diagnosis, natural language processing, and information retrieval. The remaining chapters explore some of these endeavors in greater depth. Despite the recent de-

velopment of the concept formation paradigm, there appear to be many opportunities for practical or theoretical contributions to other areas of scientific inquiry, and we expect more to emerge in years to come.

Acknowledgements

The first author was supported by Grant NCC 2-645 from NASA Ames Research Center, and the second author was supported by Grant IRI-8908260 from the National Science Foundation. We thank Pat Langley for helpful comments on an earlier draft.

References

- Allen, J. A., & Langley, P. (1990). Integrating memory and search in planning. *Proceedings of the 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 301-312). San Diego, CA: Morgan Kaufmann.
- Anderson, J. S., & Farley, A. M. (1988). Plan abstraction based on operator generalization. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 100-104). St. Paul, MN: AAAI Press.
- Arango, G. (1988). *Domain engineering for software reuse*. Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.
- Biswas, G., Weinberg, J. B., Yang, Q., & Koller, G. R. (1991). *Conceptual clustering and exploratory data analysis* (Tech. Rep. No. CS-91-03). Nashville, TN: Vanderbilt University, Department of Computer Science.
- Carlson, B., Weinberg, J., & Fisher, D. (1990). Search control, utility, and concept induction. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 85-92). Austin, TX: Morgan Kaufmann.
- Charlton, T. H., Grove, D. C., & Hopke, P. K. (1978). The Paredon, Mexico, obsidian source and early formative exchange. *Science*, 201, 807-809.

- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. *Proceedings of the Fifth International Machine Learning Conference* (pp. 54–64). Ann Arbor, MI: Morgan Kaufmann.
- Cheeseman, P., Goebel, J., Self, M., Stutz, J., Volk, K., Taylor, W., & Walker, H. (1989). *Automatic classification of the spectra from the infrared astronomical satellite (IRAS)* (Reference Publication 1217). Washington, DC: National Aeronautics and Space Administration.
- Chen, J. J., & Shiavi, R. G. (1990). Temporal feature extraction and clustering analysis of electromyographic linear envelopes in gait studies. *IEEE Transactions on Biomedical Engineering*, 37, 295–302.
- Cheng, Y., & Fu, K. (1985). Conceptual clustering in knowledge organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7, 592–598.
- Everitt, B. (1981). *Cluster analysis*. London: Heinemann.
- Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, 8, 305–336.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139–172.
- Hammond, K. (1987). *Case-based planning: An integrated theory of planning, learning, and memory*. San Diego, CA: Academic Press.
- Holder, L. B. (1988). *Discovering substructure in examples*. Master's thesis, Department of Computer Science, University of Illinois, Urbana.
- Jain, A. K., & Dubes, R. C. (1988). *Algorithms for cluster analysis*. Englewood Cliffs, NJ: Prentice Hall.
- Jones, R. (1986). Generating predictions to aid the scientific discovery process. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 513–517). Philadelphia, PA: AAAI Press.
- Knoblock, C. A. (1990). Learning abstraction hierarchies for problem solving. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 923–928). Boston, MA: AAAI Press.
- Kolodner, J. L. (1983). Reconstructive memory: A computer model. *Cognitive Science*, 7, 281–328.

- Kolodner, J. L. (1987). Extending problem solver capabilities through case-based reasoning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 167–178). Irvine, CA: Morgan Kaufmann.
- Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science*, 9, 217–260.
- Langley, P., Thompson, K., Iba, W., Gennari, J. H., & Allen, J. A. (1989). An integrated cognitive architecture for autonomous agents. In W. Van De Velde (Ed.), *Representation and learning in autonomous agents*. Amsterdam: North Holland.
- Langley, P., Zytkow, J. M., Simon, H. A., & Bradshaw, G. L. (1986). The search for regularity: Four aspects of scientific discovery. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Lebowitz, M. (1983). Intelligent information systems. *Proceedings of the Sixth International ACM SIGIR Conference* (pp. 25–30). Bethesda, MD: ACM.
- Lebowitz, M. (1985). Story telling and generalization. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 100–109). Irvine, CA: Lawrence Erlbaum.
- Lebowitz, M. (1986). Concept learning in a rich input domain. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2, 103–138.
- Lenat, D. B. (1977). Automated theory formation in mathematics. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 833–841). Cambridge, MA: Morgan Kaufmann.
- McCormick, W. T., Schweitzer, P. J., & White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20, 993–1009.
- Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.

- Mitchell, T. M., Utgoff, P. E., & Banerji, R. B. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1, 47–80.
- Mogensen, B. (1987). *Goal-oriented conceptual clustering: The classification attribute approach* (Tech. Rep. No. UILU-ENG-87-2257). Urbana: University of Illinois, Department of Computer Science.
- Nordhausen, B. (1989). *A computational framework for empirical discovery*. Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.
- O'Rorke, P., Morris, S., & Schulenburg, D. (1989). Theory formation by abduction: Initial results of a case study on the chemical revolution. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 266–271). Ithaca, NY: Morgan Kaufmann.
- Owens, C., Lewis, D. D., Belkin, N., Croft, W. B., Hunter, L., & Waltz, D. (1991). Learning in intelligent information retrieval. *Proceedings of the Eighth International Workshop on Machine Learning*. Evanston, IL: Morgan Kaufmann.
- Porter, B. W., Bareiss, E. R., & Holte, R. C. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45, 229–263.
- Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5, 239–266.
- Rendell, L., Seshu, R., & Tcheng, D. (1987). More robust concept learning using dynamically-variable bias. *Proceedings of the Fourth International Machine Learning Workshop* (pp. 66–78). Irvine, CA: Morgan Kaufmann.
- Romesburg, H. C. (1984). *Cluster analysis for researchers*. Belmont, CA: Lifetime Learning Publications.
- Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5, 115–135.
- Shen, W. M. (1989). *Learning from the environment based on percepts and actions*. Doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

- Sigleo, A. C. (1975). Turquoise mine and artifact correlation for Snaketown Site, Arizona. *Science*, 189, 459–460.
- Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Swaminathan, K. (1990). RA: *A memory organization to model the evolution of scientific knowledge*. Doctoral dissertation, Department of Computer and Information Science, University of Massachusetts, Amherst.
- Van Rijsbergen, C. (1979). *Information retrieval*. Boston, MA: Butterworths.
- Vere, S. A. (1978). Inductive learning of relational productions. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press.
- Whitehall, B. L., Lu, S. C.-Y., & Stepp, R. E. (1990). CAQ: A machine learning tool for engineering. *Artificial Intelligence in Engineering*, 5, 189–198.
- Wilcox, C. S., & Levinson, R. A. (1986). A self-organized knowledge base for recall, design, and discovery in organic chemistry. In T. H. Pierce & B. A. Hohne (Eds.), *Artificial intelligence applications in chemistry*. Washington, DC: American Chemical Society.
- Wogulis, J., & Langley, P. (1989). Improving efficiency by learning intermediate concepts. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 657–662). Detroit, MI: Morgan Kaufmann.
- Wolff, J. G. (1982). Language acquisition, data compression, and generalization. *Language and Communication*, 2, 57–89.
- Wu, T. D. (1990). Efficient diagnosis of multiple disorders based on a symptom clustering approach. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 357–364). Cambridge, MA: AAAI Press.
- Yang, H., & Fisher, D. (1989). Conceptual clustering of means-ends plans. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 232–234). Ithaca, NY: Morgan Kaufmann.

CHAPTER 12

The Formation and Use of Abstract Concepts in Design

YORAM REICH

STEVEN J. FENVES

1. Introduction

Design is an ill-structured problem, in which there are often vague criteria for success, the search space is ill defined, and resources like time are limited. Building expert systems for design, henceforth called *design systems*, requires domain knowledge and representation schemes that support the solution of this inherently ill-structured problem. However, manual knowledge acquisition has been a bottleneck in creating expert systems in general and design systems more specifically. One approach to alleviating some of the difficulties in knowledge acquisition is the introduction of learning into the development and maintenance stages of expert systems (Reich & Fenves, 1989; Steels & Van de Velde, 1986). Learning can assist in automatically building design systems, as well as in understanding a specific design domain.

The principal focus of this chapter is on acquiring expert design knowledge. One promising solution uses *inductive* methods for supervised and unsupervised concept learning. This chapter examines the relationships between these learning strategies and design. Section 2 begins this process by identifying design characteristics that constrain the acquisition of design knowledge. Section 3 shows that supervised concept learning provides limited support for acquiring design knowledge, and illustrates that unsupervised methods of concept formation may be more suitable to the task. Section 4 describes BRIDGER, a system that integrates concept formation into design. Section 5 eval-

ates BRIDGER in a number of domains, and provides a framework for understanding the relationship between design and concept formation.

2. The Nature of Design

The product of design is the description of an artifact (called the *design description*) that satisfies a set of specifications. The process of mapping from specifications to design descriptions is called *synthesis*. The process of mapping from design descriptions to behavior characteristics (which are essentially similar to the specification properties) is called *analysis* or *evaluation*.

Synthesis is difficult because its complexity is exponential in the number of properties describing designs. This complexity results from interactions among design properties, which often prevent accurate evaluations of partial designs. In contrast, behavioral analysis is relatively simple, since each behavior is directly dependent upon the design and is largely uncoupled from the remaining behaviors.

This chapter focuses on the acquisition of knowledge for a synthesis task in which, given a (possibly partial) specification, one must find a set of design description properties that satisfy the specification. In learning, this process corresponds to *prediction* of missing properties (i.e., the design description) from the partial description (i.e., the specification). The task is suitable for the first stages of *synthesis* in *conceptual design*, namely, the retrieval, assembly, and manipulation of relevant past solutions that can serve as a basis for a new design solution. However, before defining mechanisms for acquiring design knowledge, one must understand some of the important characteristics of design. Usually, these characteristics are only implicit in common, general-purpose definitions of design. Instead of trying to define design precisely, it is useful to view it as an *ill-defined* or *ill-structured* problem (Reitman, 1964).

2.1 Design as an Ill-Structured Problem

Simon (1973) defines a well-structured problem as having three characteristics: (1) there is a definite criterion for testing solutions; (2) there is a problem space that contains the knowledge required to solve the problem (i.e., representation of states and operators) and that can include any newly acquired knowledge about the problem; and (3) all processes

for solving the problem can be performed in a reasonable time. Design lacks all three characteristics, conforming to Simon's definition of an ill-structured problem. Let us address each characteristic in turn.

1. *Test criteria.* Design problems are almost always underspecified and loosely constrained. Not only is a definite testing criterion lacking in design, but also there is no criterion for determining when the design process should terminate. In addition, there is never a single objective in design; rather, many objectives are sought, such as minimizing cost and space and maximizing effectiveness. When multiple objectives compete, there is no clear criterion for selecting among alternatives. Usually, some compromise is made to the advantage of only some objectives. In most design problems there are additional criteria that cannot be quantified, such as aesthetics.
2. *Knowledge representation.* The solution of design problems requires the use of diverse knowledge sources from several disciplines. Each of the knowledge sources can raise different concerns regarding the design, which might give rise to multiple solution strategies. Practically speaking, it is impossible to capture this diversity *a priori* in a closed, static problem space. Rather, the problem space evolves dynamically as the solution progresses.
3. *Resources.* Design processes can only be allocated limited resources for their completion. Examples include time limits that stem from due dates for design, the total person-hours that can be spent, and the available design aids. Small problems can be solved within the available resources, and their solutions might even be optimal with respect to certain criteria. However, real design problems rarely achieve optimality. The inability to optimize, and therefore the need to satisfice (Simon, 1969) and possibly compromise, is a direct consequence of design complexities in conjunction with limited resources.

There is another aspect of design that is usually neglected, but that reinforces the characteristics described above. This aspect is that design knowledge is constantly evolving. Designers not only must cope with a complex task, but they must track the evolution of a domain. In this situation, designers determine whether new knowledge is related to the body of existing knowledge, or whether the new knowledge reflects a more fundamental change in technology. The latter may have a large effect on their problem-solving behavior.

An exemplar of theory change in bridge technology is the introduction of aerodynamic considerations due to the failure of the Tacoma Narrows Bridge in 1944. The phenomenon of dynamic instability due to wind was not widely known before this event. As a result, research and evaluation of existing bridges have proliferated, leading to drastic strengthening of some structures such as the Golden Gate Bridge. Such transparent changes are rapidly assimilated, whereas less obvious changes are harder to accommodate.

2.2 General Requirements for Learning

Learning from experience allows designers to improve their management of ill-structured design problems. The designers' particular experience determines their style and the organizational structure of their problem-solving behavior. These same adaptive mechanisms are required by tools that partially automate the acquisition of design knowledge. A selected learning method must support the full range of design activities that we discussed earlier.

1. *Test criteria.* Although there is no definite criterion for testing whether a design process ends, there are many ways to test whether a design conforms to a given specification and how it rates on evaluation scales such as economy and efficiency. Learning should generate knowledge structures that facilitate evaluation in terms of multiple criteria and objectives. They should also implicitly accommodate the subjective test criteria used by different designers.
2. *Knowledge representation.* Learning should support the acquisition and integration of diverse knowledge sources. The knowledge generated should be available to different tasks such as synthesis, analysis, explanation, and evaluation.
3. *Resources.* The limited resources available for design require that learning gradually produce knowledge that is more efficient to use. This can be achieved by two complementary mechanisms. The first mechanism assimilates new information into an existing knowledge base, and the second reorganizes knowledge based on feedback or self analysis. The knowledge generated should facilitate various strategies that effectively manage the limited resources available.

Finally, the evolutionary aspect of design should be supported by the use of incremental techniques. In light of the limited resources and the continuous flow of information, batch techniques are not suitable since they must reprocess all existing knowledge to accommodate any new piece of information.

3. Unsupervised Versus Supervised Concept Learning

This section describes learning mechanisms for acquiring and structuring basic design knowledge in a way that facilitates efficient synthesis. Since the ill-structured nature of design often precludes formalized theories of synthesis, we are tentatively biased against methods like explanation-based learning (Mitchell et al., 1986; Mooney, this volume) *during the early stages* of knowledge acquisition. On the other hand, inductive or empirical techniques do not require background knowledge (although they may use it if available), since they operate on the basis of syntactic regularity. We will not discuss manual knowledge elicitation techniques, as we wish to explore the limits of a fully automated approach that requires minimal resources, that needs little interaction with a domain expert, and that generates simple and flexible knowledge structures.

The source of information available for automatic learning is a collection of case histories. These cases are the product of articulating design knowledge, and may contain actual in-service behavior and maintenance records. Data like these are used by human designers to explain failures and to expand knowledge that enhances design practice (Magued, Bruneau, & Dryburgh, 1989). Machine learning techniques can also use case histories to produce similar results (Stone & Blockley, 1989). Another reason that the study of case histories is easier than manual expert elicitation is that experts can provide solved problems more readily than they can articulate their knowledge explicitly. This tendency leads to the preference of applying learning methods to the results of expert problem solving, in the hope of extracting expert rules that underlie the behavior (Michalski & Chilausky, 1980). Below we discuss the two primary classes of machine learning techniques that comprise the inductive approach: *supervised concept learning* and *unsupervised concept learning*.

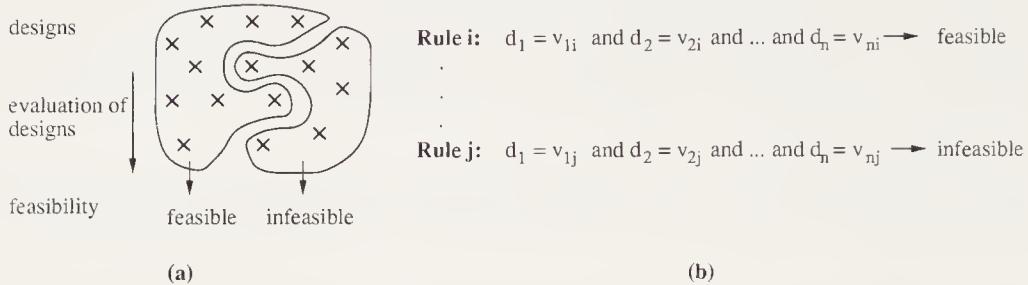


Figure 1. Classification of feasible designs.

3.1 Supervised Concept Learning

Supervised concept learning creates knowledge structures that support the task of classifying new objects into predefined classes. In the case of design, examples are represented by a list of specification property-value pairs and are classified into a set of classes that can represent a *single* design descriptor. The task is to predict this design description property based on the specification properties. Several examples of the use of supervised concept learning in simple design tasks illustrate the potential and limitations of this approach.

Arciszewski, Mustafa, and Ziarko (1987) use a supervised method to differentiate between feasible and infeasible designs. Each design is described by a list of discrete property-value pairs ($d_1 = dv_{11}, d_2 = dv_{21}, \dots, d_n = dv_{n1}, s_1 = sv_{11}$), where d_i denotes a design description property, dv_{ij} denotes one of the values of d_i , and s_1 has possible values of $sv_{11} = \text{feasible}$ and $sv_{12} = \text{infeasible}$, as in Figure 1(a). The properties d_i reflect qualitative design decisions made at the early stage of design. The goal of the acquired rule set is to predict whether or not a given combination of design description values is feasible, as in Figure 1(b).

McLaughlin and Gero (1987) present a similar approach. Instead of differentiating between feasible and infeasible designs, their task is to characterize designs that are optimal. Each design is described by a list of property-value pairs ($d_1 = dv_{11}, \dots, d_n = dv_{n1}, o_1 = ov_{11}, \dots, o_m = ov_{m1}$), where d_i denotes a design descriptor, o_j denotes an objective, and dv_{ik} or ov_{ik} denotes one of the values of d_i or o_i , respectively. Since they used a supervised concept learning technique, Quinlan's (1986) ID3, a single classification of Pareto-optimal and inferior designs is generated

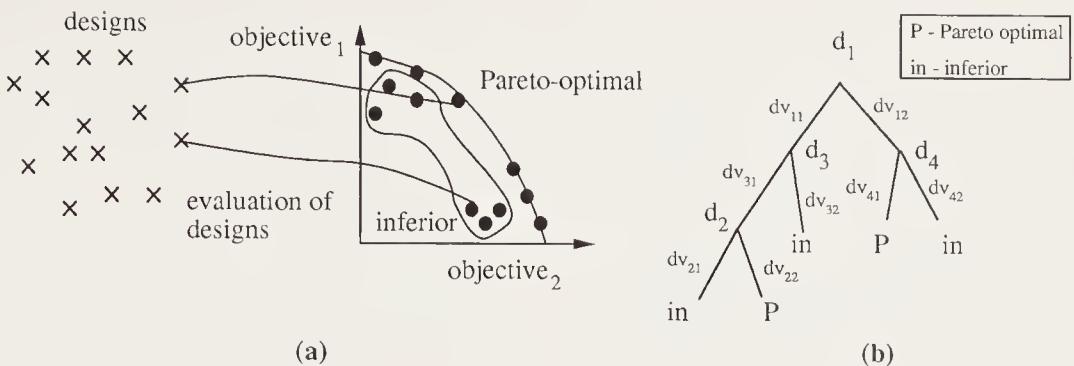


Figure 2. Differentiation of Pareto-optimal designs from inferior ones.

for each aggregation of objectives o_1, \dots, o_m . Figure 2(a) illustrates an example of this strategy for two objectives. A decision tree generated by the learning program is intended to predict whether a given design will be Pareto-optimal or inferior, as shown in Figure 2(b).

These two approaches essentially extract evaluation knowledge rather than synthesis knowledge. Neither approach synthesizes designs from specifications; instead, they map design descriptors onto behavioral descriptors, as Figure 3(a) summarizes. However, one can use similar approaches to perform synthesis. First, specification properties can be used to generate a classification over the set of designs (e.g., with ‘optimal’ and ‘inferior’ as labels). Concept descriptions in terms of the design properties can then be used to characterize subsets of the training data that were distinguished by their specification properties. For example, we might distinguish design classes using a system like ID3 (see Fisher & Pazzani, Chapter 1, this volume), using only specification properties along arcs of the resultant decision tree. At each leaf of the tree, we would then associate a pattern, much like in EPAM (see Richman, this volume), that characterizes the common design properties of artifacts classified to that leaf. This process captures a *many-to-one* mapping between designs and classes of specifications. Figure 3(b) shows how the classification could be used for a very restricted type of synthesis. In this process, new specifications are classified (e.g., via a decision tree) into a subset of designs (e.g., leaves of a decision tree). The pattern associated with this subset is forwarded as the synthesized design. Unfortunately, it appears that this strategy may yield patterns that are too general for practical purposes.

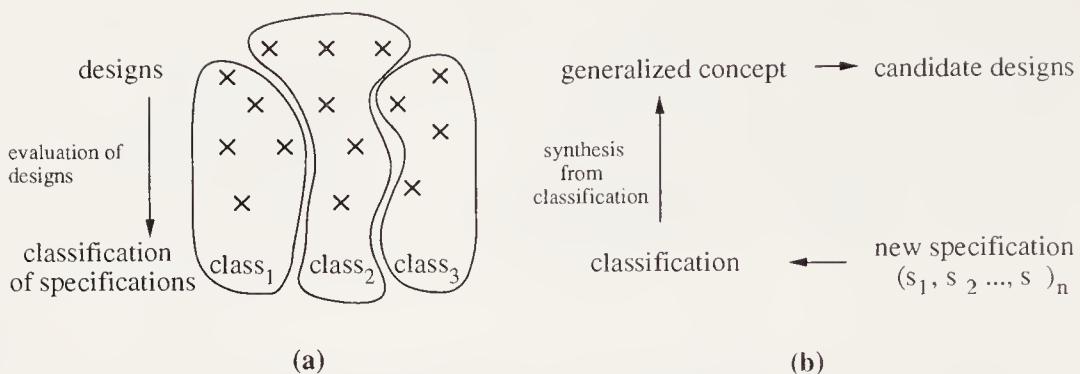


Figure 3. Supervised concept learning and synthesis.

A different approach to synthesis using supervised concept learning techniques is to discover rule sets that directly map specification properties to a single design property (Quinlan, personal communication, 1989). In this case, each design is described by a list ($s_1 = sv_{11}, \dots, s_m = sv_{m1}, d_1 = dv_{11}, \dots, d_n = dv_{n1}$), where d_i denotes a design descriptor, s_j denotes a specification property, and sv_{ik} or dv_{ik} denotes a value of s_i or d_i , respectively. Each design is used to generate n training examples, $(s_1 = sv_{11}, \dots, s_m = sv_{m1}, d_1 = dv_{11}) \dots (s_1 = sv_{11}, \dots, s_m = sv_{m1}, d_n = dv_{n1})$. Over many designs this gives n training sets that generate n decision trees, one for each d_i (see Figure 4).

Although some benefits are possible from this procedure, it may lose relationships that exist between the different design descriptors. Designs generated by this approach might all be inadequate, thus requiring redesign. To illustrate, consider a list of designs that include

1. $(s_1 = sv_{11}, s_2 = sv_{21}, s_3 = sv_{31}, d_1 = dv_{11}, d_2 = dv_{21}, d_3 = dv_{31})$
 2. $(s_1 = sv_{12}, s_2 = sv_{21}, s_3 = sv_{32}, d_1 = dv_{12}, d_2 = dv_{22}, d_3 = dv_{32})$
 3. $(s_1 = sv_{12}, s_2 = sv_{22}, s_3 = sv_{31}, d_1 = dv_{12}, d_2 = dv_{21}, d_3 = dv_{31}),$

which is used to generate three distinct mappings from the specification properties, s_j , to each design property, d_i . If we look at each design property separately, the following predictions for the three objects above might be possible:

1. $d_1 = dv_{11}, \quad d_2 = dv_{21}, \quad d_3 = dv_{32} \neq dv_{31},$
 2. $d_1 = dv_{12}, \quad d_2 = dv_{21} \neq dv_{22}, \quad d_3 = dv_{32},$
 3. $d_1 = dv_{11} \neq dv_{12}, \quad d_2 = dv_{21}, \quad d_3 = dv_{31}.$

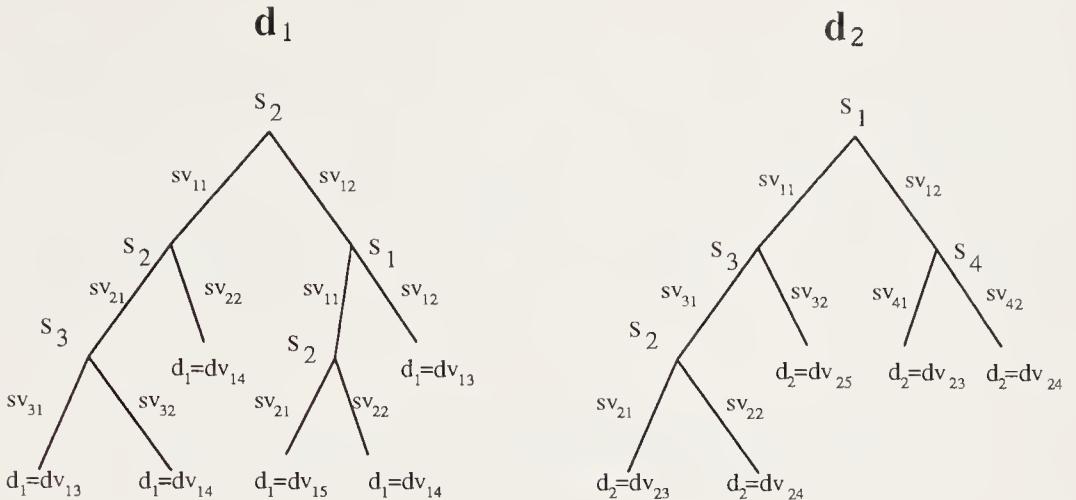


Figure 4. Multiple supervised concept learning and synthesis.

Although the accuracy level is 66% (i.e., six out of the nine design descriptions are correctly predicted), not one of the designs is completely predicted. If there are important dependencies between design descriptors, then evaluation at a later stage may reveal that one or more of these synthesized (i.e., predicted) designs are infeasible.

To generalize our discussion, the inadequacy of supervised concept learning techniques as a means of capturing synthesis knowledge is due to the nature of synthesis. This process involves a *many-to-many* mapping from specifications to design properties. If this mapping is divided into separate *many-to-one* mappings, as required by supervised concept learning, considerable information may be lost, since design descriptor properties are not independent. Furthermore, synthesis cannot be modeled by a mapping from specifications to classes of designs, since such classes are often not known *a priori*.

3.2 Unsupervised Concept Learning

Learning paradigms that are concerned with many-to-many mappings are unsupervised. Figure 5 illustrates an approach to acquiring synthesis knowledge by unsupervised learning of hierarchical (and possibly overlapping) concepts. The principal idea is that specifications and solutions (i.e., design descriptions) are correlated; specific combinations of specification properties give rise to corresponding combinations of design description properties that satisfy these specifications. A clustering based on this correspondence allows the retrieval of appropriate designs

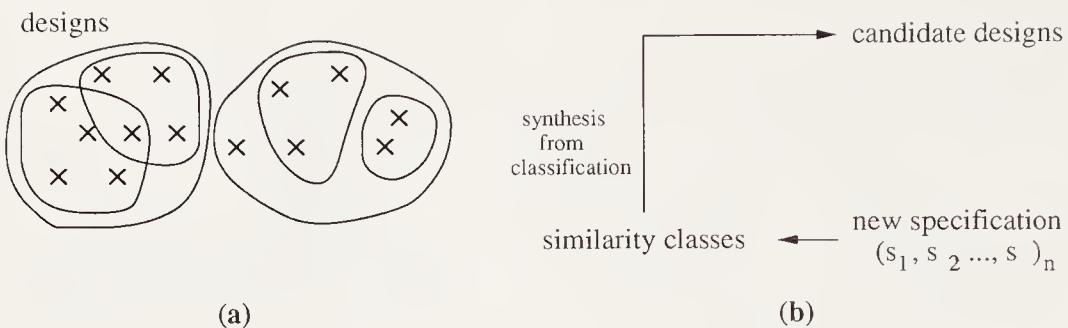


Figure 5. Unsupervised concept learning and synthesis.

given a new specification that is similar to an existing one. Ideally, one would form concepts only from acceptable designs, eliminating the need to redesign in the future. In reality, the unsupervised concept learning process involves an inductive leap from the data and cannot guarantee the generation of perfect designs for a new specification. Thus, analysis and redesign stages cannot be eliminated, but ideally the effort expended during these later stages can be reduced.

There is another dimension to synthesis that constrains the type of unsupervised concept learning method that may be used: the output of synthesis is rarely one alternative; usually, several candidates are produced. An unsupervised concept learning method should use a knowledge representation that supports generation of multiple alternatives.

There are two distinct styles of unsupervised learning: batch (or non-incremental) methods, and concept formation (or incremental) strategies. Research on unsupervised concept learning in design has experimented with both approaches. Although batch processing is not, in general, an ideal approach in design, it can provide some insights into the first stages of understanding a design domain.

A simple example of using nonincremental clustering is the creation of a metric space (Taura, Kubo, & Yoshikawa, 1989). In this approach, a metric (i.e., distance measure) is used to represent the distance between the function or behavior manifested by designs. The function or behavior is extracted from the artifacts' descriptions by using domain knowledge. A similarity function is used to measure the distance between any two artifacts, which are arranged in a similarity matrix, as shown in Figure 6. Several important dimensions $\lambda_1, \dots, \lambda_l$ are extracted by performing a principal coordinate analysis over the matrix.

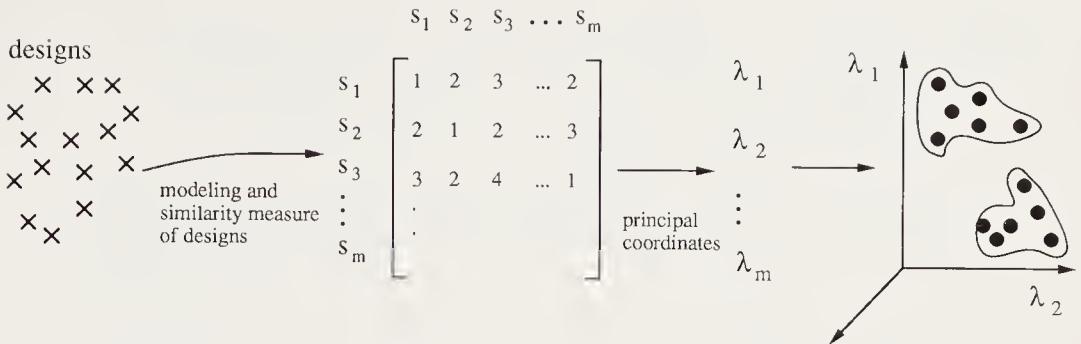


Figure 6. Creation of a metric space.

The analysis reduces the dimensionality of the description space, which in turn allows a simple clustering. Each cluster is characterized by generalizing the design descriptions of its elements. This description is then used in novel situations to construct new objects to satisfy a given functionality expressed by the principal coordinates. Two drawbacks of this approach are that it involves batch processing and it generates a single-layer clustering, which may be computationally inefficient for the retrieval of designs that are similar to the new situation. However, the principles of this approach are similar to those used in our work.

Lu and Chen (1987) provide a more elaborate form of treating multiple specification properties than those described in the previous section. Design examples are characterized by a list of specification and design descriptors ($d_1 = dv_{11}, \dots, d_n = dv_{n1}, s_1 = sv_{11}, \dots, s_m = sv_{m1}$). The specifications are clustered into a finite set of classes by Michalski and Stepp's (1983) conceptual clustering program CLUSTER/2, as shown in Figure 7. These clusters serve as classes that can be used by a supervised concept learning program to generate rules. The products of supervised learning by the program AQ15 (Michalski et al., 1985) are rules that assign the description of a design to the right class, i.e., a specification approximately satisfied by the design. Like other approaches we have reviewed, this strategy is not incremental. Moreover, it captures evaluation knowledge, as illustrated in Figure 3, not synthesis knowledge.

In conclusion, unsupervised concept learning is more suitable than supervised concept learning for the acquisition of design knowledge. This section reviewed two approaches that employed batch learning, which in general is not optimal for design. The next section describes another approach that incorporates ideas from concept formation into design.

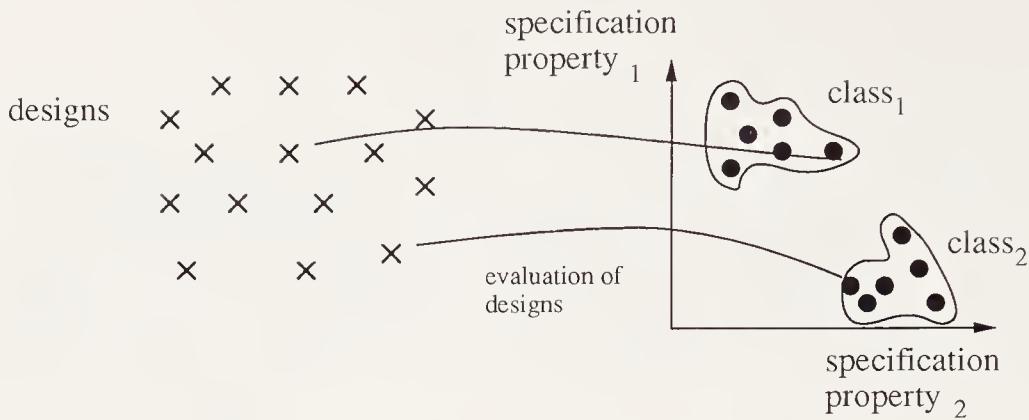


Figure 7. Clustering and supervised concept learning.

4. An Approach to Concept Formation in Design

This section reviews the COBWEB system, considers its application to the generation and use of hierarchical knowledge structures for design, and identifies several of its drawbacks. The section then describes BRIDGER, a descendent of COBWEB that addresses these issues.

4.1 COBWEB: An Incremental Concept Formation System

COBWEB is an incremental concept formation program for the creation of hierarchical classification trees (Fisher, 1987). In the context of design, COBWEB accepts a stream of designs described by a list of property-value pairs. Designs need not be classified as feasible, optimal, or by any other classification scheme. However, any *a priori* classification can be assigned to a design and treated as any other property (Anderson & Matessa, this volume).

A classification is 'good' if the description of a design can be guessed with high accuracy, given that it belongs to a specific class. COBWEB makes use of a statistical function that partitions a set of designs into mutually exclusive classes C_1, C_2, \dots, C_n . The function used by COBWEB, *category utility* (Gluck & Corter, 1985), can be stated as

$$\frac{\sum_{k=1}^n P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2}{n} \quad (1)$$

where C_k is a class, $A_i = V_{ij}$ is a property-value pair, $P(x)$ is the probability of x , and n is the number of classes. The first term in the numerator measures the expected number of property-value pairs that can be guessed correctly by using the classification. The second

term measures the same quantity *without* using the classes. Thus, the category utility measures the *increase* of property-value pairs that can be guessed *above* the guess based on frequency alone. The measurement is normalized with respect to the number of classes.

When a new design is introduced, COBWEB tries to accommodate it into an existing hierarchy starting at the root. The system carries out one of the following operators:

1. if the root has no subclasses, creating a new class and attaching the root and the new design as its subclasses;
2. attaching the design as a new subclass of the root;
3. incorporating the design into one of the subclasses of the root;
4. merging the two best subclasses and incorporating the new design into the merged subclass; or
5. splitting the best subclass and again considering all the alternatives.

If the design is assimilated into an existing subclass, the process recurses with this class as the root of the hierarchy. COBWEB again uses category utility to determine the next operator to apply.

The system generates designs using a mechanism similar to the one used for augmenting the hierarchy with new designs but allowing only the third operator to apply. COBWEB sorts the new specification through the hierarchy to find the best host in memory, a leaf node that describes a known design. The system uses the stored design descriptions to complete the description of the new design. This is a very conservative strategy, as it allows only the exact duplication of existing designs.

To illustrate, let us examine how COBWEB designs a new bridge using a hierarchy generated from 60 examples of bridges from the Pittsburgh bridge database. In this domain, a bridge is described using 12 properties, of which seven are specification properties (three continuous¹ and four nominal): the river and location of the bridge, the period in which it was constructed, the purpose of the bridge, the number of lanes, the length of the crossing, and whether a vertical clearance constraint was enforced in the design. Five design properties are provided for each design: the material used, the bridge span, the bridge type, the roadway location on the bridge (T-or-D, meaning through or deck), and the relative length of the span to the crossing length (Rel-L).

1. We discretized these properties for COBWEB; later, we relax this restriction.

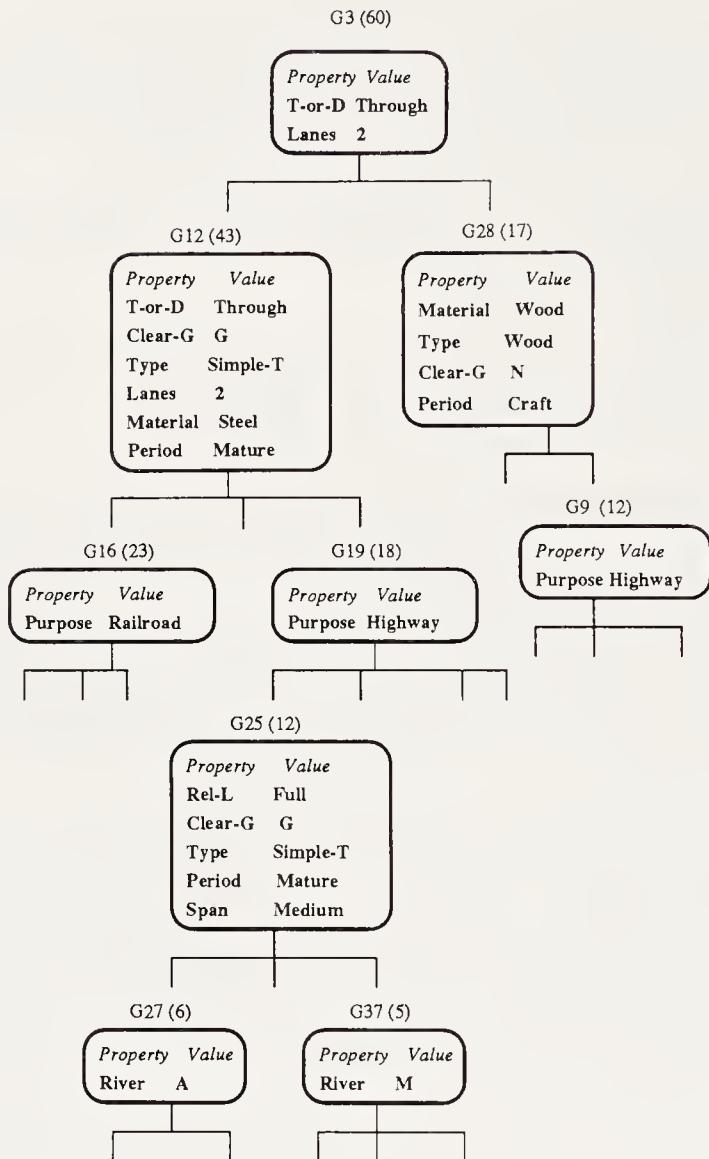


Figure 8. A COBWEB hierarchy of design concepts after 60 training examples.

Figure 8 shows the hierarchy generated by COBWEB after observing 60 examples, ordered by the date of their construction. The nodes are described using only their *characteristic properties*, which are property values whose conditional probabilities exceed a predetermined threshold.² The figure also shows the reference symbol and the number of examples

2. Potentially, this value can be learned for each domain, but in our experiments it has been fixed at 0.75. Other definitions of characteristic or normative values appear in Fisher (1987), Kolodner (1983), and Fisher and Pazzani (Chapter 1, this volume).

below each node (in parentheses). Assume that COBWEB is required to design a highway bridge on one of the rivers where vertical clearance governs. In this situation, the system first classifies the known specification and design constraints by sorting them down the hierarchy. Since vertical clearance governs, the design description is refined using class G12, characterized by this property. Class G19 is chosen next, since it represents highway bridges. At this stage we would like the design process to terminate and return candidate designs (e.g., descendants of G19) since all the known specifications have been met. However, the leaf prediction method that we described always finds a single leaf node; thus, the design process continues.

COBWEB partially supports the requirements presented in Section 2.2. It allows a flexible use of the hierarchy in that any type of specification, partial or complete, can be used to retrieve an appropriate design. However, the system has several drawbacks.

1. COBWEB has a stiff prediction/design scheme, making a strong commitment to continue the design process until it has retrieved a complete existing design. No generation of new designs or accommodation of subjective judgment is allowed. In addition, leaf prediction is not adequate since it produces only one candidate for each specification. This strategy is also susceptible to noise and may result in unnecessarily high error rates.
2. The system can only manipulate nominal properties. Gennari, Langley, and Fisher (1989) describe CLASSIT, a descendent of COBWEB that handles only continuous properties, and McKusick and Thompson (1990) have developed COBWEB/3, which handles both types of properties. Elsewhere, we contrast their approach with the extension to continuous properties implemented in BRIDGER (Reich, 1990c), concluding that our strategy is more 'natural' and flexible.
3. COBWEB uses only the syntactic measure of category utility to guide its learning and prediction. No domain dependent or independent knowledge is used, although such knowledge could enhance learning substantially if it were available.
4. Finally, predictive accuracy is dependent upon the order of example presentation. This dependency manifests itself even though two of COBWEB's operators are designed to reduce order effects.

We now turn to an extension of the basic classification strategy that is intended to mitigate the problems that we have listed above, thus yielding a more robust design tool.

4.2 BRIDGER: An Extension of COBWEB for Design

BRIDGER is designed to extend the applicability of the COBWEB approach to design (Reich, 1990b; Reich, 1990c; Reich & Fenves, in press). Although the system implements other design tasks such as analysis and redesign (Reich, in press), this chapter concentrates on synthesis.

4.2.1 PREDICTION SCHEMES

BRIDGER can use several prediction/design methods. We illustrate our system's prediction strategy using Figure 8 again.³ In the same design problem as before, the system starts by sorting the specification down the hierarchy. The top node of BRIDGER's knowledge represents through bridges with 2 lanes. These property values are assigned to the new design: the first property is a design refinement and the second property is a specification elaboration. Since vertical clearance governs the design, the description is again refined using class G12. The new design will be a simple-truss made of steel. Class G19 is chosen next, since it matches the given highway specification. At this stage the design terminates, since all the specifications have been met, but the design is only partially specified, since the main span and the relative length of the bridge have not been determined. This is expected, since the specifications are abstract. BRIDGER can use refinement strategies to complete the design or retain the abstract design as the solution (Reich, 1990b). For example, the system can deliver the set of 18 bridges under class G19 as design candidates in a pure case-based approach; it can generate a new prototype from the most frequent property-value pairs in G19 and deliver it with a set of possible variations as in a prototype-based design; or it can deliver a large set of candidates generated from the combination of all the property-value pairs of the 18 designs. In summary, the path traversed during classification progressively matches known specifications, and augments the design that is intended to support these specifications.

3. The hierarchy generated by BRIDGER while using continuous properties is slightly different.

4.2.2 CONTINUOUS PROPERTIES AND CONSTRUCTIVE INDUCTION

BRIDGER implements an extension of category utility that can handle continuous properties. In its simplest variant, the term $\sum_j P(A_i = V_{ij}|C_k)^2$ is calculated as⁴

$$\sum_j P(A_i = V_{ij}|C_k)^2 \Leftrightarrow P(A_i = \bar{V}_i|C_k)^2 \Leftrightarrow \left(\int_{-\frac{d_i}{\sigma_i}}^{+\frac{d_i}{\sigma_i}} p_i(x) dx \right)^2 \quad (2)$$

where

$$p_i(x) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-(x - \bar{V}_i)^2 / 2\sigma_i^2} ,$$

$2d_i$ is the ratio of the expected range of property values of A_i and the expected number of distinct intervals of property A_i , \bar{V}_i is the mean of the values of A_i in C_k , and σ_i is the standard deviation of the values of A_i . This approach has been tested extensively in several domains and has proven to be effective (Reich, 1989; Reich, 1990c).

A second extension is based on identifying a mechanism that groups property values into higher-order features. In this extension, continuous property values are grouped into samples from an assumed distribution, such as the normal distribution used above. In the same way, nominal properties can be grouped to support a type of constructive induction ability (Reich, 1990a; Thompson & Langley, this volume). For example, two complementing values V_{11} and V_{12} can be combined into a new feature $G_1 = \{V_{11}, V_{12}\}$. This can be viewed as adding an internal disjunct to the description language. Additional values can be accommodated to or deleted from G_1 based on additional information.

The modified form of Equation 1 that BRIDGER uses to work with such features can be stated as

$$\frac{\sum_{k=1}^n P(C_k) \sum_i \sum_j P(A_i = G_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = G_{ij})^2}{n} , \quad (3)$$

where $G_{ij} = \{V_{ijl} | V_{ijl} \in Dom(A_i)\}$ are new features generated by the constructive induction such that: $\bigcup_j G_{ij} = Dom(A_i)$, $\bigcap_j G_{ij} = \emptyset$, and $Dom(A_i)$ is the domain of values of A_i . In addition, $P(A_i = G_{ij}) = \sum_l P(A_i = V_{ijl})$, where $V_{ijl} \in G_{ij}$.

4. A detailed development of this extension appears in Reich (1990c).

Features are initialized to individual property values. Constructive induction operates as an integral part of calculating category utility, trying to modify the features each time category utility is calculated. The combination of two values occurs only if it increases category utility. In principle, the best approach is to try all the combinations of values each time category utility is calculated. However, this process is exponential in the number of values. Another approach uses the new example as a trigger for the constructive induction scheme and only allows for small changes in the current features. The system considers three basic actions for the features of each property in the new example description:

1. the features are left unchanged;
2. the feature that contains the property value of the new example is merged with another feature; or
3. the feature containing the property value of the new example is divided into either two features, one containing the new property value and the other the remaining values, or into features containing one value each.

Focusing constructive induction on the features of the new example limits the cost of constructive induction, and thus is in keeping with our incremental approach.

4.2.3 SUMMARY

BRIDGER builds on earlier concept formation systems like COBWEB and UNIMEM (in its use of characteristic values), notably in its exploitation of constructive induction and continuous values. We have experimented with the system in a number of standard domains, such as the breast cancer domain and the mushrooms domain (Reich, 1990a).⁵ In these cases, BRIDGER performs comparably to, and in some cases better than, a number of well-known supervised and unsupervised systems. However, the primary goal of this chapter is to investigate the use of concept formation and design. Thus, we will only detail our evaluation of BRIDGER in design domains.

5. Both domains are available from the repository of machine learning databases at the University of California, Irvine.

5. Evaluation of BRIDGER

BRIDGER's approach to design is similar to the General Design Theory (Yoshikawa, 1981); designs are represented by property-value pairs, where properties describe an artifact. This approach is amenable to preliminary or conceptual design, where major design decisions are made without good quantitative measures of their tradeoffs. In particular, this framework is appropriate to our target domain of bridge design. This approach is different from common schemes for VLSI design, such as top-down refinement plus constraint satisfaction (Steinberg, 1988), or similar problems in mechanical design (Ramachandran, Shah, & Langrana, 1988), where substantial domain knowledge is available.

5.1 Synthesis in Bridge Design

BRIDGER's performance was evaluated using the database of Pittsburgh bridge descriptions that we have been using throughout the chapter for illustrative purposes. Figure 9 shows a typical⁶ graph of BRIDGER's behavior on the Pittsburgh bridge database. Figure 9(a) reports the performance in a *coverage* experiment, in which the test cases include *all* the examples in the data set. The results show correct prediction accuracy averaged over the five design properties and 50 runs of randomly ordered examples. The achievement of a 78% average accuracy suggests that the example set is not sufficient to 'cover' the domain. Figure 9(b) shows the performance in a transfer experiment, in which the test examples are only the *unseen* designs. Again the results are averaged over 50 runs, showing gradual, but slow, improvement in transfer of knowledge about individual design properties.

Since the database of bridges does not have many examples, the hierarchy generated is not deep enough to allow the emergence of characteristic values for the specification properties. This impedes prediction at an internal node, which we have already noted is a preferred strategy. In fact, all the predictions of this study are based on leaf nodes, and we suspect that this prevents BRIDGER from producing better results. In a real domain, examples would be generated and evaluated continuously and should let BRIDGER better exploit prediction at internal nodes.

6. There are several parameters that can be adjusted in the prediction method. For example, one can change the threshold for specifying characteristic values and the parameters used for continuous properties. Hence we use the term 'typical'.

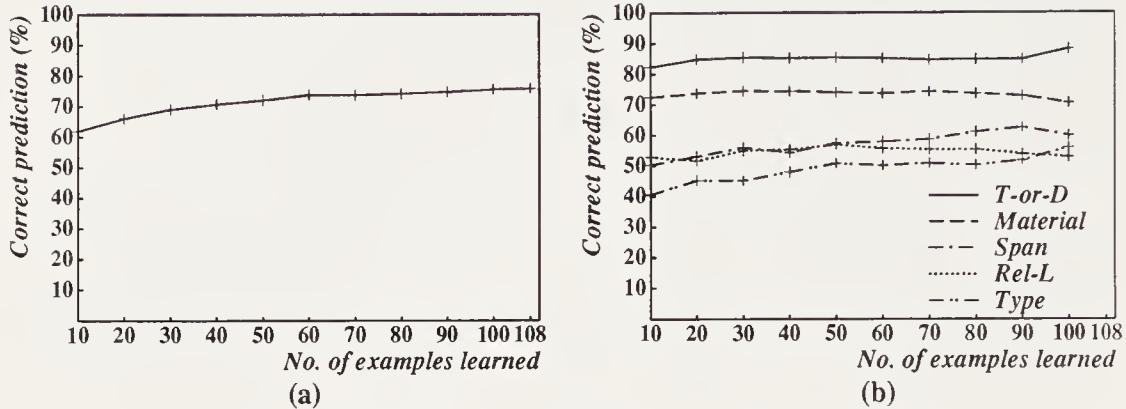


Figure 9. Design performance of BRIDGER on the bridge domain.

We should also note that the results of the individual 50 runs vary considerably. Table 1 shows the best accuracy, the worst accuracy, and the standard deviation of the predictive accuracy of the 50 runs summarized in Figure 9(b). The values in the table are based on the prediction of eight test cases after learning from 100 training examples. The variability of the results suggests that BRIDGER's performance is dependent on the order of example presentation to a larger extent than expected, even using the split and merge operators. A better understanding of order effects is an open research issue. We will turn to possible resolutions of this and other issues involving prediction shortly.

Thus far, we have examined BRIDGER's use for synthesis, which is our primary goal. However, it is also worth noting that the system can be used for evaluation or analysis. Recall that this is a process of mapping design descriptors to specification or behavioral properties. We can intuitively illustrate an evaluation-like process in an imported-automobile database compiled by Schlimmer from Ward (1985).

5.2 ‘Evaluation’ in an Automobile Database

The automobile domain contains 206 examples described by nominal, continuous, and ordinal properties. The task is to predict the price of a car (a continuous property) from descriptors such as the engine type and the body style. In BRIDGER, ordinal properties are treated

Table 1. Variability of results on predictive accuracy.

	T-OR-D	MATERIAL	SPAN	REL-L	TYPE
BEST ACCURACY	1.000	1.000	1.000	0.875	0.875
WORST ACCURACY	0.333	0.500	0.142	0.250	0.125
STANDARD DEVIATION	0.177	0.121	0.219	0.152	0.172

as continuous with the addition of rounding to the nearest integer. For this study we employed a 5×10 -CV experiment.⁷

The experiment used only the 160 examples that are completely specified. One condition was run with only the continuous properties so that comparisons could be made against CLASSIT, which only handles such properties, and against IBP, a supervised instance-based learning approach (Kibler, Aha, & Albert, 1989), which has been used previously in this domain. In addition, a second condition included complete descriptions, for comparisons with BRIDGER's performance using continuous properties only. Table 2 shows the ability of BRIDGER to predict the price of a car from the remaining properties compared to the performance of CLASSIT* (our own implementation of CLASSIT) and IBP. These figures represent the average percentage error from the true cost of each test car over the five experimental runs. This task also illustrates BRIDGER's (and CLASSIT*'s) ability to predict continuous properties, a novel idea for concept formation programs. Furthermore, it shows that BRIDGER outperforms CLASSIT*, which was developed to deal with continuous properties, and that BRIDGER slightly improves its performance when it uses the complete example descriptions. IBP performs best, but since it is a supervised program, it can only use its current knowledge to predict the price of a car, which is the class property. We might loosely interpret this as a specification property (of a buyer) that BRIDGER predicts from 'design' characteristics. In general, BRIDGER's construction of a many-to-many mapping between specifications and designs provides a flexible knowledge base that supports synthesis and analysis in a uniform manner.

7. An $n \times V$ -CV experiment is a *V-fold cross validation* experiment (Breiman, Friedman, Olshen, & Stone, 1984), which is performed over n random orderings of the examples and the results are the average of the n runs.

Table 2. Predictive accuracy of three systems on the automobile database.

IBP	CLASSIT*	BRIDGER	
		NUMERIC ONLY	ALL PROPERTIES
11.84	16.10	14.74	14.31

5.3 Behavior on the Bridge Analysis Database

Another experiment examined BRIDGER's ability to predict multiple continuous values from a single hierarchy. The bridge analysis database contained 1000 examples of bridges, each described by 26 properties. The bridge examples were generated by randomly assigning values to the 26 properties. The bridges were then analyzed by an analysis program that produced 12 continuous behavioral properties that were consistent with the object's corresponding design properties. Thus, each full description contained 38 properties. These data were used to train BRIDGER and test its ability to predict the analysis results without performing the exact analysis.

We produced a learning curve by training BRIDGER incrementally and, after each 100 examples, testing the system on the remaining examples. In testing, each prediction of the 12 analysis results made by BRIDGER was checked against the actual analysis calculated by the analysis program mentioned above.

Results were averaged over five runs with different random orderings of the examples. The results indicate that analysis prediction continuously improves from a 1000% average error rate after 100 examples to a 30% average error rate after 400 examples. The prediction of four out of the 12 analysis results converges to within 8% from the exact values after 400 examples. One property remains far from the exact analysis (200% mean relative error), but this is caused by a small number of major errors.

The studies presented demonstrate BRIDGER's performance on a variety of tasks. As we noted, studies in other domains yield good results relative to other supervised and unsupervised systems, and better illustrate BRIDGER's constructive induction abilities. To some extent,

though, the results in the bridge domain of Section 5.1 indicate additional directions for improvement. We briefly discuss two of these directions below.

5.4 Additional BRIDGER Learning Mechanisms

Experimental results with the bridge database indicated that BRIDGER may require considerable data before certain of its prediction-enhancing mechanisms can be exploited, notably the use of characteristic values and intermediate node prediction. To some extent this is a limitation of many strictly inductive systems. Additionally, the system demonstrated sensitivity to training order. We have made some initial progress on the problem of exploiting knowledge and mitigating ordering effects during learning.

5.4.1 KNOWLEDGE-DRIVEN LEARNING AND PREDICTION

An important characteristic of BRIDGER's classification scheme is that it is declarative. This enables an external process, domain dependent or independent, to inspect it. The ability to benefit from external knowledge relies on the flexible nature of our learning method, which uses weak search methods directed by the category utility function category utility. However, our eventual goal is to use category utility as a default mechanism in the absence of knowledge, though explicit knowledge can be used to prefer a specific operator (e.g., merging and splitting) and override category utility preferences.

Another way of using knowledge is to construct examples that will enhance the performance of the system (Scott & Markvitch, this volume). Work is under way to construct such experimentation mechanisms. A final kind of domain knowledge that we are investigating is knowledge about modeling. For example, such knowledge can suggest important features of objects. Given an initial description of objects, the knowledge can augment it by additional properties, compress the description of several properties, or erase irrelevant properties. To some extent these abilities are similar to those proposed by Stepp and Michalski (1986) in their CLUSTER/S system (also see Fisher & Pazzani, Chapter 6, this volume). Such knowledge can be augmented by a constructive induction mechanism that generates higher-level features from observable properties.

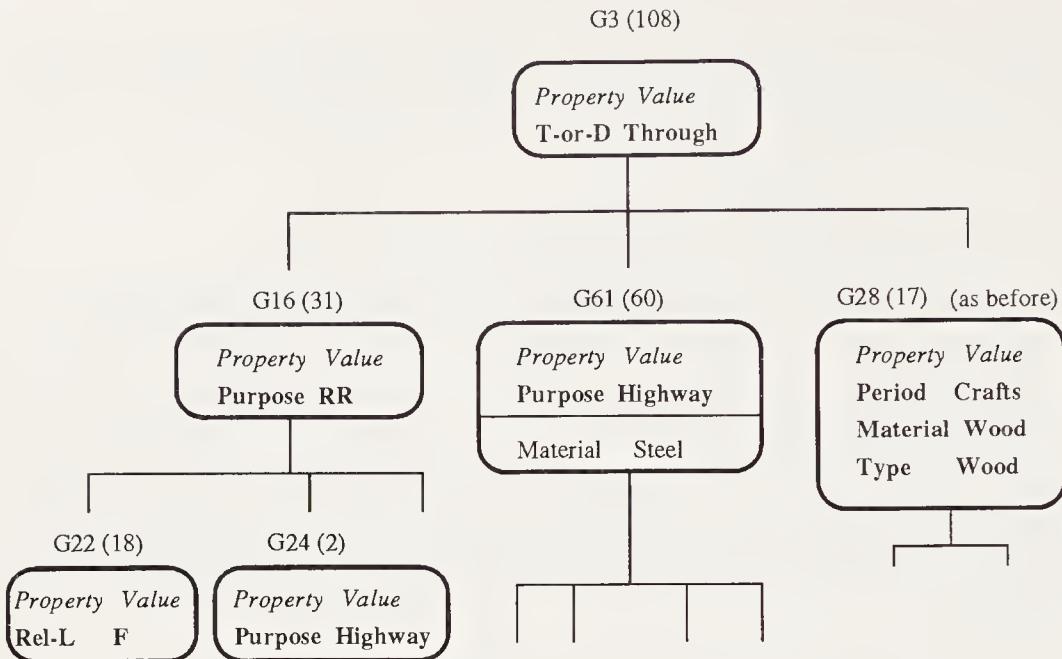


Figure 10. Top hierarchy after learning 108 examples.

5.4.2 MITIGATING ORDER EFFECTS

Large variance in the results of the bridge domain indicates BRIDGER's sensitivity to the order of training examples. Our approach to mitigating this problem involves using knowledge about specifications to reorganize the hierarchical knowledge structure in a process called 'hierarchy correction'. Figure 10 depicts the hierarchy generated from 108 bridge examples. Class G24, which is characterized as a highway class, is placed below a large railroad class. In this case, it might be beneficial to move this class to a more appropriate branch of the hierarchy.

The hierarchy-correction scheme follows three steps. First, properties that are deemed most critical by a domain expert are manually selected as 'triggers'. Second, the hierarchy is traversed in a top-down manner. Each class with a characteristic property value that differs from a characteristic in one of the class' ancestors is removed from the hierarchy, along with its subtree. Third, the examples at the leaves of all the removed subtrees are reclassified into the hierarchy. The process can iterate several times until no change of the hierarchy is obtained. The application of this procedure enhances the predictive accuracy of BRIDGER, although it loses some of the algorithm's incremental flavor.

6. A Framework for Design and Concept Formation

One of the most important aspects of a concept formation or a design system is the *knowledge* provided and its interaction with the system's *control*. The behavior of the system is determined by the controlled application of its knowledge on the input received from the environment. The less knowledge a system has, the more it must rely on exploratory and exhaustive search, whereas the more knowledge it has, the more it can be conservative and fixed.⁸

Figure 11 shows several concept formation and conceptual clustering programs classified along the dimensions of exploration and knowledge (without being precise about the relative location of the systems). Let us elaborate on the classification of some of these systems.

- COBWEB has a fixed control strategy based on the category utility function and rudimentary knowledge about the collection of properties describing objects in a domain. Its control strategy is highly restricted, since it can only make one of a few local changes (e.g., through merging and splitting) to its hierarchy at any one time.
- CLUSTER/S (Stepp & Michalski, 1986) employs domain knowledge and a goal dependency network to identify the most 'relevant' properties in the formation of clusters. Although CLUSTER/S employs a more extensive search control strategy than COBWEB, it is not amenable to incremental refinement.
- GENESIS (Mooney, this volume) uses domain knowledge with explanation-based mechanisms to form concepts, which may be unavailable for some types of design. Its control strategy is fixed.
- EXOR (Yoo & Fisher, this volume) uses a domain theory to build explanations and then clusters over these explanations. The hierarchical clustering guides the construction of new explanations, much as BRIDGER uses the hierarchy to guide the construction of designs.
- As a clustering system, BRIDGER extends COBWEB in that it can use knowledge in its hierarchy-correction scheme, and can invoke additional processes such as constructive induction. Thus, it increases COBWEB's abilities in both the knowledge and control directions.

8. This view is similar to Newell's (1989) tradeoff between knowledge and search, which states that the more domain knowledge that is available, the less search deliberation is needed to solve a problem.

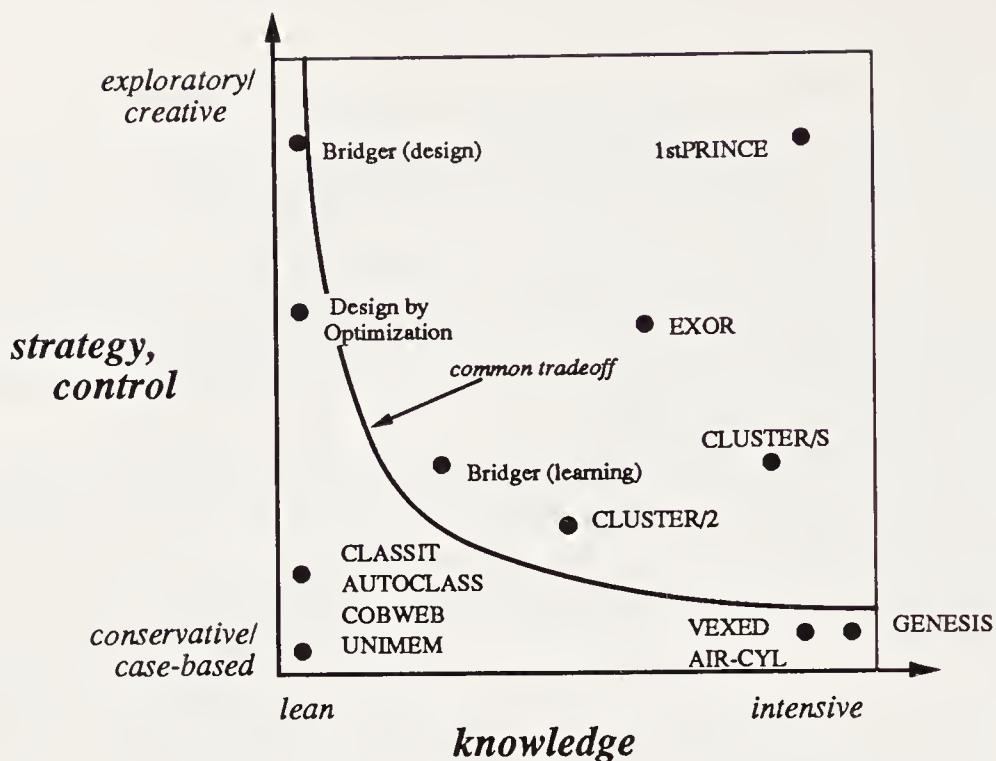


Figure 11. A framework for concept formation and design systems.

We can also classify design systems within this framework. Usually, the systems' desired behavior (e.g., routine design versus innovative design) determines the knowledge and control required.

- 1stPRINCE uses qualitative knowledge to reason from first principles (Cagan & Agogino, 1987). Its control is guided by optimization techniques that are exploratory in nature. These two ingredients facilitate innovative, albeit simple, design behavior.
- AIR-CYL has stiff control composed of fixed design plans, and limited knowledge representation schemes composed of well-understood design alternatives. Considerable compiled knowledge is necessary to formulate these ingredients, and they can only support a certain class of routine design (Brown & Chandrasekaran, 1986).
- As a design system, BRIDGER can use certain exploratory and redesign procedures that allow considerable flexibility during design. Section 4.2.1 discussed some of this flexibility when describing the various methods for synthesis. This contrasts with BRIDGER's learning mechanism, which always generates a new leaf node from

a new example. Therefore, as a design system, it falls high along the exploratory axis. BRIDGER's design procedures are purely syntactic in that domain knowledge is exploited only at learning time; therefore, the system falls low along the knowledge axis.

This framework can aid in the selection of learning techniques for a particular design task, but it also shows the gaps in our knowledge about the use of learning in design. The sparseness of examples in the framework suggests that much work remains to be done.

7. Conclusions

Design is a complex task. This chapter concentrated on the synthesis aspect of design: the assembly of candidate designs from an infinite set of alternatives. Synthesis is an ill-structured process with respect to all three characteristics of such problems. The only process that can reduce the complexity of synthesis is the acquisition and reorganization of knowledge. An appropriate knowledge structure can guide the search for alternatives in an efficient manner.

Mapping the requirements of synthesis onto the functionality of machine learning techniques reveals that incremental concept formation is an appropriate means for acquiring design knowledge. In particular, we presented BRIDGER, a concept formation system that partially automates the acquisition of design knowledge for a class of design domains in which the artifact's structure is fixed. There are also aspects of the learning/design mapping that we have not stressed. For example, specification properties roughly correspond to operational predicates exploited by explanation-based systems (Mooney, this volume; Yoo & Fisher, this volume). Ideally, learning should render synthesis more efficient by allowing a design system to construct designs appropriate to the specification (i.e., operational) predicates.

Our experiences with BRIDGER demonstrate that studying concept formation in conjunction with design can benefit the understanding and development of both. Work in design domains has forced the development of several extensions to COBWEB and the results obtained with BRIDGER provide experimental support for a general design theory (Reich, 1990b). The study of the two tasks also provides a better understanding of the reasons why concept formation is appropriate for the

acquisition of task-specific design knowledge. We believe that this understanding is further enhanced by a simple framework for classifying concept formation and design systems. This framework motivates and directs our continuing research.

Acknowledgements

The work described in this chapter was supported in part by the Engineering Design Research Center, a National Science Foundation Engineering Research Center, and by the Sun Company Grant for Engineering Design Research. Doug Fisher, through dedicated editorial work, has provided many helpful comments on drafts of this chapter.

References

- Arciszewski, T., Mustafa, M., & Ziarko, W. (1987). A methodology of design knowledge acquisition for use in learning expert systems. *International Journal of Man-Machine Studies*, 27, 23-32.
- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.
- Brown, D. C., & Chandrasekaran, B. (1986). Knowledge and control for a mechanical design expert system. *IEEE Computer*, 19, 92-100.
- Cagan, J., & Agogino, A. M. (1987). Innovative design of mechanical structures from first principles. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 1, 169-189.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139-172.
- Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11-61.
- Gluck, M., & Corter, J. (1985). Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283-287). Irvine, CA: Lawrence Erlbaum.
- Kibler, D., Aha, D. W., & Albert, M. K. (1989). Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5, 51-57.

- Kolodner, J. L. (1983). Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7, 243-280.
- Lu, S. C., & Chen, K. (1987). A machine learning approach to the automatic synthesis of mechanistic knowledge for engineering decision-making. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 1, 109-118.
- Magued, M. H., Bruneau, M., & Dryburgh, R. B. (1989). Evolution of design standards and recorded failures of guyed towers in Canada. *The Canadian Journal of Civil Engineering*, 16, 725-732.
- McLaughlin, S., & Gero, J. S. (1987). Acquiring expert knowledge from characterized designs. *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, 1, 73-87.
- Michalski, R. S., & Chilausky, R. L. (1980). Knowledge acquisition by encoding expert rules versus computer induction from examples: A case study involving soybean pathology. *International Journal of Man-Machine Studies*, 12, 63-87.
- Michalski, R. S., Mozetic, I., Hong, J., & Lavrac, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 1041-1045). Philadelphia, PA: Morgan Kaufmann.
- Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1, 47-86.
- Newell, A. (1989). Putting it all together: Final comments. In D. Klahr & K. Kotovsky (Eds.), *Complex information processing: The impact of Herbert Simon*. Hillsdale, NJ: Lawrence Erlbaum.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81-106.
- Ramachandran, N., Shah, A., & Langrana, N. A. (1988). Expert system approach in design of mechanical components. *Engineering with Computers*, 4, 185-195.

- Reich, Y. (1989). *Combining nominal and continuous properties in an incremental learning system for design* (Tech. Rep. No. EDRC-12-33-89). Pittsburgh, PA: Carnegie Mellon University, Engineering Design Research Center.
- Reich, Y. (1990a). Constructive induction by incremental concept formation. *Proceedings of the Seventh Israeli Symposium on Artificial Intelligence and Computer Vision* (pp. 195-208). Tel-Aviv, Israel: Elsevier Science Publishers.
- Reich, Y. (1990b). Converging to "ideal" design knowledge by learning. *Proceedings of the First International Workshop on Formal Methods in Engineering Design* (pp. 330-349). Colorado Springs, CO.
- Reich, Y. (1990c). *Incremental clustering with mixed property types*. Unpublished manuscript, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.
- Reich, Y. (in press). Design knowledge acquisition: Task analysis and a partial implementation. *Knowledge Acquisition*.
- Reich, Y., & Fenves, S. J. (1989). The potential of machine learning techniques for expert systems. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 3, 175-193.
- Reich, Y., & Fenves, S. J. (in press). Automatic design knowledge acquisition by concept formation. In T. Arciszewski (Ed.), *Knowledge acquisition in civil engineering*. New York: American Society of Civil Engineers.
- Reitman, W. R. (1964). Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M. W. Shelly & G. L. Bryan (Eds.), *Human judgments and optimality*. New York: John Wiley & Sons.
- Simon, H. A. (1973). The structure of ill-structured problems. *Artificial Intelligence*, 4, 181-201.
- Steels, L., & Van de Velde, W. (1986). Learning in second generation expert systems. In J. S. Kowalik (Ed.), *Knowledge-based problem solving*. Englewood Cliffs, NJ: Prentice-Hall.
- Steinberg, L. I. (1987). Design as refinement plus constraint propagation: The VEXED experience. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 830-835). Seattle, WA: Morgan Kaufmann.

- Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering of structured objects: A goal-oriented approach. *Artificial Intelligence*, 28, 43–69.
- Stone, J., & Blockley, D. (1989). Finding patterns in structural failures by machine learning. *Proceedings of the IABSE Colloquium on Expert Systems in Civil Engineering* (pp. 47–56). Zurich, Switzerland: IABSE.
- Taura, T., Kubo, S., & Yoshikawa, H. (1989). Generation of design solutions by a metric space. *Proceedings of the Third IFIP Workshop on Intelligent Computer-Aided Design*. Osaka, Japan.
- Ward (1985). 1985 model import car and truck specifications. In H. A. Stark (Ed.), *Ward's Automotive Yearbook* (Vol. 47). Detroit, MI: Ward's Communication.
- Yoshikawa, H. (1981). General design theory and a CAD system. *Proceedings of the IFIP Working Conference on Man-Machine Communication* (pp. 35–57). Amsterdam: North-Holland.

CHAPTER 13

Learning to Recognize Movements

WAYNE IBA

JOHN H. GENNARI

1. Introduction

Human motor behavior covers a remarkable range of abilities. A comprehensive theory of human motor learning should explain a wide range of phenomena — from simple tasks such as an infant’s learning to reach for and grasp toys, to a complex task such as learning to play a violin or to throw a knuckle ball. These examples emphasize our interest in how humans acquire and improve motor skills through observation and practice.

Although motor learning is usually thought of as improvements in performance as a result of repetitive practice, an agent must first acquire an initial movement in order to improve it. A learner acquires initial movement representations when it is either generating movements by chance, observing another agent (such as a teacher) perform a particular skill, or problem solving to achieve a particular goal. In this chapter, we consider the case in which the learner observes movements as they are performed by another agent. As a function of multiple observations, a person acquires the ability to “understand” or recognize a new movement as being similar to a set of previously observed movements. This understanding consists of two steps: breaking a stream of sensory information into a sequence of states (parsing the movement), and finding the most appropriate match of the parsed movement with movements that have been previously experienced and stored in memory (classifying the movement).

Acquiring the ability to understand motion involves clustering sets of similar movements that, taken together, correspond to “concepts”. For example, we would think of “throws” as a class of movements involving an arm and an object (say a ball) that are similar in many ways. Furthermore, we could distinguish among types of throws; for pitching a baseball we might have classes for fast balls, curve balls, and sinkers. As the system learns from observing throw movements, its set of classes should adjust to accurately reflect the domain. Over time, this set of concepts should let the learning agent better recognize and classify movements it observes in the environment.

In this chapter we will focus on movement recognition, and show its relationship to the more general task of concept formation. In the next section, we provide context for the current work and define the problem that we specifically address here. Next, we introduce OXBOW, a computer system that embodies some of our ideas about motor learning. To do this, we discuss the system’s representation for movements and concepts, its approach to parsing and classifying movements, and the learning that occurs during movement recognition. Finally, we present an initial evaluation of the OXBOW system, and conclude with a discussion of its relation to a more complete theory of motor control.

2. Research Orientations

In order to provide a context for our research, it is important that we clearly state our approach to artificial intelligence. This has become more important as AI has become more and more diverse, including researchers from biology and engineering, in addition to computer science and psychology. In this section we present our general research goals, including methods for determining success or failure and a precise definition of the problem.

2.1 Goals of the Research

Our long-term goal for research in motor learning is to use known phenomena from psychological studies of motor control to constrain our computational models of learning. As with most work in machine learning, we are interested in creating a computer system that embodies our model, and showing that its ability improves over time. We view this approach as an important middle ground between previous work in

robotics, which is usually not concerned with learning at all, and work in psychology, which usually does not specify a complete computational model to explain the observed phenomena (Iba, 1989).

This approach implies two criteria for success. First, our model should account for the psychological phenomena in motor behavior. Our current work focuses on a particular part of the motor learning process, namely the recognition of movements and the application of concept formation to this task.¹ In this chapter, we simply indicate where our design has been motivated by psychological constraints, leaving direct comparisons between our system and psychological phenomena to future work.

A second measure of success is the ability of the model to demonstrate improved performance over time. More precisely, we must define a measurable *performance task*, and show that as the system is presented with more training instances, its performance improves. In this chapter, we present some initial empirical results demonstrating that our system provides a viable means for representing, acquiring, and generating movements and motor skills. We claim that our model presents an interesting and useful representation and extends previous concept formation systems toward more structured domains.

2.2 Problem Statement

Movement recognition is the process that occurs when an agent observes others performing particular skills. To attach meaning to an observation, it must be *classified* (given a label) and related to previously stored knowledge or experiences. We refer to this performance task as *movement recognition*, and define it as:

- *Given*: an observed movement in the environment;
- *Classify*: the movement according to stored knowledge.²

Classifying a movement means that the system chooses some “movement concept” (a stored description for a class of movements) as most appropriate for the new movement.

1. For a description of the relationship between concept formation and psychological results, see Fisher and Langley (1990) and Anderson and Matessa (this volume).
 2. In order to classify the movement, it must first be parsed into a sequence of states. We will describe this process in more detail in Section 4.

Movement recognition requires that each observed movement be compared to previously stored knowledge. One way to access and update a set of experiences is to cluster them into concepts and arrange these hierarchically. This is a task of unsupervised concept formation:

- *Given*: a sequential presentation of instances and their associated descriptions;
- *Find*: clusterings that group those instances in categories;
- *Find*: a hierarchical organization for those categories.

These two task descriptions define both learning and performance for concept formation. Learning occurs whenever the hierarchy is modified and whenever a cluster is modified or augmented. The learning task is to improve the ability to recognize or classify movements as a result of experience. The performance task is the same as that described above: classifying a new instance as a member of a set of previously stored movements. A general metric for evaluating a system's ability to classify instances is *prediction accuracy* (Fisher, 1987; Gennari, Langley, & Fisher, 1989). For movement recognition, the metric we use compares an observed movement trace to the movement trace stored with the chosen concept for classification; the result of this comparison is an error score over the course of the movement. We expect that the chosen class should become increasingly predictive of the observed movement as the system acquires experience.

One important aspect of concept formation is that it is an *incremental* process. This means that learning occurs with each instance, and that the system does not need to reprocess all previously seen examples in order to learn. This is a fundamental constraint imposed by psychological results: humans observe a never-ending sequence of instances, and they can use their learned knowledge at any point in time.

Given the specification of our performance and learning tasks, we can now present OXBOW, a system designed to form concepts for use in movement recognition. OXBOW is a module in a larger system called MÆANDER (Iba, 1991) that addresses a wide range of learning and performance issues related to recognition and generation of human motor behavior. The methods used here also incorporate many of the ideas from CLASSIT, an earlier concept formation system (Gennari, Langley, & Fisher, 1989).

3. Representation and Data Structures in OXBOW

Any computational model of motor skills requires some representation to operate upon. Likewise, if such a model is to store and retrieve skills, then it must also have a means of organizing their representations in a flexible manner. In this section we introduce OXBOW's format for representing observed movements and its method for organizing these representations.

3.1 Representation of Movements

We assume that movements in OXBOW are generated by a jointed limb and that information about each of the joints is available to the system. This generation may either be observed or performed by the learning agent. In this chapter we focus on observed movements, but our representation is similar for the generation of movements. Furthermore, although the representation described in this section describes movements of only a single limb, the extension to multi-limb movement is straightforward. A *movement* is presented to the system as a sequence of *state descriptions* characterizing the arm at uniform intervals of time. These intervals reflect the granularity of the system's perception of continuous time.

The state of an arm is described by listing the positions, rotations, and velocities for each of the joints at a given time. In addition, the system may optionally specify events that occur at particular times, such as grasping or releasing an object. Such a facility allows motor skills to manipulate the environment in interesting and complex ways. The set of commands that can be specified for a particular state is limited only by the manipulator hardware.

Although we present movements to the system as a complete sequence of state descriptions, OXBOW does not store these representations in a long-term memory. Instead, the information necessary to recall and carry out a movement is stored as a *motor schema*. This is similar in intent to Schmidt's (1982) use of the term. As with our definition of an observed movement, we represent a motor schema as a sequence of state descriptions. However, instead of storing state descriptions for every time slice, a schema specifies the state of the arm at only a few times during a movement. That is, we claim that smooth continuous movements are often adequately described by just a few state descriptions.

The intermediate positions of the arm (between state descriptions) are implicitly specified by an interpolation mechanism. Because a motor schema explicitly represents arm positions at only a few selected points over the course of a movement, we refer to a schema as *sparse* with respect to time.

More formally, we define a motor schema as a sequence of states, (S_1, S_2, \dots, S_n) , where each state

$$S_i = (t_i, \{\langle J_k, \mathbf{p}, \dot{\mathbf{p}} \rangle, \dots\}, C_h)$$

contains a time value t_i , a set of 3-tuples, and an optional hand command, C_h . The states, S_i , are ordered such that the time values, t_i , are in an increasing sequence: $t_i < t_j$ for $i < j$. Each 3-tuple contains: a joint name J_k , which identifies the joint described by the 3-tuple; a position \mathbf{p} , which is the expected position of the specified joint at time t_i ; and a velocity vector $\dot{\mathbf{p}}$, which describes the desired velocity of the joint upon reaching the position \mathbf{p} . Each state contains a set of such 3-tuples, each of which describes one of the effector's joints, although not all joints need be specified.³ The exceptions are the first and last state descriptions in the schema, which must specify a 3-tuple for every joint. We assume that for any pair of unique 3-tuples in this set, $J_k \neq J_l$.

Figure 1 gives a pictorial example of a movement and a schema. The movement in Figure 1(a) shows the position of the arm at equal time slices or snapshots during the course of the movement. Tightly packed arm positions correspond to slow velocities, whereas more loosely spaced positions indicate higher speeds. Note that the movement shows the position of the arm at every time during the movement (with respect to the granularity of the sensory and control system). In contrast, motor schemas specify arm positions only at a few times during the course of a movement. This can be seen in the schema shown in Figure 1(b), which *represents* the movement shown in Figure 1(a) but only *specifies* information for the arm three times. In our framework, movements and schemas are closely related. In Section 4 we will discuss the parsing mechanism that takes a movement and returns a schema based upon that movement.

The representation used here derives its flexibility for both recognition and generation of movements by way of alternate formats used to

3. In this chapter we do not utilize this capability. In general, the information for each of the joints may not be available initially and so we have designed our representation to handle such situations.

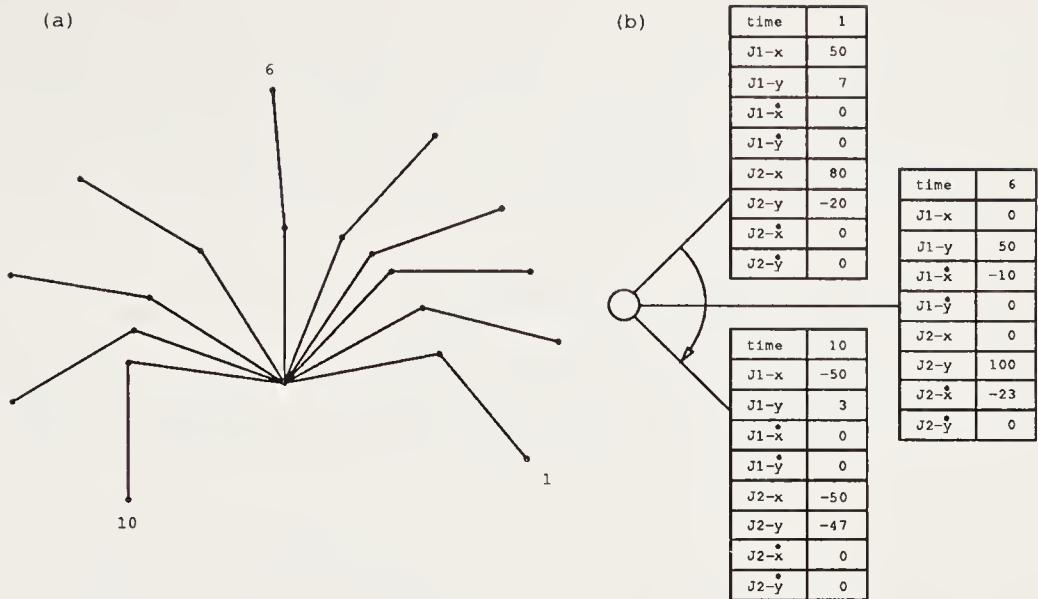


Figure 1. Pictorial representation of (a) a movement and (b) a motor schema.

specify joint information. The positions and velocities of the joints as given in the 3-tuples can be represented in either viewer-centered or joint-centered coordinates. Because these two formats are based upon differing coordinate systems, they give rise to two types of schemas that lend themselves to different performance tasks. In this chapter we will only consider *viewer-centered* schemas, but we return briefly to joint-centered schemas in the closing discussion.

A viewer-centered schema represents the position and velocity vectors using Cartesian two-space coordinates with the origin centered at the agent. For the purposes of this chapter, the center of an agent will always be located at the base of its arm. Therefore, in a viewer-centered schema, the first 3-tuple (describing joint J_0) would specify the x and y coordinates at the end of the first arm segment (actually the location of joint J_1) relative to the origin located at the base (or joint J_0). Similarly, the information stored at each joint J_i reflects the position and velocity of joint J_{i+1} relative to the base at joint J_0 .

The viewer-centered representation gets its name from the source of this information — the agent's visual sensors. These can be thought of as generalized world sensors: anything that lets the agent observe objects and their positions relative to the agent's current location. In the case of a more complete agent, one can imagine other origins for a

viewer-centered schema, such as the agent's eyes. The choice of origin and axes should not affect the behavior if we assume a linear translation from the chosen origin to the base of the effector. This translates any given viewer-centered representation into our canonical viewer-centered representation.

3.2 Probabilistic State Descriptions

When motor schemas are combined to form abstractions or generalizations, one can think of the resulting structures as *concepts*. In order to represent multiple instances with a single item, the values representing a movement must somehow be relaxed. One way to represent concepts in this type of model is to use probabilities (Smith & Medin, 1981). In the previous section we described a motor schema as a sequence of states in which each of the states contained specific values describing the set of joints. Now, let us extend this to include a probability distribution for each attribute describing the state of the arm. Further, let us include a probability for each entire state. That is, a state has a certain probability of appearing in a given schema and, if it does, then the values for its time and joint positions each have associated probability distributions. Our notion of concepts is identical to our original description of motor schemas except that each value is replaced by a mean and a variance. Therefore, any given motor schema can be either a very specific concept with minimal variance (a schema representing a single movement), or a more abstract "movement concept" with higher variance (a schema representing many movements).

In general, concept formation systems may use discrete (nominal or ordinal) attributes or continuous (real-valued) attributes. In this chapter, we will mostly be concerned with continuous, real-valued attributes since we describe the positions and rotations of joints numerically. However, OXBOW allows either nominal or continuous attributes: for example, the hand command is a non-numeric attribute. Whether discrete or continuous attribute values are used to describe the joints, the information can be represented with a probability distribution. The only difference between the two cases is that in the nominal case the probabilities are stored explicitly for each possible value of a given attribute, while in the continuous case, we assume that the observed data can be summarized with a normal distribution (i.e., we store the mean and standard

deviation of the distribution). This is a frequently made assumption in concept formation (Fried & Holyoak, 1984; Cheeseman et al., 1988; Gennari et al., 1989; Anderson & Matessa, this volume).

In the following sections we discuss in detail how these schemas are put to use, focusing on using viewer-centered schemas to observe and recognize another agent's movement. In the discussion section we briefly describe how these schemas, together with joint-centered motor schemas, are used to *generate* movement. Thus, our framework can be used for both recognizing a movement and monitoring the progress of a self-initiated movement.

3.3 Memory Organization

We have introduced a representation for movements that we refer to as the *motor schema*. However, in order to access or retrieve stored schemas, they must be organized in some consistent manner that facilitates efficient access according to some retrieval mechanism and that fares well with respect to representational economy. Here we describe the organization used to store these schemas in long-term memory.

In OXBOW, knowledge about movements is organized into a *hierarchy* of motor schemas. Nodes in this hierarchy are partially ordered according to generality, with concepts lower in the hierarchy being more specific than their ancestors above them. Thus, the root node summarizes all instances that have been observed, terminal nodes correspond to single instances, and intermediate nodes summarize clusters of observations. Fisher and Langley (1990) review arguments for organizing probabilistic concepts in a hierarchy.

Figure 2 shows a possible hierarchy for baseball pitching schemas. This represents the memory of an agent that has experienced a *sidearm* pitch, and three overhand throws — a *fast-ball*, a *curve-ball*, and a *fork-ball*. The leaf nodes of the tree in the figure represent the motor schemas from specific observed pitches. However, instead of simply storing the observed values, these values become the means (with a very small standard deviation) for the most "specific" concepts in the hierarchy. The node labeled *overhand* represents a generalization of the three specific throws stored below it in the hierarchy. This generalization is also a motor schema, but instead of specific values, the generalization contains means and variances for each attribute in its state descriptions. The higher variance makes the representation more abstract than a

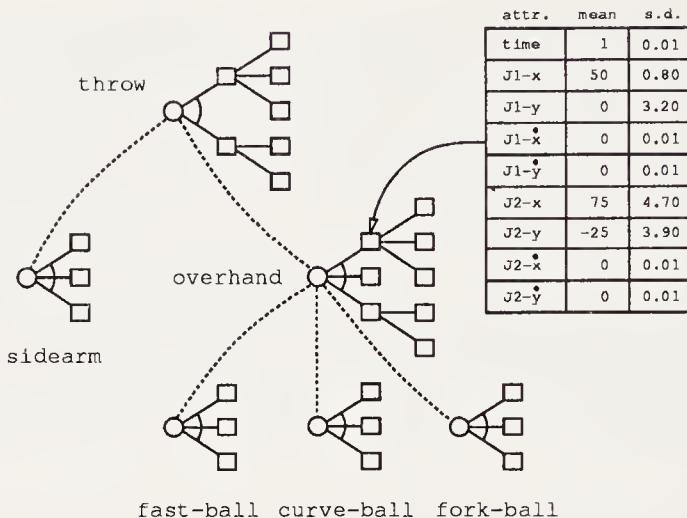


Figure 2. An is-A hierarchy of schemas for “throw” movements, along with a portion of their internal structure.

motor schema resulting from a single observed movement in that more instances will readily match an abstract concept than a specific one.

Recall that our motor schemas consist of a sequence of state descriptions. These descriptions can be thought of as components of the entire schema. It is important to note that this sequential representation implies a partial structure to these concepts. Each state description is “part of” the entire motor schema, although the states themselves are not represented structurally and a strict ordering (with respect to time) exists on the states. This structure is based upon *temporal* relations as opposed to the more traditional *spatial* relations in the context of PART-OF hierarchies. This significantly complicates the concept formation task as it is commonly conceived.⁴ As a further complication, there may be a variable number of states in a given motor schema. In Section 5 we discuss our response to these issues, but for now one need only to understand the structural nature of our representation.

The way OXBOW stores and organizes these state descriptions introduces an additional hierarchy of *state* descriptions. Earlier we said a node in the schema hierarchy represented a movement concept that generalized some set of observed movements. Now let us add that within

4. For one conceptualization of the concept formation problem in the context of structured representations, see Gennari et al. (1989). Thompson and Langley (this volume) present another approach to solving this extended problem.

the node, the state descriptions comprising the motor schema are organized into a separate IS-A hierarchy of state descriptions. Thus, each node in the *schema* hierarchy has its own private *state* hierarchy.⁵ The top level of this hierarchy represents the PART-OF relations between each state and the schema as a whole. That is, the set of classes at the top level of the state hierarchy will be the state descriptions comprising the motor schema and will be ordered according to the values for the time attribute in the respective nodes.

Figure 2 shows the node in the movement hierarchy corresponding to overhand throws in slightly more detail; the other nodes in the hierarchy are similarly represented but we have not attempted a complete presentation of the memory structures for the purpose of clarity. The internal hierarchy of state descriptions introduced above is rooted in the node that it describes. This tree of state descriptions captures the structure of the abstract schema, and the time values stored in the state descriptions determine the temporal ordering. The figure shows the internal nature of one node in the hierarchy of state descriptions within the overhand node of the schema hierarchy. The mean and standard deviation for each of the attributes correspond to the first node in the three "overhand" schemas. Remember that each node in the hierarchy of motor schemas has its own internal hierarchy of state descriptions analogous to that shown for the overhand movement concept.

4. Recognizing a Movement with OXBOW

As previously described, OXBOW's performance task is to recognize an experienced movement in the environment according to the current knowledge base of movements. The recognition process can be broken into two sub-processes — parsing the motion and classifying the resulting parsed structure. This section describes these performance aspects of OXBOW, while the next section will focus on the learning methods used to modify and update the knowledge base in response to experience. As we mentioned in Section 2, learning and performance are closely tied in our view of concept formation. We separate them here only for the sake of presentation.

5. Although there are many individual hierarchies of state descriptions, we will refer to a single state hierarchy when the context of a particular node in the schema hierarchy is obvious.

4.1 Parsing a Movement

A movement is a continuous experience over some period of time. In order to understand a movement, OXBOW breaks the continuous experience into a set of discrete representations. This results in a sequence of snapshots of the environment (specifically the arm) over the course of a particular movement. Recall that our motor schema representation for movements stores only a few points in time for a given movement. The movement *parser* is responsible for selecting the points that are to be used for recognition and remembering.

We base our parser on Rubin and Richards' (1985) theory of elementary motion boundaries. They propose four primitive motion boundaries: starts, stops, steps, and impulses. The first two, starts and stops, represent zero crossings in velocity and are obvious choices for boundary points since without them it would be impossible to distinguish a period of movement from a period of rest. The second two, steps and impulses, refer to discontinuities of force. However, as Rubin and Richards admit, this set of motion boundaries is insufficient to represent many of the movements that we are interested in. We have augmented these elementary boundaries with an additional boundary that represents zero crossings in acceleration. This gives us the desired representational power at the expense of additional motion boundaries or states in a schema.

Given the boundaries defined above, we must still define how these are used to parse a given movement. The agent observes a movement (in discrete time slices as described in Section 3.1) and maintains current values for position, velocity, and acceleration for each of the joints in the arm. Whenever the value for either velocity or acceleration at any one of the joints in the arm has a change in sign, the position and velocity information for all the joints⁶ is collected and formed into a state description as specified in Section 3. Over the course of a movement, these boundary states are identified, generated, and collected. At the completion of the movement, the resulting sequence of states is returned as a single motor schema.

6. This represents a simplification on our part. Alternatively, we could store only the information for the joint that triggered a break point. Although our representation handles this, our implemented mechanisms would become rather more complicated.

Table 1. OXBOW's algorithm for movement classification.

```

Classify(movement, schema-node)
  If leaf(schema-node) or recognized(movement, schema-node),
    Then return schema-node;
  Else for each child of schema-node,
    Compute a score for Incorporating(child, movement).
    Let best be the child with the largest of these scores.
    If the score for putting movement by itself
      is greater than the score for best,
      Then return the schema-node;
    Else Classify(movement, best).

```

Note that the entire movement is parsed and that it is the resulting schema that is given to the classification mechanism for recognition. Theoretically, it would be possible (and perhaps desirable) to have the parser and classification mechanisms working more hand in hand. That is, as each boundary is observed and the associated state is generated and appended to the end of the partial schema, this partial schema could be classified. This could conceivably lead to advantages in constraining the work necessary for later classifications of the more complete schema. We will return to this idea in our discussion of future work.

4.2 The Classification Mechanism

Table 1 presents the basic OXBOW classification algorithm. At this level of abstraction, the classification process is no different from that used in Fisher's (1987) COBWEB and Gennari et al.'s (1989) CLASSIT. In these concept formation systems, the classification and hierarchy formation processes are tightly coupled. We have separated these two components to provide a different perspective on this algorithm.

Upon encountering a new instance I , the system starts at the root and sorts the instance down the hierarchy, using an evaluation function (described below) to decide which action to take at each level. The termination condition of this recursive algorithm corresponds to the instance already having been recognized. This can occur in two cases: the current node may be a leaf in the concept hierarchy, or the evaluation function may consider the current node to be close enough to the

instance that no further descent is necessary. The latter case requires the use of a ; as described in Gennari (1990), this parameter determines when the system “recognizes” an instance and is especially useful in noisy domains.

At a given node N where the instance I is still unrecognized, OXBOW retrieves all children and considers placing the instance in each child C_k in turn; it also considers the case where the instance would be treated as a separate child. The algorithm uses its evaluation function to determine which of the resulting partitions is “best”,⁷ and then continues either by recursively classifying with the chosen best or stopping and returning the current node as the classification value of the new instance.

More specifically, if the instance I is sufficiently different from all the concepts in a given partition according to the evaluation function, I is considered to be a member of a new category and no further classification is necessary (or useful). The current parent class is returned as the label of the new instance. The classification process halts at this point, since the new node has no children.

4.3 OXBOW’s Evaluation Function

We have mentioned that OXBOW uses an evaluation function to determine the appropriate branch to sort new instances down during classification. Since a major goal of concept formation is to let the agent categorize new experience and make predictions, the system employs — an evaluation function that attempts to maximize predictive ability. Gluck and Corter (1985) originally derived this measure from both game theory and information theory in order to predict basic-level effects in psychological experiments, and Fisher (1987) adapted it for use in his COBWEB model of concept formation. The measure assumes that concept descriptions are probabilistic in nature, and it favors clusterings that maximize a tradeoff between intra-class similarity and inter-class differences.

One can define category utility as the in the expected number of attribute values that can be correctly predicted, given a set of K categories, over the expected number of correct predictions without such knowledge, normalized by the size of the partition. This expression was

7. This lets the system avoid the need for an all-or-none match between the nodes in a given partition and a new instance being classified.

originally designed for nominally valued attributes, and summations of probabilities of attribute values. As used by COBWEB, these probabilities were computed simply by keeping counts of attribute values.⁸

OXBOW works with continuous attributes, and the original expression for category utility had to be modified for such domains (Gennari et al., 1989). For such attributes, probabilities are computed by assuming a normal distribution of values and finding the standard deviation over observed instances. More precisely, category utility for continuous attributes is

$$\frac{\sum_k^K P(C_k) \sum_i^I \frac{1}{\sigma_{ik}} - \sum_i^I \frac{1}{\sigma_{ip}}}{K}, \quad (1)$$

where $P(C_k)$ is the probability of class C_k , K is the number of classes at the current level of the hierarchy, σ_{ik} is the standard deviation for an attribute i in class C_k , and σ_{ip} is the standard deviation for attribute i in the parent node.⁹

However, this expression assumes that every class consists of a simple list of attributes. For OXBOW, we must extend this to consider classes with a set of components, or in this case, state descriptions. Because the number of states is not the same for all movement instances, the information in each component is weighted by the probability of that component. For OXBOW, category utility is

$$\frac{\sum_k^K P(C_k) \sum_j^J P(S_{kj}) \sum_i^I \frac{1}{\sigma_{kji}} - \sum_m^M P(S_{pm}) \sum_i^I \frac{1}{\sigma_{pmi}}}{K}, \quad (2)$$

where $P(S_{kj})$ is the probability of the j th state description in class C_k , or the proportion of all the state descriptions from schema instances of node C_k that are classified at state description S_{kj} . The probability $P(S_{pm})$ is similarly defined for the m th state description in the parent of the current partition.

8. See Fisher and Pazzani (Chapter 1, this volume) or Thompson and Langley (this volume) for more details and discussion of COBWEB's category utility equation.

9. As discussed in Gennari et al. (1989), the value of $1/\sigma$ is undefined for any concept based on a single instance. We adopt the same solution using an *acuity* parameter, but are not greatly concerned with its value. See Gennari (1990) for empirical analysis of the impact of this parameter on system performance.

5. Learning from Unsupervised Experience

In Section 2, we introduced a learning task associated with the recognition of motor schemas. To review, the task involves incorporating a newly experienced movement and parsed motor schema into long-term memory in such a way that one can more accurately recognize similar movements when they are presented in the future. In Section 6, we define exactly what we mean by "more accurately" and present some experimental results that support our claim that OXBOW does succeed at this learning task. We begin this section by describing the learning algorithm at a high level, at which the system borrows many ideas from Gennari's CLASSIT and Fisher's COBWEB. Then, we proceed to the details of incorporating new movements into an existing schema class; this is where OXBOW makes some important extensions to previous work.

5.1 The Learning Algorithm

Table 2 provides a brief description of OXBOW's learning algorithm. Again, because learning and performance are integrated, the learning algorithm looks similar to the classification algorithm introduced earlier. The primary difference is that the system makes permanent changes to memory structures when learning, whereas the original memory structure is retained for future use when classifying. There are some subtle differences as well. As with classification, the system considers incorporating the new instance in each of the existing children of the current node, as well as creating a new child with the single instance. However, when learning, the instance must be permanently incorporated into the nodes of the schema hierarchy along the path from the root to the leaf where the instance I is placed. If the instance I is sufficiently different from all the concepts in the current partition, a new singleton class is created containing I . In this case, the learning procedure halts since the new node has no children.

Sometimes peculiarities in the order in which movements are observed can lead to an "incorrect" hierarchy structure. For example, this can occur when, after forming two classes of movements based on experience, the system observes several new instances which at first appear to be minor variants of one of the two existing classes. However, as OXBOW gains additional experience, it becomes clear that this "variant" is actually a distinct class representing a separate movement concept. In

Table 2. OXBOW's learning algorithm.

```

Build-Tree(movement, schema-node)
  If leaf(schema-node) or recognize(schema-node, movement),
    Then halt;
  Else for each child of schema-node,
    Let best be the result of
      Incorporate(child, movement) with the best score.
    Let second be the result of
      Incorporate(child, movement) with second best score.
    Compare four cases, letting selected-child be the best of:
      best;
      by-itself;
      merge(best, second);
      split(best).
    If selected-child is by-itself,
      Then halt;
    Else Build-Tree(movement, selected-child).

```

such cases, the concept formation system should be able to gracefully recover from previous errors. Therefore, in addition to comparing the result from incorporating instance *I* into the best of the current children and creating a new singleton class containing *I*, OXBOW considers two alternative actions. One involves combining the two best existing children into a single node. In this case the subtrees are spliced together such that the new node's children are the union of the children of the best and second best nodes. This new combined node is evaluated within the partition with the remaining nodes. The other alternative considers replacing the best child with its constituent subtree branches. That is, all the children of the best node are promoted and become direct children of the current node, and the best node disappears.

These final two actions are referred to as *merge* and *split* operations. They are intended to aid the system in recovering from poor choices earlier in training, perhaps due to order effects. Fisher and Pazzani (Chapter 1, this volume) argue that some form of "backtracking" operators are necessary for any incremental learning system, particularly in pure hill-climbing systems such as OXBOW, which can get stuck at local optima. One can imagine additional operators that would take

larger steps through the space of possible partitionings. However, these amount to some sequence of applications of the simple merge and split operators. This does not mean that such compound operators may not be necessary in order to find an ideal concept hierarchy (a learning evaluation issue), but they are not necessary in theory.

We now turn to the largest difference between OXBOW and concept formation systems such as CLASSIT — the instance incorporation process. This difference arises due to the structural nature of our motor schema representation.

5.2 Incorporation of Motor Schemas

Every concept formation system must address the question of how to incorporate a new instance into an existing class. This is the essence of learning in these systems. An evaluation function can be used to determine which node, out of a set of candidates, should have the instance incorporated. But the incorporation process actually changes the memory structures and lets one make useful predictions from the stored information.

In Fisher's COBWEB, incorporating a new instance was a simple matter of updating the counts associated with a class node based on the attribute values occurring in the instance. The system assumed that each instance had a fixed set of uniquely labeled attributes (although an instance could be missing a value for a given attribute). Gennari's CLASSIT (1989 version) extended this to include a notion of structured objects made up of multiple components. These objects were restricted to a single level of components, where each component was a primitive object analogous to the objects given to COBWEB. Also, CLASSIT assumed that each structured object had the same number of components and that each component occurred in each instance. That is, the structure of these objects was uniform across all classes and did not vary.

Here we are interested in forming concepts of movement representations (as defined earlier), and neither COBWEB nor CLASSIT has satisfactory mechanisms for handling this task. Recall that a motor schema is a compound object made up of components that represent the states of the arm at specified time values. Each state satisfies the notion of a primitive object since we are mainly dealing with the parts of the arm; each joint contributes its own unique attributes to the total description

of a state.¹⁰ However, because a motor schema may have any number of state descriptions, there may not be a one-to-one correspondence between two schemas' states. Therefore, one cannot uniquely associate the attributes (at the state description level) from one schema to another.

OXBOW includes a solution to this correspondence problem that is specific to temporally structured domains, but that allows more flexibility than the one implemented in CLASSIT. Both systems must determine mappings between components in an instance and components in a stored concept. However, OXBOW can combine an instance and a concept with differing numbers of states by allowing multiple states in the instance to map onto a single state in the concept, and by allowing individual states in the instance to become new and separate states in the concept. The category utility scores for incorporating single states from the instance into the hierarchy of state descriptions determines the mapping between the instance and the concept. This method is clearly more flexible and (we believe) more elegant than CLASSIT's, although both methods have the same $O(n^2)$ computational complexity, where n is the number of components in the concept.

For example, suppose OXBOW observes a movement that is parsed into a schema having three states. In the process of incorporating this schema into the memory presented in Figure 2, the system must consider including it as an "overhand" schema. This involves establishing the mapping between the states in the observed movement and those in the schema node. The general solution applied here is to use category utility as an evaluation function for determining how to match states from respective schemas with each other and for deciding when to leave states unmatched (in the case where category utility prefers creating a new disjunct). This application of category utility is based upon treating each state of a new schema to be incorporated as a separate instance in and of itself. However, instead of passing each state down through the hierarchy of motor schemas, they are passed through the separate PART-OF hierarchy within the schema node under consideration.

More specifically then, for a given state, S , and a hierarchy of states associated with a node in the hierarchy of schemas, we execute the same learning algorithm as described in Section 5.1 with the following

10. Of course, one could think of each state again as a compound object made up of components corresponding to the parts of the arm. This goes beyond the scope of the current research and we leave it to future work.

differences. First, at this level “incorporate” simply involves updating all the attribute counts, means, and variances for the given state. State descriptions can be thought of as primitive objects with a fixed set of attributes that can be compared between states. Second, the evaluation function used is a simplified version of category utility. Because its goal is to capture the temporal structure present in the data, OXBOW only considers the time attribute in determining the score, instead of summing over all the attributes. The resulting form of the equation is

$$\frac{\sum_k^K P(C_k) \sum_j^J P(S_{kj}) \frac{1}{\sigma_{kjtime}} - \sum_m^M P(S_{pm}) \frac{1}{\sigma_{mp_{time}}}}{K}, \quad (3)$$

where σ_{kjtime} and $\sigma_{mp_{time}}$ are the standard deviations for the time attribute in the j th state of class C_k and the m th state of the parent, respectively. All of the attributes that describe a state are updated when a new state is incorporated, but only the time attribute is considered when evaluating the score for a node and its children.

The incorporation of a new schema instance effectively establishes a mapping among states. As each state in the new instance is considered individually, it is either “mapped” onto one of the existing states when it is incorporated, or onto nothing when it becomes a separate state by itself. Given this implicit mapping, we can ignore the structure of the schema and compute the score of the partition at the motor schema level. We remove the structural information by treating the attributes of each state’s description as unique at the motor schema level. That is, a schema consisting of three state descriptions, each with 13 attributes, would have three times that many ($3 \times 13 = 39$) unique attributes used in the calculation of category utility. This process is reflected in the additional nested summation in equation (2): sum over states, and for each state, sum over the state’s attributes. In other words, states are classified only with respect to time, whereas schemas are classified with respect to all of the attributes.

Since schemas are composed of the top-level nodes in the state hierarchy,¹¹ we may think of this hierarchy as representing the PART-OF structure for the schema. We believe this is justified because the top level of a concept hierarchy reflects a *partitioning* of some outer envi-

11. Lower levels of the state hierarchy are retained in case subsequent merges and splits are necessary. They do not enter into the current argument.

ronmental context, in this case the entire motor schema. Our claim is not that the top-level nodes of a state hierarchy are parts of the root for that hierarchy, but instead are parts of the associated schema concept. For example, the first state in the internal hierarchy of the overhand node from Figure 2 is not PART-OF its parent, which summarizes all the state descriptions of the overhand schemas, but rather is PART-OF the overhand concept represented in the hierarchy of motor schemas.

The COBWEB and CLASSIT systems use category utility to determine IS-A relationships between instances (and classes) and more general classes. OXBOW uses the same function to determine appropriate matches between parts of complex objects. Every instance processed by a concept formation system can be thought of as PART-OF the environment being addressed. That is, some agent or mechanism parses the world and hands "instances" to the learning agent one at a time to be incorporated. These instances are used to construct a concept hierarchy where the children of every node except the root share IS-A relations with their parent. Note that the top-level nodes are not instances (or specializations of an abstract description) of the environment but rather are parts of the environment. Therefore, we have a PART-OF structure at the top-most level of the hierarchy and IS-A relationships below this.

OXBOW takes advantage of this characteristic by creating hierarchies of states in which the top level provides the states to be used in the motor schema. This works out conveniently because motor schemas are presented as parsed structures consisting of a sequence of states. Although we do not propose our system as the final solution to learning structured concepts, we consider it satisfactory for our present purposes and the intended context of this work.

6. The Performance of OXBOW

OXBOW provides a method for representing jointed limb movements and for acquiring a concept hierarchy of movement concepts. As stated in Section 2, one research goal is to demonstrate that the system is effective. To show this, we present some initial experiments that evaluate how accurately OXBOW can learn about and recognize movements.

As described in Section 4, the general performance task for our system is to recognize movements. More precisely, OXBOW must *classify* a newly presented movement given an existing concept hierarchy. This

classification means that the system associates the new instance with the node in the hierarchy representing previously observed movements that are most similar to the new movement. The system is evaluated by comparing the movement described by this chosen node with a prototypical, or idealized, movement. This comparison is performed by finding the Euclidian distance between corresponding joints of the arm at a given time. We average over the joints of the arm and over the testing movement period. The error scores we report in the following experiments reflect this averaging over joints and simulated time slices as well. The units given are for an arm with two joints operating in a reachable workspace of 100 unit radius.

6.1 Experiment 1: Learning Single Movement Concepts

Recall that OXBOW's representation of movements (as schemas) consists of parts (states) and the temporal relationships among them. The learning algorithm treats schemas in two passes — first as a complete sequence, and then within this as individual states. One of the first things we should verify is that the inner treatment — the determination of the PART-OF structure for the movement concept at large — is behaving appropriately. Therefore, in our first experiment we trained the system on instances from a single concept, so as to control any possible confusion between different movement concepts. This should also let us determine the system's sensitivity to variance in the observed data.

To this end, we have created four artificial movement templates that roughly correspond to a slap, a throw, a wave, and a salute. Observed movements in this experiment (and those following) were produced by motor schemas instantiated from the templates. The time, position, and velocity values in a motor schema used for training were drawn from normal distributions given in the templates. We adjusted the *variability level* of the distributions by a scale factor to produce different degrees of variation in the movements.

In this experiment, we tested one movement type and averaged the learning curves over 20 runs of 15 learning trials. We repeated this procedure at four different levels of variation for each of the four movement types and measured the system's performance after every other learning trial. We measured this performance by comparing the prototype movement with the schema stored at the root node of the schema hier-

archy. This approach avoids all issues of retrieval: in this experiment, there is only one movement type, and thus, the root node is always the “recognized” movement because all the schemas in the hierarchy below the root are of the same movement type.

Figure 3(a) shows four learning curves, summarizing the reduction of errors as a function of experience. The curves are labeled according to the corresponding level of variability: a relatively clean movement with little variation, and movements generated by doubling, tripling, and quadrupling the standard deviations in the generating template. Each learning curve represents the average improvement over the four different types of movement, for a single level of variation.

We can draw two conclusions from this figure. First, as might be expected, the learning rate decreases as the amount of variation increases. Note that after the first instance, error has decreased drastically at all four levels.¹² However, at the lowest level, error drops immediately to its asymptote, and at the highest level, it requires approximately eight more training instances. Second, we see that asymptotic levels increase as a function of variability. These results indicate that OXBOW has trouble finding the central tendency in domains with high variance. Because the data come from a single prototype, we would expect that the prototype would be recoverable. This could be due to problems determining the values within the states of the learned motor schemas, to problems finding the correct structure of states themselves, or to a combination of both.

To help discern this issue, Figure 3(b) shows a different look at the data, graphing the asymptotic error levels for each movement type separately as a function of the structural complexity inherent in the data. Additionally, it shows how the asymptote and complexity change with different levels of variation in the data. We define complexity as the number of states in a parsed description of an observed movement. For a given movement type and a single level of noise, we computed the average complexity over 20 randomly generated movement instances.

From Figure 3(b) we see that changing the variability in the generated movements does not cause large changes in the structure of the parsed movements. For example, with the “slap” movement, as the

12. Prior to any learning, we can define error to be the prototypical movement compared to a stationary arm. For random starting positions, this would be on the order of 150 units.

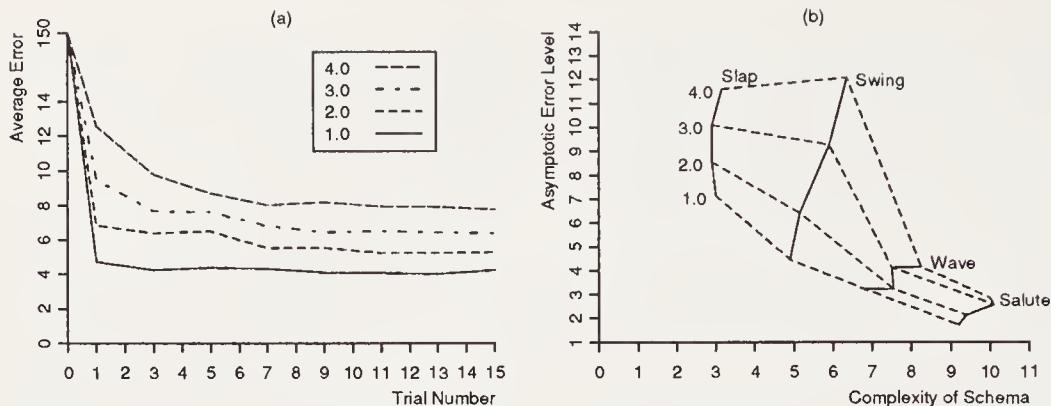


Figure 3. (a) Average learning curves when trained separately at four levels of variation in the data, and (b) the relation between asymptotic error levels and complexity of input data at different levels of variation.

variance of the observed movements increases, the error increases, but the structural complexity of the learned schema changes only minimally. The "wave" and "salute" movements do show some increase in structural complexity, but these have relatively little increase in asymptotic error. This stability of the parsed structures with respect to variability suggests that the increased asymptote levels in Figure 3(a) are not a result of a failure to determine the appropriate PART-OF structure for the movement concept, but rather are due to problems in determining the correct values within states.

This figure also reveals an unexpected result — that increasingly complex movements may have lower asymptotic error levels. This non-intuitive result is not without precedent; for instance, vision researchers found that more complexity in the environment made things easier to disambiguate. This would suggest that OXBOW could scale up to more complex environments and movements. We are currently considering these issues further to determine where to assign credit and blame, and the extensibility of our methods.

To conclude, this experiment indicates that OXBOW is able to capture the PART-OF structure found in observed movements when faced with only a single type, but that its ability to form accurate state descriptions is influenced by variability. We next turn to the larger problem of acquiring movement concepts from observed data when presented with instances of more than one type.

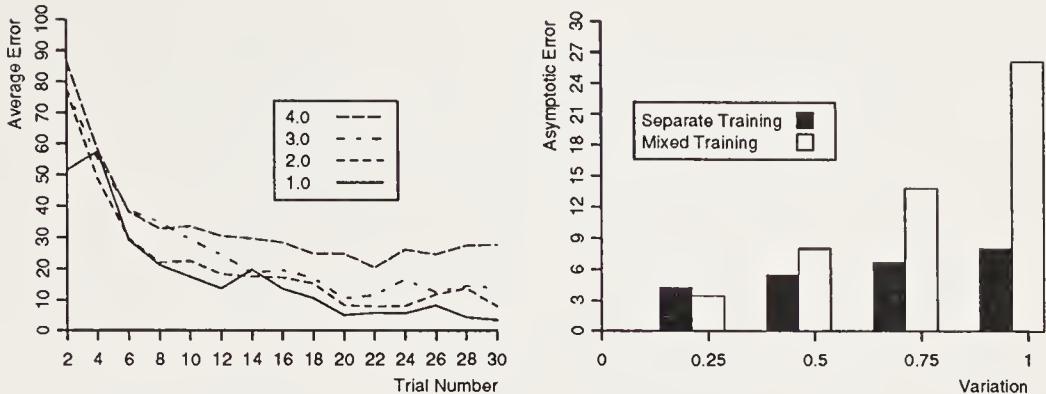


Figure 4. (a) Learning curves when OXBOW is trained on multiple classes simultaneously at four levels of variation, and (b) a comparison of the average performance after seven learning trials when trained separately to the average performance after 30 trials when trained together on four different classes.

6.2 Experiment 2: Forming Multiple Movement Concepts

If we had first tested OXBOW on acquiring multiple concepts simultaneously, we would not have known whether performance errors were caused by confusions between categories when classifying an observed movement, by problems identifying the appropriate PART-OF structure for a particular node in the hierarchy, or both. The previous study established a baseline for comparison. We can expect that errors above and beyond those reported in Section 6.1 result from classification.

To study such errors, we ran a second experiment in which OXBOW observed movements from all four of the classes, each with an equal likelihood. We presented 30 training instances, from which the system constructed its hierarchy of movement concepts. After every other training instance, we stopped learning and tested the system's performance as described above. We repeated this experiment at the same four levels of variation in the movement generators. Figure 4(a) shows the average error (over the four movement types), again as a function of experience and variance level. The errors are averaged over ten runs of 30 training instances with random orders of the movement types.

The figure gives some indication of how well OXBOW is distinguishing between observed movements of different types, even though after 30 training instances the system has only seen (on average) one-fourth that many training instances of each type. Figure 4(b) shows a comparison

between the performance after 30 training instances from Figure 4(a), under mixed training, and the performance level after seven training instances in Figure 3(a), under separate training. This reveals an interaction between the training method (mixed/separate) and the variability in the domain; increased variability has a much greater impact on error when learning multiple concepts than when learning single concepts. In Experiment 1, we were able to control for retrieval errors because only a single movement type was given. In this experiment, we relied on OXBOW's retrieval mechanism and therefore increased errors could be attributable to misclassifications during either training, retrieval, or both. One option to clarify this would be to repeat Experiment 1 and rely on OXBOW's retrieval mechanism, but another option is to evaluate the retrieval mechanism in isolation, as we report in the next section.

6.3 Experiment 3: Predicting Unseen Movement

In the previous experiments, the performance measure corresponded to what has been termed *recognition* in the psychological literature. That is, the complete prototype of a particular movement class was classified and a comparison was made across the entire duration of the movement. In real life, one would more likely observe a partial movement and need to predict the continuation of the movement. Such a task directly tests the retrieval mechanism, allowing an assessment of its performance.

Thus, in a third experiment, we trained OXBOW as described before, but altered the performance mechanism slightly. When testing, we presented only a portion of the prototype movement and measured error over the remaining unobserved movement. Because we average over time when determining error, we can compare errors among different ratios of observed movement to predicted movement, even though the number of predictions changes. Any differences in errors will be attributable to classification problems during retrieval because the knowledge base is the same for each level of observation at a given point in training.

This formulation of the task suggests a prediction: as less of the movement is observed, classification should become more difficult and mistakes should lead to greater measured error. Simply stated, the more one is able to observe, the more certainty there should be about succeeding movement. Figure 5 shows the learning curves from an experiment in which we varied the portion of the movement to be predicted. We

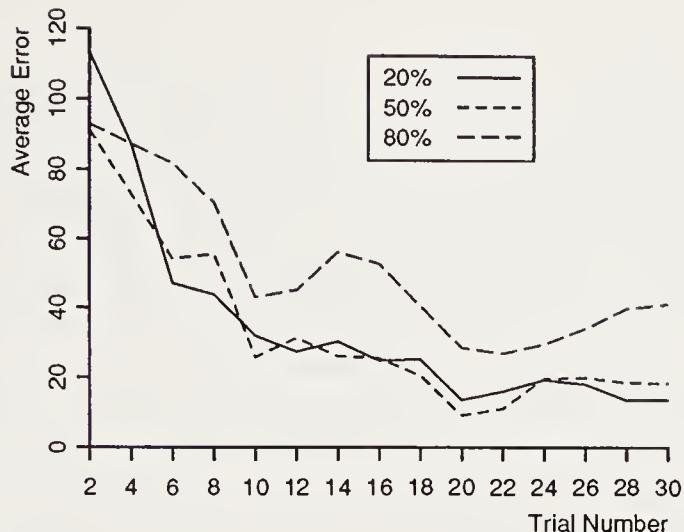


Figure 5. Learning curves showing error as a function of the amount of missing test movement that must be predicted.

fixed the variation level at 2.0 and averaged the results over ten runs of 30 training instances each.

The figure shows that when OXBOW is predicting 80% of the movement (observing only the first 20% of the movement), the errors are consistently the highest (except very early in training, when not all the movement types have yet been seen). However, there is little difference between observing 50% of the movement and observing 80%. This result suggests that the system is not adversely affected by having less information available for classification except in extreme cases like the 80% condition. Our model of motor learning makes an additional prediction for this type of performance task. As the variability in the training data increases, OXBOW should need to observe larger portions of the test movement in order to prevent the predictive error from increasing. This is an intuitively appealing prediction that lends to the appeal of our model.

In conclusion, this experiment holds other factors constant while varying the amount of information in the test movement, thus indicating the sensitivity of the classification process during retrieval. The results suggest that OXBOW is able to correctly classify movements when given only partial structures as input. Again, this suggests that performance problems are due to the mechanism for storing state descriptions rather than to the classification or retrieval process.

7. Discussion

In this chapter, we have presented a computational model of movement recognition. Such a model is one part of a complete model of motor learning. We claim that a full model of motor behavior must address both recognition *and* generation of movements. Likewise, both of these tasks should improve over time and experience. In this context, OXBOW has been developed as one of two subsystems for a larger, more complete model of human motor control — MÆANDER (Langley, Thompson, Iba, Gennari, & Allen, *in press*). MÆANDER, with its sub-systems OXBOW and MAGGIE, describes a complete computational model of human motor behavior. The entire model has been instantiated as a working program in LISP. Evaluating the success or failure of this system is our ongoing research goal.

The other subsystem, MAGGIE, takes a viewer-centered schema as a description of a desired movement and attempts to manipulate an arm in an identical manner (Iba & Langley, 1987). MAGGIE takes the motor schemas acquired by OXBOW and uses them to generate similar behaviors. However, the viewer-centered schemas cannot be used by the arm to control movements because the movement at each joint must be specified in local polar coordinates. Therefore, the movement generation process involves converting the viewer-centered schemas described in this chapter into joint-centered schemas with local joint information. Using the joint-centered schema, the system interpolates between the times of the state descriptions in order to generate the actual movement.

As we stated at the beginning of this chapter, we are interested in two methods of evaluation: comparison to psychological results, and demonstrating improvement with a performance task over time. Additionally, for a complex system with subsystems, one should test both the individual components as well as the system in entirety. This chapter includes initial empirical studies that test OXBOW's ability to group complete schemas and to determine the appropriate temporal structure for classes of movements. However, to completely evaluate the entire system requires six different types of experiments: psychological and performance studies for each of MÆANDER, OXBOW, and MAGGIE.

Experiments with the MAGGIE component have already been carried out and are reported in Iba and Langley (1987). Currently, we are carrying out additional experiments with OXBOW and testing MÆANDER

as a whole. Measuring performance for the entire system involves presenting a set of different (or repeated) movements, letting it build a hierarchy of "acquired" motor skills, and then letting it use one of these stored concepts to generate movement. This means handing the chosen motor schema to MAGGIE for movement generation. Evaluation would involve comparing the generated movement to the original observed movement. As with a novice observing an expert, the learner's movement should more closely resemble the expert's skill as the learner observes and practices these movements.

Clearly, we need additional experimental evaluation before we can make firm statements about the utility of MÆANDER and its subsystems. However, we can outline several weaknesses of OXBOW itself independently of any empirical results. First we note that each node in the motor-schema hierarchy has its own internal hierarchy of state descriptions. Although some separation is necessary from the partitioning of schemas, it seems sensible to have some shared structure among similar schemas. A second problem is that our motor schemas have a single level of component structure. It does not immediately seem practical to extend our model to structured representations with arbitrary nesting of components, since memory requirements would grow exponentially in the number of primitive component descriptions. Finally, recall that in the process of incorporating a new schema instance into a node in the schema hierarchy, we treat the time attribute as special. This is appropriate because we are concerned with storing and organizing state descriptions according to a temporal ordering. However, in the general case we are interested in relations other than temporal orderings.

The preceding discussion points out several directions for further research. As we stated above, we intend to further evaluate the current version of OXBOW. In addition to examining the psychological plausibility of the model and the predictions that it makes, we want to answer the questions raised by the current experimental results. For example, how does asymptotic performance vary as a function of representational (structural) complexity? How does overall performance vary with the number of movement concepts being presented simultaneously? Why do the values in the state descriptions seem to cause errors when the states themselves are formed and mapped appropriately? Finally, what are the individual problems in the classification and incorporation mechanisms?

We also want to extend this work in the direction of increased flexibility within the movement concepts stored in memory. Consider a situation in which a frisbee stuck in a tree must be retrieved by throwing a ball upwards to dislodge the disc. Ideally, the pitching schema for throwing straight ahead would be adjusted to recognize the event as a throw. Currently an abstract schema represents a set of movements, but only the average movement can be recalled. Of course, more specific schemas that are summarized by the schema in question can be accessed farther down in the schema hierarchy. However, in general we would like to be able to use any given schema in a slightly modified manner or context.

8. Conclusion

Despite any shortcomings of our model, we feel that it does make a number of important contributions. First, we have built a flexible representation for modeling movements. This representation can be applied to a wide range of motions and we have found it to be satisfactory for our model of human motor learning. As we said earlier, this representation fills a gap between robotics, which generates movements with low-level models, and psychology, which works with high-level models without complete computational models.

Second, we have uncovered an interesting and exciting duality between IS-A and PART-OF relations. The duality depends upon the context of the instances that are being observed by the concept formation system and the interpretation of the root node. An instance stored in the hierarchy "IS-A" member of the set of all experiences, but it is also a "PART-OF" the learning agent's environment (at least at some point in time). We are currently exploring the implications of this duality and believe that a more complete understanding of concept formation will result from this insight.

Finally, by exploiting this duality, we have been able to extend concept formation methods to a new class of domains. Although there has been some research in concept formation with structured data (Stepp & Michalski, 1986; Thompson & Langley, this volume), most work has been restricted to instances described by simple attribute-value vectors. By using category utility on the nodes in the "part-of" tree, and therefore by establishing a labeling between states in a new instance and

states in a stored motor schema, we have applied the concept formation ideas of COBWEB and CLASSIT to structured objects.

Our empirical results lend weight to the claim that the representation and methods used in OXBOW are viable solutions to concept formation for the class of temporally structured domains. Furthermore, our results indicate that the system makes appropriate matches between state descriptions in the input data and those in the stored concepts, and that predictive errors are not attributable to this process. Although additional research is necessary to determine the limits of the system's applicability, our results to date cause us to be optimistic.

Acknowledgements

The work described in this chapter was supported in part by Contract MDA 903-85-C-0324 from the Army Research Institute. We owe thanks to Doug Fisher, Michael Pazzani, and Pat Langley for useful comments on earlier drafts, and to other members of the ICARUS group — John Allen, Pat Langley, Kathleen McKusick, and Kevin Thompson — for useful discussions. The experimental results presented in the chapter are also reported in Iba (in press).

References

- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54–64). Ann Arbor, MI: Morgan Kaufmann.
- Fried, L. S., & Holyoak, K. J. (1984). Induction of category distributions: A framework for classification learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 10, 234–257.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139–172.
- Fisher, D. H., & Langley, P. (1990). The structure and formation of natural categories. In G. H. Bower (Ed.), *The psychology of learning and motivation: Advances in research and theory* (Vol. 26). Cambridge, MA: Academic Press.
- Gennari, J. H. (1990). *An experimental study of concept formation*. Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

- Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11-61.
- Gluck, M., & Corter, J. (1985). Information, uncertainty and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283-287). Irvine, CA: Lawrence Erlbaum.
- Iba, W. (1991). *A computational model of human motor control*. Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.
- Iba, W. (in press). Learning to classify observed motor behavior. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*. Sydney: Morgan Kaufmann.
- Iba, W. (1989). *Human motor behavior: A short review of phenomena, theories, and systems* (Tech. Rep. No. 89-34). Irvine: University of California, Department of Information & Computer Science.
- Iba, W., & Langley, P. (1987). A computational theory of motor learning. *Computational Intelligence*, 3, 338-350.
- Langley, P., Thompson, K., Iba, W., Gennari, J. H., & Allen, J. A. (in press). An integrated cognitive architecture for autonomous agents. In W. V. De Velde (Ed.), *Representation and learning in autonomous agents*. Amsterdam: North Holland.
- Rubin, J. M., & Richards, W. A. (1985). *Boundaries of visual motion* (AI Memo 835). Cambridge, MA: Massachusetts Institute of Technology, Laboratory for Artificial Intelligence.
- Schmidt, R. A. (1982). The schema concept. In J. A. S. Kelso (Ed.), *Human motor behavior: An introduction*. Hillsdale, NJ: Lawrence Erlbaum.
- Smith, E. E., & Medin, D. L. (1981). *Categories and concepts*. Cambridge, MA: Harvard University Press.

CHAPTER 14

Representation Generation in an Exploratory Learning System

PAUL D. SCOTT

SHAUL MARKOVITCH

1. Introduction

The majority of machine learning systems can only operate successfully with the assistance of some external agent to provide guidance in the learning task. Such guidance takes various forms. Supervised concept learning systems, which learn classification rules from sets of classified examples (e.g., Winston, 1975; Mitchell, 1977; Dietterich & Michalski, 1981; Quinlan, 1983), require an external agent to define the set of classes, to supply a set of training examples, and to classify the training examples. In contrast, conceptual clustering systems, which construct their own classification schemes from a set of training examples (Michalski & Stepp, 1983; Fisher, 1987), only require the external agent to supply the examples. Comparatively little work has been done on exploratory learning systems that learn without any form of external assistance. Lenat's (1982) AM is one of the best known programs in this category.

This paper describes some of the features of DIDO, a program we have developed to investigate possible solutions to the problems that arise in exploratory learning. The basic task is to acquire the knowledge necessary to solve problems in a novel domain, which is made up of discrete entities that DIDO can observe. For a given domain, the program is equipped with a repertoire of operations that can be applied

to entities in the domain, but it has no prior knowledge of the consequences of applying any of the operations. DIDO's goal is to acquire sufficient knowledge to predict the outcome of applying any operation to any object that may be encountered. Such knowledge is what would be required to solve problems by discovering sequences of operations that lead to solutions. This learning task must be carried out with minimal external assistance.

Several additional constraints are placed on the program. The domain may contain a very large number of observably distinct entities, and the consequences of applying operations may not be consistent; that is, the effects of operations may be non-deterministic or may vary over time. DIDO only has limited computational resources available, so the learning procedure must be conservative in both space and time. The program has no knowledge of the type of problems it may be called upon to solve and is therefore unable to direct its learning towards knowledge relevant to a particular class of problems. Finally, DIDO may be asked to solve a problem at any time, and so it is highly desirable that, at any given time, the knowledge representation should incorporate as much as possible of what can be derived from the program's experiences up to that time.

The present paper is concerned with the methods DIDO uses to construct its knowledge representation.¹ Other aspects of the program, including methods for generating informative experiences and the type of knowledge representation employed, are discussed in greater depth elsewhere (Scott & Markovitch, 1988, 1989). Here we describe such features only as far as is necessary for an understanding of the operation of the program as a whole. In particular, Section 2 discusses the problems that must be solved in building an exploratory learning system. Section 3 briefly describes the solutions adopted in DIDO and provides an overview of the way the program operates. The fourth section is devoted to a detailed account of the techniques DIDO uses to modify the knowledge representation on the basis of experience. Section 5 discusses a variety of experimental results obtained with the program, and the final section contains an assessment of DIDO's strengths and weaknesses, together with a discussion of the broader implications for work on exploratory learning.

1. In referring to the 'construction of representations', we mean the creation and definition of concepts, and not the deeper sense of this phrase used in research on change of representation languages (e.g., Benjamin, 1990).

2. Problems of Exploratory Learning

Because of the complete absence of external assistance, exploratory learning systems must solve problems that are in addition to, or take a different form from, the problems confronting other types of learning systems. In this section, we examine these problems.

2.1 Learning as Two Interacting Search Processes

Any learning procedure can usefully be viewed as comprised of two interacting search processes (Shalin, Wisniewski, Levi, & Scott, 1987). The first, *representation generation*, is a search of the space of possible representations for a good description of the domain (Simon & Lea, 1974; Mitchell, 1979, 1982). The other, *experience generation*, is a search of the space of possible experiences for informative examples (Scott & Vogt, 1983). These search processes interact in two ways: the examples generated by the search of experience space are used to guide the search for a good representation, and the current state of the representation is often used to guide the search for informative examples. The components of a learning system responsible for each of these search processes may conveniently be termed the *representation generator* and the *experience generator*.

The task of developing a learning system can therefore be viewed as one involving the creation of two search processes that work effectively within the constraints imposed by the particular situation in which the system is required to learn. Typically, the spaces to be searched are very large and available resources are limited, so exhaustive methods are not feasible, and heuristic search techniques must be employed. We therefore proceed to consider the problems involved in developing suitable heuristic search procedures for each search component of an exploratory learning system.

2.2 Problems Arising in Experience Generation

The task of the experience generator is to select informative examples that will lead to changes in the system's representation of its domain. Whether or not a specific example is informative at a particular time depends not only on the characteristics of the example itself but also on the current state of the learning system's representation of the domain,

on the method employed by the representation generator to modify that representation, and on the problem domain as a whole. Since a very large proportion of potential examples may be uninformative, an effective experience generator requires some method of identifying those that are likely to be among the small informative minority.

Both supervised concept learning systems (Winston, 1975; Mitchell, 1977, Dietterich & Michalski, 1981; Quinlan, 1983) and conceptual clustering systems (Michalski & Stepp, 1983; Fisher, 1987) depend upon an external agent to generate sets of training examples. In contrast, an exploratory learning system must include an internal experience generator that can select informative examples without such external assistance. Such a generator does not have the same information available to it as a typical external teacher. Almost by definition, the learning system knows less about the domain; otherwise, there would be no learning task to perform. Hence the experience generator must use an example selection criterion that is essentially domain independent. However, this does not imply that domain-specific knowledge cannot be used in the selection process. The learning system is engaged upon acquiring knowledge of the domain and this learned knowledge could be used to guide the experience generator. Consequently it may even be desirable to build a representation that incorporates knowledge specifically for the guidance of the experience generator.

The internal experience generator's disadvantage of domain ignorance is offset to some extent by access to information not normally available to an external agent: the current state of the representation. The discrepancies, between the problem domain and learning system's model of it, largely determine which examples will be informative. An external agent must usually infer the current state of the representation from evidence obtained from incorrect responses or from knowledge of the learning system's mechanism and history. An effective experience generator should exploit its direct access to the representation as far as possible in selecting informative examples.

DIDO addresses these problems by selecting training examples on the basis of the degree of uncertainty present in the current representation. The details of this experience generation technique and our reasons for adopting it are outside the scope of this paper, but they can be found in Scott and Vogt (1983) and in Scott and Markovitch (1988, 1989). Gross (1988) presents a related approach to experience generation.

2.3 Problems Arising in Representation Generation

The representation generator must find a good representation of the problem domain among all those that may be constructed in the system's representation language. Supervised learning programs (e.g., Winston, 1975; Mitchell, 1977; Dietterich & Michalski, 1981; Quinlan, 1983) receive considerable help in this task from an external agent that classifies training examples. Such classifications guide the search for a good representation both by defining the set of categories to be discriminated and by indicating training examples that are classified incorrectly.

Exploratory learning systems resemble conceptual clustering programs (e.g., Michalski & Stepp, 1983; Fisher, 1987) in that their representation generators do not receive the guidance that conceptual learning systems derive from example classification. Both types of programs must therefore contain their own internal domain-independent criteria for evaluating the quality of representations. How is it possible to construct such an evaluation criterion without reference to an external agent to decree what is correct?

Some aspects of a representation, such as consistency and parsimony, can be evaluated without any reference to any external standard, but these are not sufficient to identify good representations: even the null representation is both consistent and parsimonious. There must therefore be some reference to an external standard, but this cannot be an external evaluation of the system's performance. Rather it must relate to the ultimate goal of the learning system. It is reasonable to assume that the representation produced by the exploratory learning or conceptual clustering will be used for some purpose. A good representation is one that serves this purpose well. For example, COBWEB (Fisher, 1987) builds a representation that will be used to predict unknown attributes of an entity from those that are known. It therefore uses a measure of the likelihood of such predictions being correct as the evaluation criterion to guide its search for a good representation. In contrast, DIDO builds a representation that will use known attributes to predict *changes* in attributes consequent upon an action, and hence requires a different evaluation criterion that we describe later.² In general, one might expect there to be at least as many good evaluation criteria as there are purposes that the resulting representations will serve.

2. Shen (1990) describes a related approach to exploratory learning about the effects of actions on the environment.

3. An Overview of DIDO

In this section we present the main features of DIDO's structure and operation. In order to interact with a domain and hence discover its characteristics, the program requires some form of interface with the domain. Any such discoveries must be represented internally and hence the program also requires a knowledge representation scheme. The systematic exploration of the domain in order to acquire a good representation depends upon a strategy for searching the space of possible experiences, while the process of building such a representation using the information supplied by experiences depends upon a strategy for searching the space of possible representations. Each of these components of DIDO is described below.

3.1 Interface with the Domain

DIDO is implemented in a completely domain-independent fashion. In order for the program to explore a new domain, it must be supplied with a *sensory interface* and a *motor interface* that provide communication with a *domain implementation*.

A sensory interface is a set of named two-place predicates, called *perceptual properties*, that are used to determine whether specified entities satisfy particular relations in the current state of the problem domain. The first argument of a perceptual property must be an entity in the domain, and the second can be either an entity or a constant denoting the value of some attribute. DIDO's knowledge of the sensory interface is confined to the names and the arities of the perceptual properties, plus the fact that the first argument must be an entity. Perceptual properties form the basis of the classification scheme that DIDO develops through exploratory learning.

DIDO's motor interface comprises a set of named functions called *operations*. These take a single argument, which is the name of the entity to which the operation is to be applied. DIDO is given no information about the effect the operations may have. When the program determines that an operation should be performed on some entity, the name of the operation and the name of the entity are passed to the domain implementation. This implementation then modifies its current state and returns a list of all the changes that have occurred. The elements of this list are pairs of perceptual properties, the first indicating the

value of a property of a specific entity before the operation was applied and the second indicating the new value of the same property.

3.2 Knowledge Representation

DIDO's knowledge representation scheme serves two purposes: it must enable the program to make predictions about the likely outcome of applying operations to objects, and it must provide the information needed to guide both the experience generator and the representation generator. The need to predict outcomes implies that DIDO must be able to represent partitions of the set of all objects with respect to each possible operation that might be applied to them. Members of each partition should behave in the same way when the operation is applied to them. The need to guide the search processes implies the existence of some form of meta-knowledge that describes the current state of the representation of the domain.

3.2.1 PRACTICAL CONDITIONALS

Both of these purposes are served by what we term *practical conditionals*, which form the core of the representation scheme used by DIDO. A practical conditional is composed of an operation name and a list of outcomes that have been observed to be consequences of applying that operation. Outcomes take the form of changes in the observable features of objects. Associated with each outcome is an estimate of the probability that the outcome will occur. The following is an example of a practical conditional for the action Wave_Wand:

```
(Wave_Wand ( (Type(X, Magician) → Type(X, Dwarf), 0.15)
              (Vitality(X, Alive) → Vitality(X, Dead), 0.10)
              (NIL, 0.75) )
              1.05)
```

This particular practical conditional will be constructed when DIDO has found that the action of waving a wand at an entity has the following effects: 15% of the time it turns the entity from a magician into a dwarf; 10% of the time it kills the entity; and the remaining 75% of the time it has no effect. The figure 1.05 is the associated uncertainty (see below).

The probability estimates are used to guide both the experience generator and the representation generator. The outcome with the highest probability estimate (other than the null outcome) is termed the *ex-*

pected outcome. (In the example above, the transformation of a magician into a dwarf is the expected outcome). Thus a practical conditional contains the knowledge required to predict the consequences of operations, together with a measure of the likelihood that the prediction is correct.

Probability estimates are updated when the operation is applied using a first-order lag,

$$P_{t+1} = P_t + (X_t - P_t)/\tau \quad ,$$

where τ is a time constant (10 in the runs reported later), and X_t is 1 if the outcome occurred and 0 otherwise. This produces a probability estimate that is weighted towards recent events: any single event will eventually make a negligibly small contribution to the estimate. Outcomes whose probability estimates drop below a threshold are eliminated from the practical conditional and the remaining estimates are normalized so that they sum to one. In some circumstances, this results in a practical conditional containing only one outcome whose probability estimate is necessarily one. Such a practical conditional is said to have *fixated*.

For each practical conditional, DIDO computes and stores the Shannon uncertainty function (Shannon & Weaver, 1949)

$$H = - \sum_{i=1}^n p_i \log_2 p_i \quad ,$$

where the sum is taken over the probabilities of all the outcomes. This gives a measure of DIDO's current uncertainty regarding the consequences of the action, thus providing the meta-knowledge used to guide the experience generator. Practical conditionals containing several outcomes with significant probability estimates will have high uncertainty; practical conditionals with only one outcome (fixated practical conditionals) will have zero uncertainty.

3.2.2 THE CLASS NETWORK

DIDO builds a hierarchical network of classes to categorize the objects found in a domain, as shown in Figure 1. In contrast to many learning programs, the hierarchy produced is not restricted to a tree form, but may be any directed acyclic graph determined by the class inclusion relation. Each class has an intension, a current extension, and a set

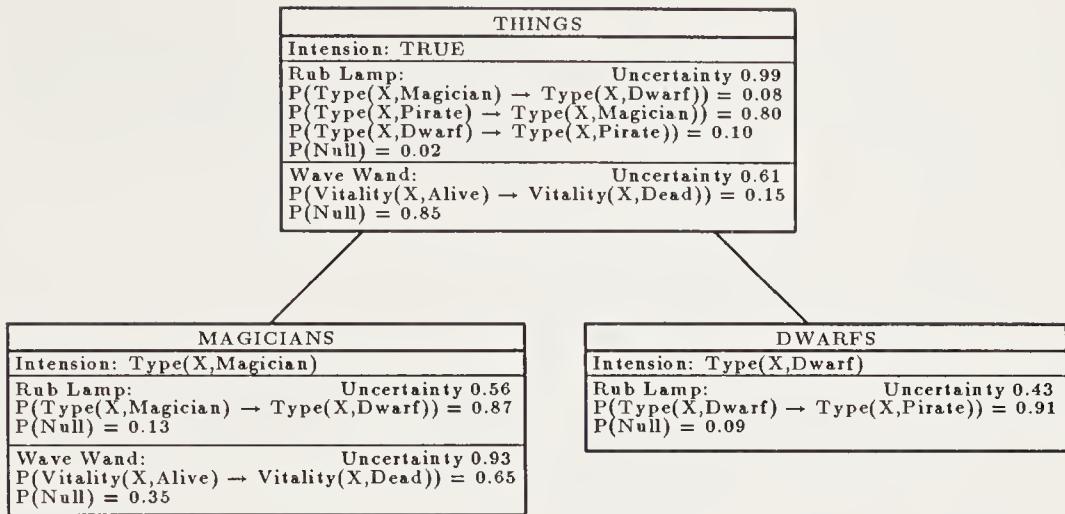


Figure 1. A simple class network.

of practical conditionals. The intension defines a predicate for class membership and takes the form of a conjunction of observable features of objects. The intension need not contain terms that appear in the intensions of the class's superclasses, since these will be inherited. In certain cases this means that there are no new terms in the intension of a class: such classes are given the intension term **TRUE**, which is satisfied by any object. Disjunctive concepts are represented by forming separate classes for each disjunctive term. The extension is a list of all currently observable objects that have satisfied the intension.

The practical conditionals in a class contain outcomes and associated probability estimates based on observations made on instances of that class. They may therefore be used to predict the outcomes of applying operations to arbitrarily chosen instances of the class. In making such a prediction for a given operation, **DIDO** will use the practical conditional for that operation if it is included with the class. However, if it is not included in the class, then it may be inherited from a superclass. Thus, one must explicitly represent the practical conditional for a given operation only if the operation produces effects on instances of the class that are different from those produced on instances of the superclasses.

An object is said to be a *residual instance* of a class with respect to a particular operation if it is not an instance of any subclass of the class that contains a practical conditional for that operation. For example, in Figure 1, a dwarf would be considered a residual instance of the class *Things* with respect to the operation *Wave Wand*, but not with

respect to *Rub Lamp*. Predictions about the outcome of applying an operation to an object are always derived from a practical conditional in a class that includes the object as a residual instance with respect to the operation.

3.3 Search of Experience Space

The method DIDO employs to search experience space, and our reasons for choosing it, are discussed elsewhere (Scott & Vogt, 1983; Scott & Markovitch, 1988, 1989). Here we summarize the method in order to provide a basis for understanding the representation generator, which forms the main topic of this paper.

DIDO selects training examples by first finding the practical conditional that currently has the highest uncertainty value. This will be in some class, C, and include an operation O. The program then chooses instances of the class C that are residual with respect to O, and obtains new training examples by applying O to the chosen instances. Thus DIDO's search of experience space is guided by uncertainty: the instances chosen for further investigation are those about whose behavior the program is currently least certain.

In practice, the effect of this experience selection strategy is to force DIDO to distribute its learning resources fairly across the whole set of available operations. If the uncertainty associated with one operation is higher than any of the others, then that operation will be the focus of learning activity until its uncertainty has been brought down. Hence a typical run of the program comprises a sequence of episodes, in which sets of related examples are investigated as DIDO attempts to lower the current uncertainty peak.

3.4 Search of Representation Space

Ecologists classify living organisms on the basis of the strategies they employ to ensure their reproductive success: *k-strategists* produce few offspring and invest considerable effort in ensuring their survival, while *r-strategists* produce large numbers of offspring but make little or no attempt help them survive.³ The same classification scheme may be ap-

3. The terms derive from the parameters of a logistic equation describing the change in a population, $dn/dt = rn(k - n)/k$, where r is the intrinsic growth rate and k the environment's carrying capacity (MacArthur & Wilson, 1967; Pianka, 1970).

plied to learning systems: k-strategy learning systems are those which make comparatively few inductive steps but invest considerable effort in ensuring that they are likely to be correct. In contrast, an r-strategy learning system makes many inductive leaps but does not spend much computational effort on determining that they are likely to be justified: such a system will necessarily make errors and must therefore be able to discard unsuccessful hypotheses. The candidate elimination algorithm (Mitchell, 1977), ID3 (Quinlan, 1983), and COBWEB (Fisher, 1987) are all k-strategists, since each devotes considerable computational effort towards determining the best modification to make to the current representation. In contrast, PRISM (Langley, 1987), connectionist models (Rumelhart & McClelland, 1986), and genetic algorithms (Holland, 1975) are all r-strategists, because representation changes are generated frequently and each change involves only a small amount of computational effort. Etzioni's (1988) method for improving any learning program can be viewed as treating the program concerned as an r-strategist by adding a filter to discard unsuccessful hypotheses.

DIDO is an r-strategist. Changes to the representation are generated frequently on the basis of a limited amount of evidence and computation. We call these changes *conjectures* to emphasize their tentative nature. As with all r-strategies, the success of this approach depends upon the existence of a winnowing process, termed *retraction*, which eliminates the unsuccessful changes. These processes are described in detail in the next section. Two constraints influenced our choice of an r-strategy. First, the requirement that DIDO always be ready to make use of the evidence accumulated so far suggests that new data should have an immediate influence on the representation, even if such a change cannot be justified statistically. Second, the requirement of being parsimonious in the use of computing resources argued against either engaging in complex computations to determine the best representation change or remembering large numbers of training examples.

3.5 DIDO's Control Structure

DIDO's overall control structure is shown in Table 1. It comprises a loop in which first a round of experiments is performed to collect information about the effects of a particular operation on instances of a particular class, and then an attempt is made to use the information collected to improve the representation. Note that all experiment results

Table 1. DIDO's basic control structure.

```

WHILE uncertainty exists within representation DO
  Find practical conditional with highest uncertainty.
  Identify associated class C and operation O.
  REPEAT
    Randomly select instance of class C that is
      residual with respect to operation O.
    Apply operation O to instance and record outcome.
    Update probability and uncertainty estimates.
    Remove any useless practical conditionals.
    Remove any useless classes.
  UNTIL
    At least two expected outcomes and
      one unexpected outcome have been observed,
    OR no more instances are available,
    OR practical conditional has been removed.
  IF enough outcomes have been recorded,
    THEN modify representation in attempt to reduce uncertainty.
  Discard instance and outcome records.

```

are discarded at the end of each iteration. DIDO does not retain any permanent record of the behavior of instances other than the averages stored in the outcome lists.

4. Conjecture-Driven Search of Representation Space

Initially the program's representation contains a single class, *Things*, whose membership includes every object in the domain. This class is given a practical conditional for each operation in the motor interface. The outcome lists contain the null outcome and a special dummy outcome (which never occurs), each with a probability of 0.5, thus giving all practical conditionals a high initial uncertainty. DIDO therefore begins with a complete representation, since it makes predictions about the application of any operation to any object. This property of completeness is preserved by all the representation modification procedures described below.

4.1 Conjectural Specialization

DIDO's principal method for extending the representation is a discrimination learning procedure termed *conjectural specialization*. Outcomes that confirm the expected outcome are considered *successes*; the rest are classed as *failures*. Conjectural specialization is invoked whenever a round of experiments produces at least two successes and one failure. The objective is to produce a representation with less uncertainty by attempting to construct a subclass which contains a higher proportion of objects that lead to the expected outcome. (Note that this is why the null outcome is never treated as the expected outcome: if it were, DIDO would invest much effort in finding ways to make nothing happen.) If successful, this procedure will also reduce the uncertainty of the practical conditional in the parent class, since instances of the subclass will no longer be residual instances of the parent class.

The program attempts to find an attribute value that can be added to the intension of the parent class to form an intension for a new subclass. This is done using the sample of outcomes obtained during the round of experiments. Typically there will be about five or six such outcomes. The procedure used can be stated as follows:

For each attribute value appearing in the sample:

Determine S, the number of times it appears in successes;

Determine F, the number of times it appears in failures.

Select the attribute value for which $(S - F)$ is largest.

This is a very crude method of finding an attribute value that correlates with the expected outcome, but we have found it to prove effective in practice. Though it may sometimes lead to erroneous conjectures, this is of no lasting significance, as they are rapidly retracted. Note also that, unlike decision tree programs such as ID3, DIDO creates a single subclass based on an attribute value rather than a partition based on all the values of an attribute.

At this point DIDO has conjectured the intension of a subclass that is likely to contain a higher proportion of objects giving the expected outcome. If this class does not already exist (it may have been created to hold a practical conditional for a different operation), the program creates it. If the parent class is its only superclass, the practical conditional from the parent is copied. Otherwise, the practical conditional from the superclass having the highest probability for the expected outcome is copied. This completes the process of conjectural specialization.

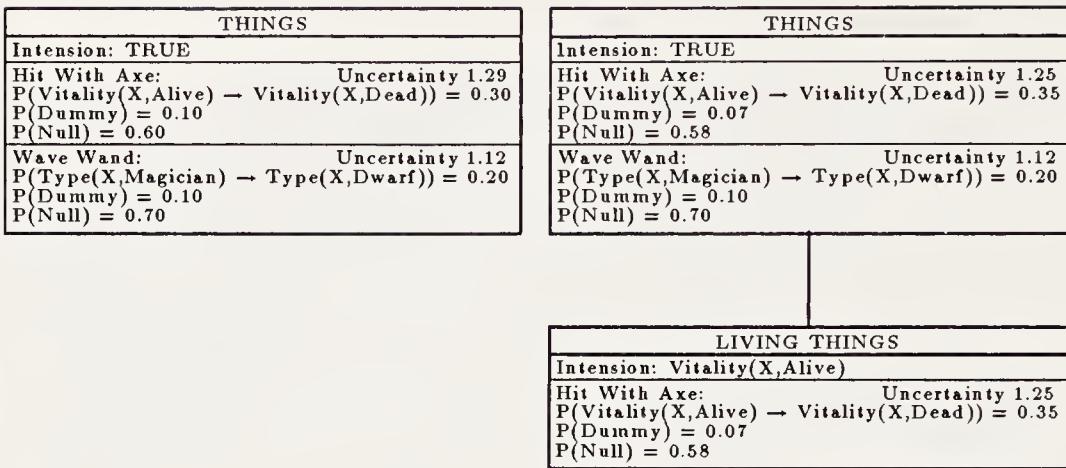


Figure 2. A simple network before (left) and immediately after (right) conjectural specialization for the action *Hit With Axe*.

Figure 2 shows a very simple network before and after conjectural specialization takes place. Initially the most uncertain action is *Hit With Axe* in the class *Things*, and the expected outcome is that the object will be killed. After a few axe blows to various objects, DIDO finds that this outcome is most likely to occur if the objects are initially living. The subclass *Living Things* is therefore constructed, and the practical conditional for *Hit With Axe* in the parent class (which has itself been modified during the round of experiments) is copied to it.

This process, if successful, will eventually lead to a situation in which each practical conditional contains a single fixated outcome; this is DIDO's goal, in which each prediction is made with no uncertainty. Note that conjectures are made on the basis of small samples using crude methods for identifying statistical associations. This approach is fast but necessarily error prone. DIDO therefore requires an efficient method for detecting and removing erroneous conjectures as quickly as possible. This is provided by the program's *retraction* procedures.

4.2 Retraction

A practical conditional serves a useful purpose in the representation if its removal would lead to either different or less certain predictions. If a practical conditional is removed, its role is filled by a practical conditional for the same operation inherited from a superclass. Thus the utility of a given practical conditional can be assessed simply by compar-

ing its outcome list with that of the corresponding practical conditional that would be inherited. This comparison is performed every time the probabilities change, and any practical conditional that is not useful is immediately removed.

This removal process may result in a class that has no practical conditionals. If a class has neither practical conditionals nor subclasses, then it serves no useful purpose. Tests for this situation are performed whenever a practical conditional is removed, and all classes that have become useless are also discarded.

The relationship between conjectural specialization and retraction is best understood by considering DIDO's typical behavior following the creation of an erroneous conjecture. The new conjecture is likely to be amongst the least certain practical conditionals in the representation, because it is a copy of what had been the most uncertain practical conditional. It will therefore soon (typically immediately) become the subject of a round of experiments. If the conjecture is erroneous, then the effect of this experimentation is usually to change the probabilities in the outcome list so that the practical conditional becomes useless and is removed. Hence incorrect conjectures are normally removed promptly. Since refutations are based on small samples, the system can sometimes remove a correct conjecture. Fortunately, this has no lasting consequences, as the conjecture will then be regenerated, because DIDO makes no attempt to avoid repeating conjectures that have been tried unsuccessfully.

4.3 Conjunctive Specialization

Conjectural specialization eliminates conflicting predictions within a single class by driving the representation towards a state in which each practical conditional contains only one outcome. However, there is an alternative way in which conflicting predictions may arise. An object may be a residual instance, with respect to some operation, of two classes, neither of which is a subclass of the other. For example, DIDO may have built a representation containing the classes *Large Things* and *Round Things*, each containing a practical conditional for the operation *Kick*. Previous experience may have led to the conclusions that round things change position when kicked but that large things do not. The first time DIDO encounters a large round object, it will make two incompatible predictions.

This problem could be eliminated by constraining the representation generator so that such conflicts can never occur. For example, DIDO might check for potential conflicts every time a new practical conditional is created or the expected outcome of an existing one changes. This would be computationally expensive, since it involves a search of the whole representation, and the effort invested may be wasted since the potential conflict may never actually arise. In the example discussed above, the conflict was of concern only because DIDO encountered a large round object. Since in all but the most impoverished domains the vast majority of attribute combinations will never occur, many conflicts will arise in theory but not in practice.

DIDO's method for handling such conflicts exploits the fact that many of them will never arise; briefly, the system postpones dealing with them until it encounters an object that leads to conflicting predictions. Whenever an example object is selected, a check is made to see whether another class holds a practical conditional that would lead to a conflicting prediction. Since this involves considering only those classes for which the object is a member, it is not computationally expensive.

When DIDO encounters an object that leads to such a conflict, a procedure termed *conjunctive specialization* is invoked. The intension for the most general common subclass of the two classes concerned is constructed. If this subclass does not already exist, it is created. A practical conditional is then copied down from one of the two parent classes. This eliminates conflicting predictions because the object that gave rise to the conflict will be a residual instance of the subclass and is thus no longer a residual instance of either of the parents. The prediction made may be incorrect, but the normal operation of the representation generator will rectify this.

Figure 3 shows a simple network before and after conjunctive specialization takes place. The original hierarchy generates different predictions for the operation *Wave Wand* when applied to magicians and to things wearing boots. This causes no problem until DIDO encounters a magician wearing boots. The intersection class *Magicians Wearing Boots* is created and the practical conditional for *Wave Wand* is copied from the class *Things Wearing Boots*, because the expected outcome has a higher probability than its counterpart in the class *Magicians*.

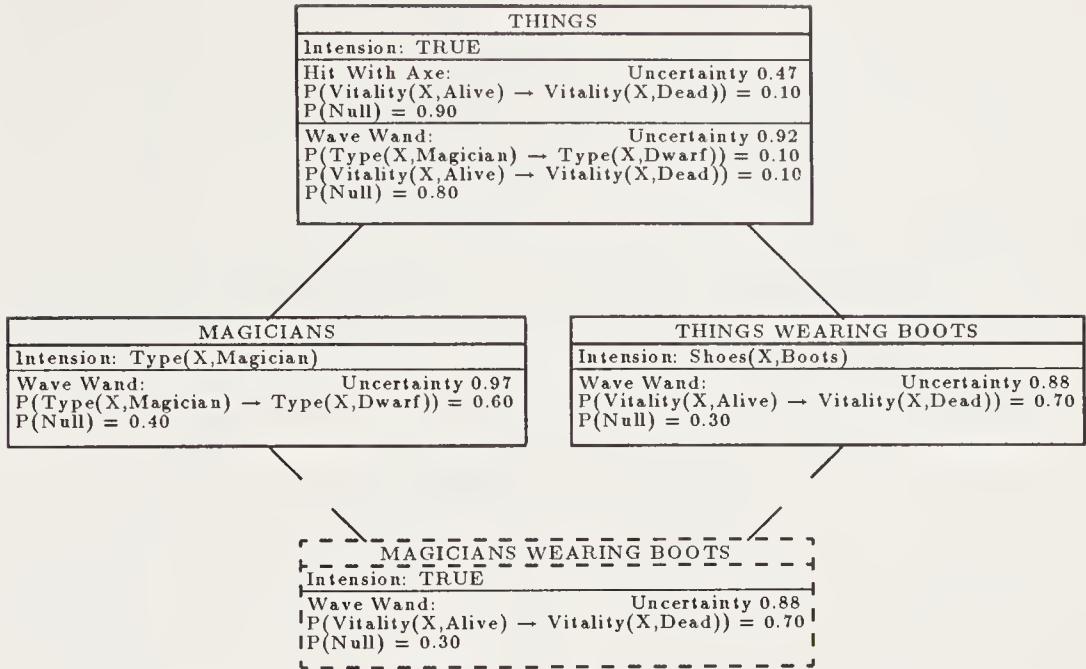


Figure 3. A simple class network immediately before (solid) and after (dashed) conjunctive specialization for the action Wave Wand takes place.

Conjunctive specialization thus complements conjectural specialization by providing a second mechanism for forming classes that are restrictions of existing classes. Both procedures serve to eliminate conflicting predictions from the representation: conjectural specialization attempts to remove the conflicts within classes, while conjunctive specialization removes conflicts between classes. Conjectural specialization does most of the work of discovering new classes because it is invoked much more frequently. In principle, conjunctive specialization is redundant, since, if conflicting predictions were ignored, conjectural specialization would eventually discover the subclasses that it produces. Conjunctive specialization accelerates learning by exploiting the discovery of a conflict as evidence for the need to extend the representation.

4.4 Intensional Generalization

The procedures described so far enable DIDO to discover representations that are correct and complete, but that may not be the most parsimonious possible for a given domain. The representation generator therefore includes further procedures which eliminate various types of redundancy that may arise during the specialization process.

One common form of redundant representation can arise when the program discovers a special case of a more general rule. The result is likely to be a set of rules that deal with each of the special cases rather than a single rule to cover all them. For example, suppose DIDO discovers that both small round things and large round things move when kicked. None of the procedures described so far would enable DIDO to derive the simpler rule that round things move when kicked.

DIDO's representation generator therefore includes a procedure termed *intensional generalization*, which is invoked whenever a practical conditional fixates in some class, F. When this happens, DIDO has discovered a rule it believes to be correct. The program then investigates the possibility that the rule is actually a special case of a more general one by looking for evidence that other parts of the representation seem to be moving towards a similar rule. A search is made for other practical conditionals, containing the same operation and the same expected outcome, which are in classes that are neither subclasses nor superclasses of the class F. If any such practical conditional is found to be in a class, C, whose intension differs by only one term from the intension of F, a generalized intension is formed by deleting the term that differs from the intension of F. This defines a class that is a superclass of both F and C; if it does not already exist, it is created. Finally the relevant practical conditional is copied into the superclass from class C.

This procedure constitutes a conjectured generalization of the rule whose special form has already been discovered. What happens next depends upon whether or not the generalization proves to be valid. If it is valid, then the practical conditional in the generalized superclass will eventually fixate; the corresponding practical conditionals in both C and F will be useless and therefore removed. Thus the general rule will replace the special form originally discovered. However, if the generalization is not valid, the practical conditional in the superclass will either fixate with a different outcome or prove useless and be removed, while the original special rule will remain in place.

Figure 4 shows part of a network before and after intensional generalization has taken place. Initially, the operation *Hit With Axe* has just fixated in the class *Living Dwarfs Wearing Slippers*. A search of other similar classes shows that the same action has the same expected outcome in the class *Living Magicians Wearing Slippers*. DIDO there-

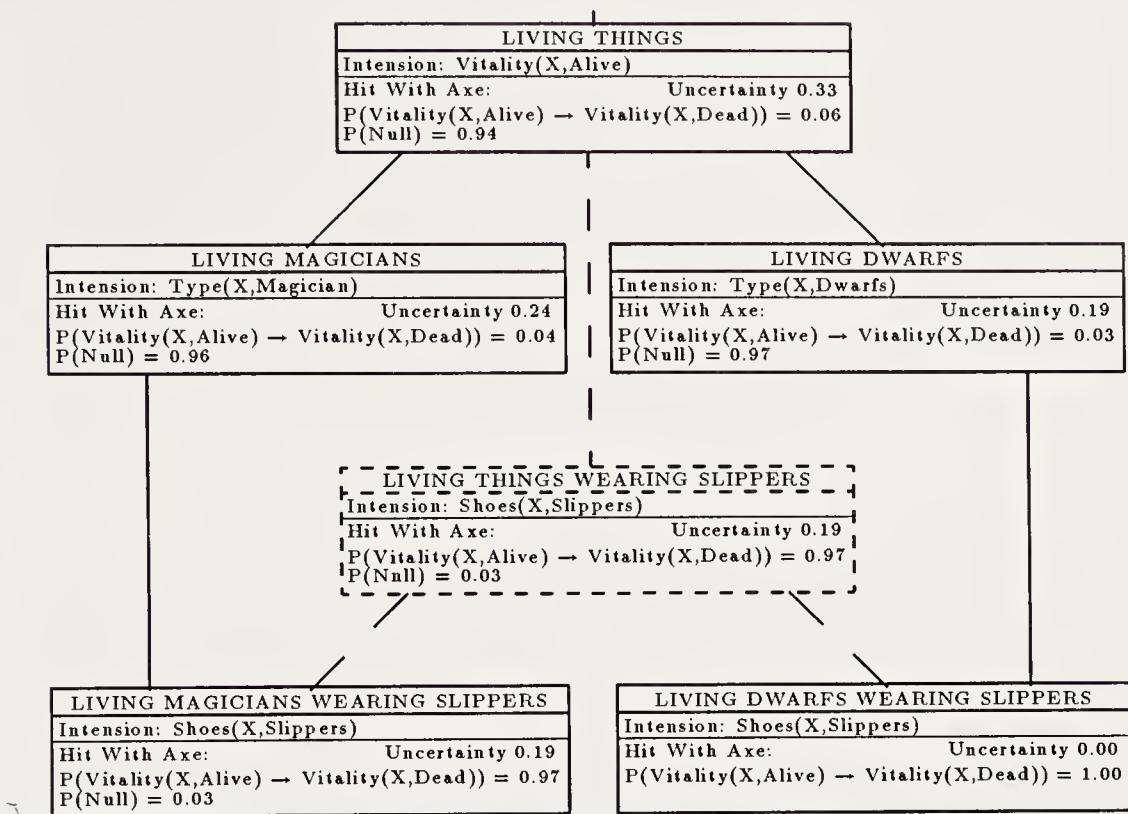


Figure 4. Part of a class network before (solid) and after (dashed) intensional generalization for *Hit With Axe* takes place.

fore constructs the generalization class *Living Things Wearing Slippers* and copies the appropriate practical conditional from the class where fixation has not occurred. If this generalization subsequently proves to be correct, the normal retraction procedures will eliminate all but the two classes *Living Things* and *Living Things Wearing Slippers*; but if it proves to be incorrect, then the class *Living Things Wearing Slippers* will be discarded.

This generalization procedure is rather cautious since it is invoked only when a practical conditional fixates, and generalization is only attempted between two very similar classes. There is scope for experimentation with relaxing both of these constraints. In its present form the generalization procedure has proved useful not only by simplifying representation but also by accelerating learning through eliminating the need to discover further special cases.

4.5 Partitions

DIDO differs from programs such as ID3 in that it does not construct a partition of a class when building a specialization. However, the normal process of conjectural specialization may produce a complete partition of a class with respect to some particular operation. This may constitute a good representation if the outcomes associated with the compartments of the partition are not all the same. Unfortunately, in some circumstances partitions may be constructed that serve no purpose, because members of each compartment of the partition behave in the same way.

4.5.1 USELESS PARTITIONS

Such useless partitions slow down progress towards a fixated representation because they will force DIDO to repeat the same learning process for each compartment of the partition. Hence, it is desirable to eliminate them as early as possible. Partitions can only come into being when a practical conditional is added to a class, so DIDO checks for partition formation whenever this occurs. This process requires knowledge of the full range of values that each attribute can take. In current versions of DIDO, this information is given when the sensory interface is specified, but it would be possible for the program to accumulate this knowledge on the basis of experience with the various perceptual properties.

Once a partition has been found, determining whether it is useless is simply a matter of checking whether all the expected outcomes are the same. Useless partitions are eliminated by removing the practical conditional from each compartment class, and copying the least uncertain one into the partitioned class. Any classes that become useless as a result of these changes are removed by the normal retraction procedure. When a useful partition has been found, DIDO keeps a record of it, because it may subsequently become useless as a result of changes in the expected outcomes of its compartments. If this happens, it is eliminated.

4.5.2 USEFUL PARTITIONS

A partition may remain useful right up to the time when the practical conditionals in all its compartments fixate. This creates another opportunity for simplification of the representation. Because the partition covers all possible instances of the partitioned class and all the out-

comes are certain, the practical conditional can be removed from both the partitioned class itself and all the subclasses of the partitioned class that are not part of the partition.

This procedure, which is triggered when the last compartment of a useful partition fixates, has proved to be very effective in cleaning up the representation during the latter stages of learning. It can be viewed as a process of discarding a number of weaker hypotheses once a stronger one has proven itself. It saves the program a considerable amount of effort in refuting the weaker hypotheses experimentally.

4.6 Residual Specialization

The last component of the representation generator is only included to remove asymmetries from the representation. It does not contribute to correctness, parsimony, or speed of learning. Its purpose can best be understood by considering an example.

Consider a domain containing three types of buttons, colored red, blue, and green, which when pressed produce a squeak, a honk, and a clang, respectively. Suppose DIDO has progressed to the stage where the subclasses for red and blue buttons have been created to hold practical conditionals for the operation press. At this point, all the residual instances of the more general buttons class will necessarily be green. Hence the program has no need to construct a subclass for green buttons since their behavior will be correctly predicted by the practical conditional for the buttons class.

A practical conditional predicts the behavior of the residual instances of a class, so when all but one possible outcome have been covered by subclasses, the practical conditional is left predicting the remaining outcome. Since DIDO has a bias that avoids constructing classes to predict the null outcome, that is usually the one remaining. When, as in the example, the null outcome never arises, an arbitrary non-null outcome is the one remaining and the resulting representation does not reflect the symmetry of the domain. Although this does not affect the correctness or the parsimony of the final representation, we felt it was aesthetically unsatisfactory.

DIDO therefore includes a procedure, which we call *residual specialization*, that constructs the additional subclass needed to make the representation symmetric. Whenever a round of experiments produces a

Table 2. Conditions that determine when DIDO uses the various components of its representation generator.

PROCEDURE	CONSIDERED WHENEVER	DONE WHENEVER
<i>Conjectural specialization</i>	Round of experiments ends.	At least two successes and one failure obtained.
<i>Retraction</i>	Estimates of probability change.	Practical conditionals or classes become useless.
<i>Conjunctive specialization</i>	Operation applied to an object.	Two practical conditionals make conflicting predictions about the same object.
<i>Intensional generalization</i>	Practical conditional fixates.	Similar class has practical conditional with same expected outcome.
<i>Useless partition removal</i>	Practical conditional added to class, or expected outcome of such conditional in useful partition changes.	Useless partition has arisen.
<i>Fixated partition procedure</i>	Practical conditional in useful partition fixates.	All practical conditionals in useful partition fixated.
<i>Residual specialization</i>	All outcomes of round of experiments successes.	Adding subclass would create useful partition.

run of successes, there is evidence that all the residual instances of the class have the same non-null outcome. A check is made to determine whether the addition of a single extra subclass would result in a useful partition. If so, the subclass is added.

4.7 Summary of Learning Procedures

Before moving on, we should summarize the various procedures for representation modification that we introduced in this section. Table 2 lists the seven DIDO procedures, along with the conditions under which each is considered and applied. In addition, Table 3 describes each procedure in terms of its frequency of use and its effects on prediction accuracy, simplicity of the class hierarchy, and learning rate.

Table 3. Consequences of applying the various components of DIDO's representation generator.

PROCEDURE	FREQUENCY OF USE	IMPROVES PREDICTION ACCURACY	SIMPLIFIES CLASS HIERARCHY	INCREASES LEARNING RATE
<i>Conjectural Specialization</i>	HIGH	YES	No	No
<i>Retraction</i>	HIGH	No	YES	No
<i>Conjunctive Specialization</i>	Low	YES	No	YES
<i>Intensional Generalization</i>	Low	No	No	YES
<i>Useless Partition Removal</i>	Low	No	YES	YES
<i>Fixated Partition Procedure</i>	Low	No	YES	YES
<i>Residual Specialization</i>	Low	No	No	No

Naturally, some readers will wonder whether all seven learning procedures are needed for successful operation. In our experience, DIDO's representation at any time during learning largely reflects a balance between the construction of new classes by conjectural specialization and their removal by retraction. The other modification procedures are less important and are only invoked occasionally, their main role being to accelerate the convergence on a correct representation of the domain.

5. Experimental Evaluation of DIDO

In this section we describe a number of results that we have obtained running DIDO in a variety of problem domains. First we show the system's progress during a run in a domain based on computer games. After this, we examine DIDO's behavior in domains that involve different levels of complexity, noise, and change.

Table 4. Perceptual properties and operations in the Adventure domain.

PROPERTY	POSSIBLE VALUES
TYPE	DWARF, MAGICIAN, PIRATE
SIZE	SMALL, MEDIUM, LARGE
VITALITY	ALIVE, DEAD
COAT COLOR	RED, BLUE, GREEN, YELLOW, BLACK, WHITE, GOLD
SHOES	LEATHER, BOOTS, SLIPPERS, SANDALS, NONE
OPERATION	CONSEQUENCES
HIT WITH AXE	KILLS ANY LIVING THING WEARING SLIPPERS.
WAVE WAND	KILLS ANY LIVING THING WEARING BOOTS; OTHERWISE TURNS ANY MAGICIAN INTO A DWARF.
RUB LAMP	TURNS ANY DWARF INTO A PIRATE; TURNS ANY PIRATE INTO A MAGICIAN; TURNS ANY MAGICIAN INTO A DWARF.

5.1 A Typical Run

Our first problem domain is an artificial one, very loosely based on computer games like Adventure. It includes five perceptual properties and three operations, and Table 4 summarizes the essential features. Figure 5 shows the final representation, reached after 417 experiments had been performed, when all practical conditionals had fixated. This is a complete, consistent, and correct representation of the domain, which is optimal in that no representation with fewer classes is possible. The only practical conditional for *Hit With Axe* with a non-null outcome appears in the class *Living Things Wearing Slippers*: all other classes inherit the practical conditional with a null outcome from the class *Things*. No practical conditional for *Rub Lamp* appears in the class *Things* because the classes *Dwarfs*, *Pirates*, and *Magicians* form a useful partition for this operation. The greater complexity of the rule for *Wave Wand* has necessitated the construction of the intersection class *Living Magicians Wearing Boots*, while the class *Living Things*, which holds no practical conditionals, is retained because it has subclasses.

Those of us who build machine learning systems have one major advantage over psychologists studying human or animal learning: we can

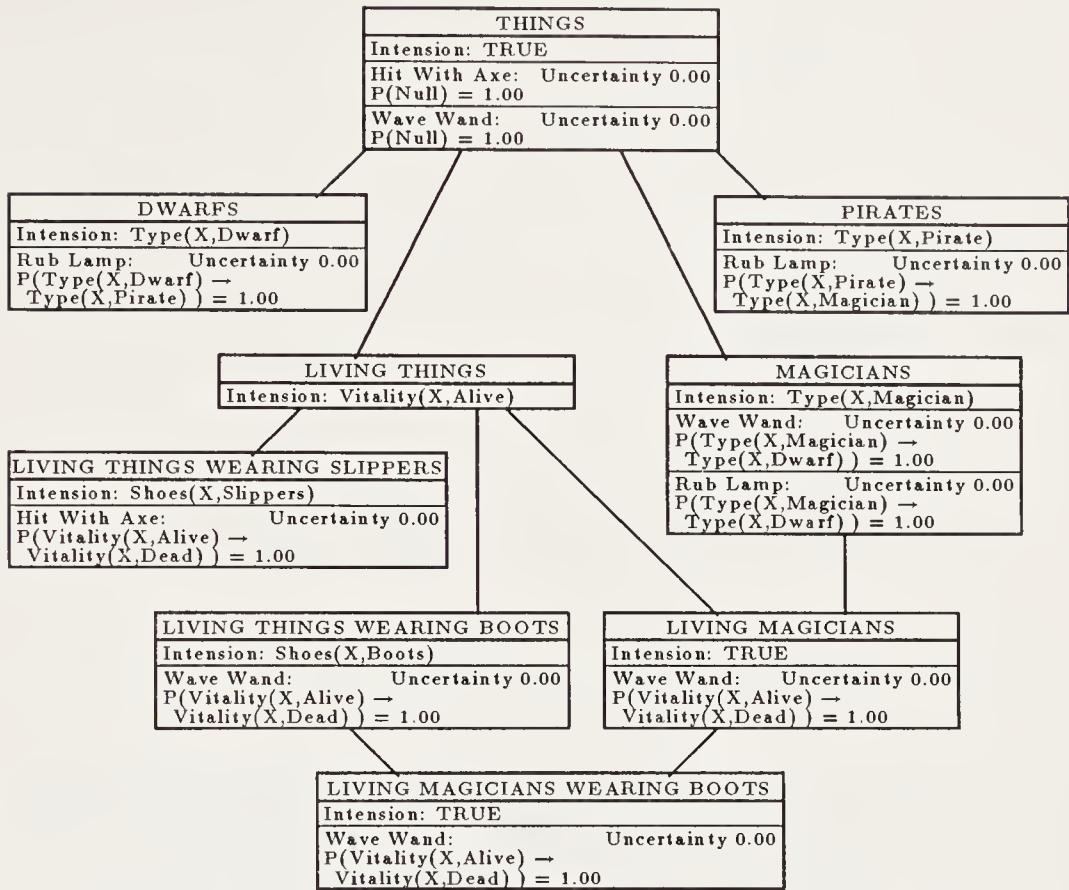


Figure 5. Final representation produced by DIDO for the Adventure domain.

suspend learning and evaluate performance without the evaluation tasks themselves contributing to further learning. We employed this technique to measure DIDO's learning progress by setting *exams* at regular intervals during learning. Exams are administered by turning the learning off and randomly generating 100 objects in the domain. The program then predicts the outcomes of applying randomly selected operations to each object, and the percentage of correct predictions is the exam score. This approach has two advantages over the alternative of measuring the error rate during learning itself: it is unaffected by biases due to the example selection strategy employed, and it enables performance measures to be based on large samples.

Figure 6 shows the exam scores produced as DIDO learned the domain described above. There is steady progress towards a 100% exam score during the first 120 experiments, as DIDO rapidly learns to predict correctly the effect of all three operations on any object. There then

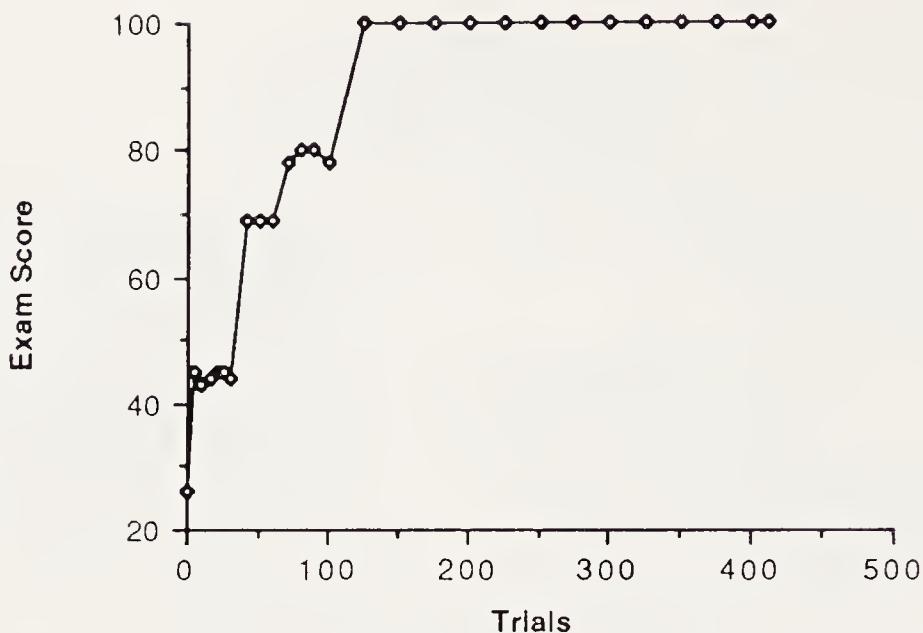


Figure 6. Exam scores during learning in the Adventure domain.

follows a period of nearly 300 experiments before all the practical conditionals have fixated. During this period the system is confirming and refining the representation conjectured during the initial period. Learning ceases only when the program is certain about all aspects of the representation. DIDO thus has two criteria for completion of learning: 100% exam score and fixation. This behavior, which is characteristic of all the program's runs, is a clear demonstration of DIDO's ability to make effective use of tentative conclusions.

Figure 7 shows that the number of classes in the representation rises steadily during most of learning. In other runs, the number rises somewhat above the final total but the number always remains small. Since DIDO has no permanent memory for training examples, this result demonstrates that the program is parsimonious in its use of storage.

5.2 Classification Tasks

Such tests demonstrate that DIDO can learn a set of rules rapidly, with parsimonious use of both training examples and resources, but they provide no basis for comparison with alternative methods of machine learning. This presents us with a difficulty, since few programs have been written that attempt to perform exploratory learning. In contrast, there is a large literature describing programs that learn from sets of

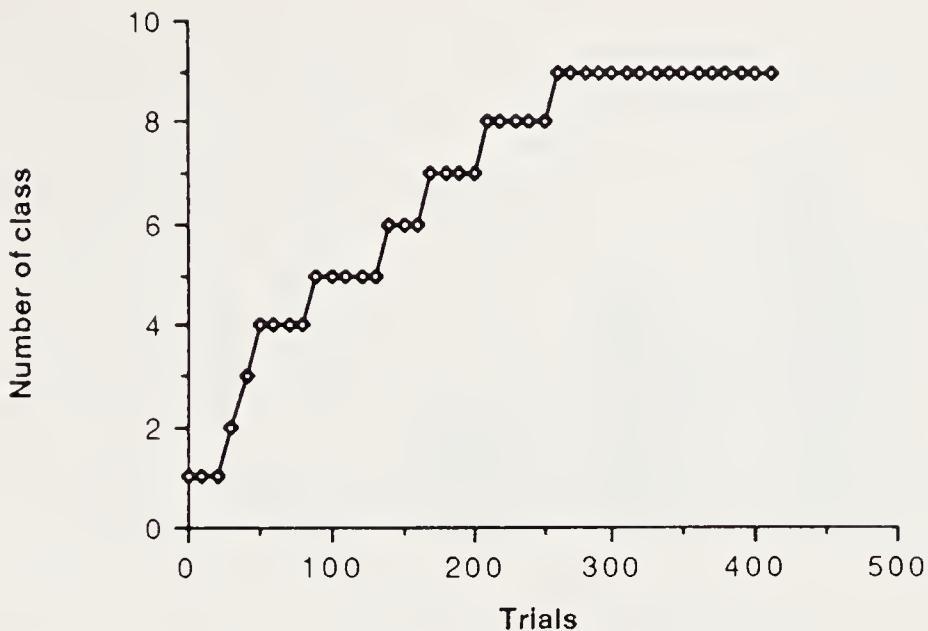


Figure 7. Class count during learning of Adventure domain.

classified examples. Fortunately, it is easy to devise a problem domain such that DIDO performs an equivalent task.

In such a domain DIDO is given a single operation that consists of pointing at an object and asking its category. The category name, supplied by an external oracle, is the outcome of the action. DIDO's goal is thus to predict the category name of any object that may be encountered. This task is more complex than concept learning, since it is DIDO rather than the external oracle that selects the examples to be classified, but it is sufficiently similar to provide a basis for comparison with other learning methods.

5.2.1 METHODOLOGY

In recent years there has been a growing tendency to evaluate concept learning programs by running them on widely available data sets taken from 'real world' problems. This permits comparisons between different programs run on the same data set and affords assurance that the data set has not been chosen to suit the program. However, it suffers from two serious disadvantages as a technique for systematically evaluating the capabilities of particular programs: it is hard to assess the difficulty of finding rules to describe a particular data set, and, since each data set is idiosyncratic, one cannot observe the behavior of a learning procedure

over a sample drawn from a population of similar learning tasks. This latter disadvantage is the more serious one. Our experience, both with DIDO and with reimplementations of other systems, suggests that there can be a very high variance in learning speed for very similar concepts.⁴

In such circumstances it is unsound to attempt to determine the learning speed based on a single run. Enough similar runs must be made to produce a standard error that is very much less than the mean. In practice this means that as many as 20 runs are needed to provide reliable estimates of learning speeds. This requires a population of very similar learning tasks from which the data sets for the runs may be drawn.

We have therefore used a technique in which large numbers of similar learning tasks are produced by a task-generating procedure that requires a specification of the objects in the domain and the type of learning task required. Object specifications consist of two numbers: the number of attributes per object and the number of alternative values per attribute. Task specifications comprise the number of attributes relevant to the concept and the relations among them, which may be conjunctive or disjunctive. Given this information, one may randomly generate many similar tasks of known complexity. We used this method to generate the problem domains in the experiments described below.

The task generator is also used to generate a population of objects from which DIDO will select training examples. The simplest approach is to generate random examples from the object specifications. However, this has the disadvantage that positive examples of concepts involving the conjunction of many terms will be extremely scarce. We therefore arranged that the population from which DIDO selects training examples should contain equal numbers of positive and negative examples.

5.2.2 RESULTS FOR CONJUNCTIVE AND DISJUNCTIVE CONCEPTS

Figures 8 and 9 show the number of examples needed to learn conjunctive and disjunctive concepts of increasing complexity. In all cases there were five attributes, each of which could have one of three possible values. The number of relevant attributes was varied to increase the difficulty of the learning task. Figure 8 shows the mean number of

4. For example, in a series of 50 runs of the candidate elimination algorithm (Mitchell, 1977) learning different three-term conjunctive concepts from randomly generated training examples, the mean number of training examples required was 207, but the standard deviation was 108 and the standard error was 15.

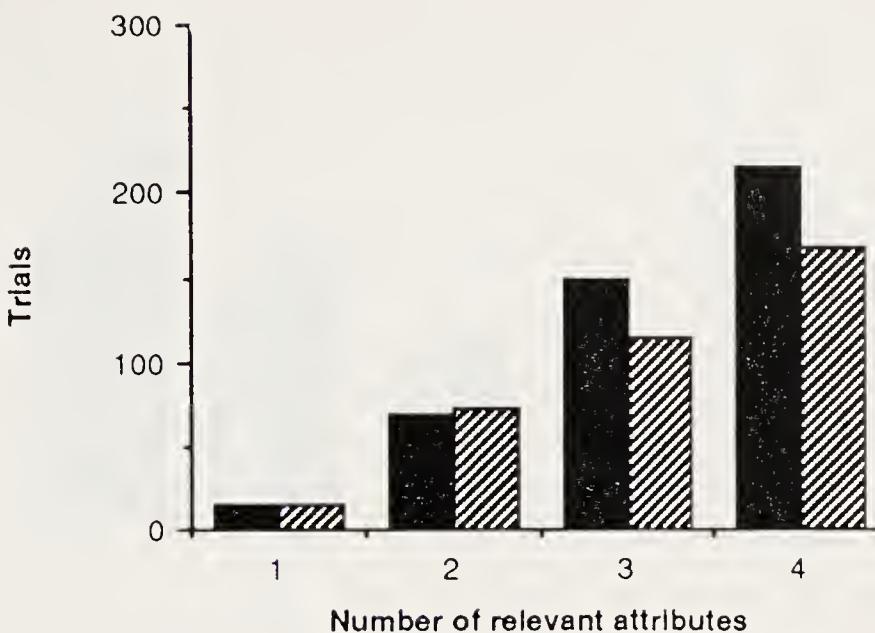


Figure 8. Number of training examples required to reach 100% exam score for conjunctive and disjunctive concepts.

training examples needed to reach the 100% exam score criterion, while Figure 9 shows the number of examples needed to reach fixation. All the results are averaged over 20 runs.

Simple concepts are learned rapidly, and the number of examples required increases linearly with the complexity of the target concept over the range considered. Simple disjunctions and conjunctions require similar learning times but more complex disjunctions require more training examples than comparably complex conjunctions. These results are in marked contrast to those reported for PRISM (Langley, 1987), which appears to find conjunctions very much more difficult than disjunctions. The results for disjunctions are similar, although DIDO appears faster at the simpler concepts while PRISM appears faster at the more complex concepts. However, Langley's results for conjunctive concepts are markedly nonlinear, showing an apparently exponential increase in the required number of training examples as the number of relevant attributes is increased. Consequently PRISM requires nearly five times as many examples to learn a four-term conjunctive concept. At least part of this difference may be due to DIDO's experience generator, which ensures that training examples continue to be informative, even during the later stages of learning when the representation is almost correct.

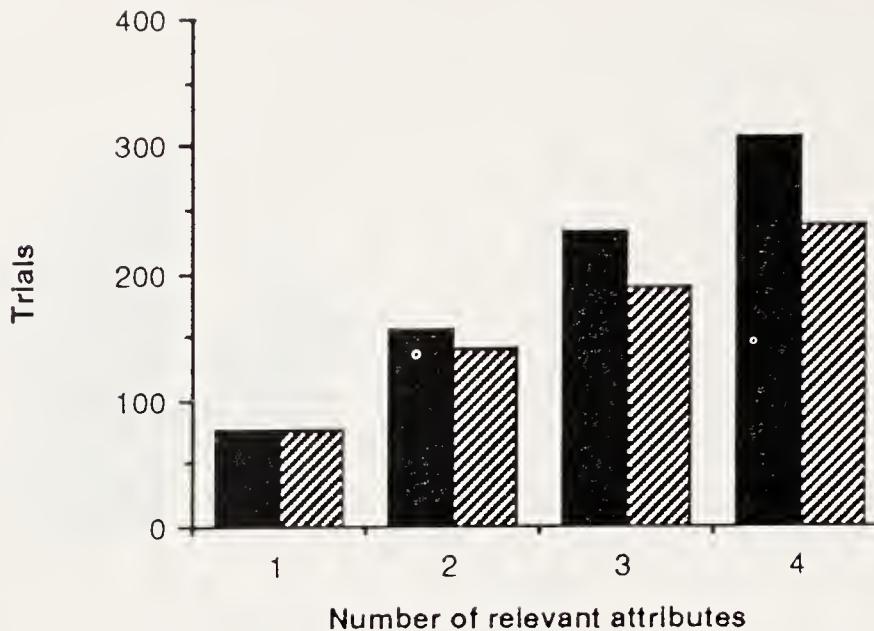


Figure 9. Number of training examples needed to reach fixation for conjunctive and disjunctive concepts.

5.2.3 LIMITATIONS

Although DIDO's representation scheme is capable of representing any Boolean concept, there are some concepts that it does not learn effectively. Its performance with two-term exclusive disjunctions is erratic and we have had no success in getting the program to handle the six-bit multiplexor problem (Wilson, 1987). In this respect DIDO's performance is inferior to that of decision tree learning systems (Quinlan, 1988; Utgoff, 1989). This limitation, which is shared with many machine learning algorithms, arises from the monothetic method of specialization (Fisher & Chan, *in press*), in which only single attributes are considered. Thus, an obvious solution is to employ a polythetic method in which combinations of attributes are considered. Unfortunately, this is computationally expensive and so we are currently investigating the possibility of a mixed strategy in which DIDO only makes polythetic conjectures when progress has ceased using the standard monothetic method.

5.3 Handling Noise

To test DIDO's behavior in non-deterministic domains, we carried out an experiment with classification tasks involving different levels of noise. Various authors have defined noise level in different ways (e.g., Langley,

Table 5. Learning conjunctive concepts in presence of noise. Final exam scores and uncertainties are means of 20 trials terminated after 1000 examples if not already fixated.

NOISE LEVEL	FINAL EXAM SCORE	FINAL UNCERTAINTY	INHERENT DOMAIN UNCERTAINTY
0%	100%	0.00	0.00
5%	100%	0.00	0.29
10%	100%	0.03	0.47
15%	100%	0.42	0.61
20%	96%	0.69	0.72
25%	93%	0.82	0.81
30%	89%	0.89	0.88

1987; Quinlan, 1986). In our experiments, the noise level is simply the probability that the classification of an example will be inverted. This is equivalent to combining both positive and negative noise following Langley's (1987) definition. Table 5 shows the results obtained when DIDO attempted to learn a conjunctive concept with two relevant attributes in the presence of various levels of noise. The exams used to evaluate learning were all noise free.

The table shows that the program's response to low levels of noise is qualitatively different from its response to higher levels. When the noise level is below 15%, DIDO builds a representation that is the best deterministic approximation of the problem domain. Consequently, the system achieves exam scores of 100% and very low uncertainty levels. Had the exams contained noise, the mistake rate during exams would have equaled the noise level. Thus, DIDO ignores low levels of noise, although learning speed is reduced as the level rises: it takes almost twice as long to reach 100% exam scores in the presence of 10% noise.

In contrast, when the noise level is higher than 15%, the program is unable to ignore it, and therefore attempts to build a representation that accounts for all the training examples. Since no such representation exists, this attempt is unsuccessful. Consequently, mistakes are made in the exams, the representation contains a significant amount of uncertainty, fixation does not take place, and the search for a good representation continues indefinitely as DIDO vainly attempts to discover rules

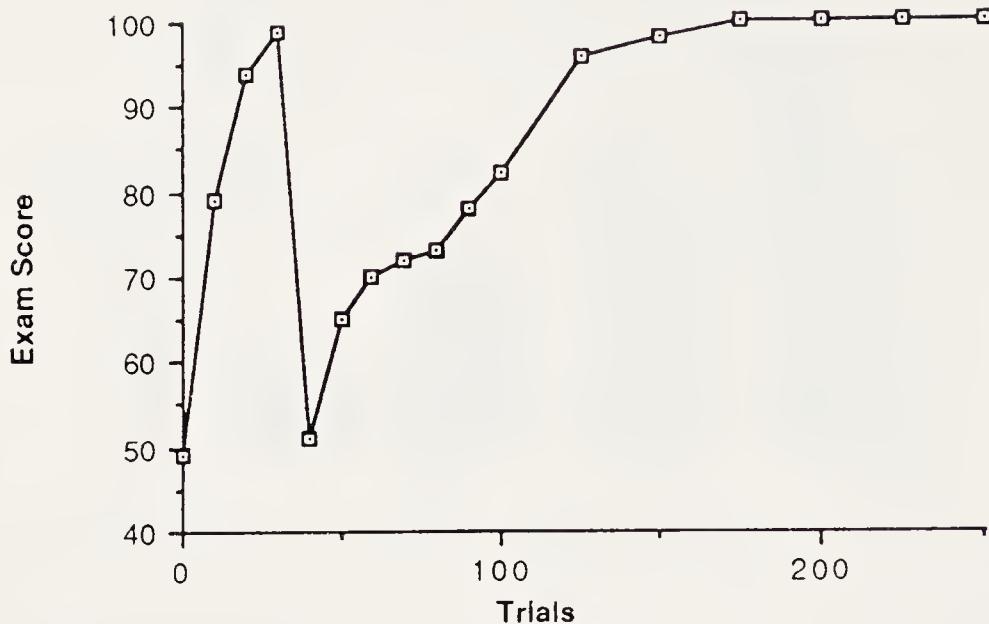


Figure 10. Exam scores for Dido during a run in which the concept changed after 40 trials.

to account for the aberrant training examples. The uncertainty remaining in the representation after learning directly reflects the amount of noise present. The noise level may itself be expressed as an uncertainty measure by applying the Shannon formula to the probability of noise occurring. The fourth column of Table 5 shows this inherent uncertainty of the domain. At higher levels of noise, Dido builds a representation whose residual uncertainty reflects that of the domain.

Clearly, Dido's behavior in the face of noisy training examples is robust. Low levels of noise have no effect beyond a small increase in learning time, and even a noise level as high as 30% leads to a representation that produces the correct response almost 90% of the time. Both methods for handling noise, ignoring failed predictions and attempting to account for them, are reasonable. The more appropriate strategy ultimately depends on the learning application. The transition between the two methods, which occurs at about 15% noise in our experiments, reflects the time constant of the exponential lag that Dido uses to estimate probabilities. Reducing this constant would cause the program to ignore higher noise levels, but would also lead to fixation of incorrect representations because of chance biases in the training examples.

5.4 Handling Change

Another type of inconsistency that may occur arises when the domain itself changes so that a new set of rules is required to describe its behavior correctly. Langley (1987) has described how PRISM successfully adapts when the relevant attribute of a concept is changed during learning. We have replicated this experiment with DIDO and obtained similar results, which we present in Figure 10. The major difference is that the learning curve during the second phase of the experiment is shallower for DIDO than for PRISM. This is because DIDO deliberately seeks out training examples to eliminate the now incorrect concept acquired during the first phase, whereas PRISM ceases modifying the incorrect rule once it has been weakened enough to prevent its firing. Schlimmer and Granger (1986) report similar studies with their STAGGER system, as do Martin and Billman (this volume) in their work on concept formation.

6. Conclusions

The implementation of DIDO has tested solutions to three of the problems that arise in exploratory learning. The first problem concerns the absence of an external agent to define the set of categories to be learned. DIDO's solution is to partition the domain into classes of objects whose behavior in response to actions is equivalent. Thus the representation built is inherently 'DIDOCentric' and provides the knowledge needed to plan sequences of actions to solve problems.

The second problem, which also arises because of the absence of an external agent, is the need for an evaluation of the quality of a representation. DIDO's solution is to attempt to construct a representation with no uncertainty. Since uncertainty arises as a result of experience, and since DIDO's experience generator deliberately seeks experiences likely to increase the uncertainty, this leads to a representation whose predictions are both unique and consistent with the behavior of the domain. In fact, the operation of DIDO can be viewed as a struggle between the experience generator, which tries to increase the uncertainty in the representation, and the representation generator, which tries to minimize it. This is a struggle that the representation generator wins in all but very noisy domains.

The third problem can arise in any form of inductive learning. The program is required to be ready to solve problems at any time, using

what evidence has already been acquired, and must achieve this goal in a manner that is economical in the use of resources. In other words, the system must maintain a *complete* representation learned in an *incremental* manner. DIDO's solution is to adopt what we term an r-strategy: many hypotheses are conjectured on the basis of little evidence or computational effort; those that are sound survive, while those that are not are retracted.

We consider that the experiments reported in this chapter show that DIDO's solutions to these problems are effective. Direct comparison with other work is difficult because we are not aware of any programs that attempt to tackle the same combination of problems. Related work has already been discussed in the appropriate context. We make no claim to have found the best solutions to the problems DIDO faces or that any solutions will be optimal for all types of inductive exploratory learning. Much work remains to be done in this area. The work on representation generation reported here, along with the accounts of experience generation given elsewhere (Scott & Markovitch, 1988, 1989), are perhaps best regarded as exploratory learning about exploratory learning.

Acknowledgements

The work reported in this paper was supported by NSF Grant No. MCS-8203956. A donation of computing equipment to the University of Michigan by the Xerox Corporation made the implementation possible.

References

- Benjamin, D. P. (Ed.). (1990). *Change of representation and inductive bias*. Hingham, MA: Kluwer.
- Dietterich, T. G., & Michalski, R. S. (1981). Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods. *Artificial Intelligence*, 16, 257-294.
- Etzioni, O. (1988). Hypothesis filtering: A practical approach to reliable learning. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 416-429). Ann Arbor: Morgan Kaufmann.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139-172.
- Fisher, D.H., & Chan, P.K.(1990). Statistical guidance in symbolic learning. *Annals of Mathematics and Artificial Intelligence*, 2, 135-148.

- Gross, K. P. (1988). Incremental multiple concept learning using experiments. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 65–72). San Mateo, CA: Morgan Kaufmann.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
- Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.
- Lenat, D. B. (1982). AM: Discovery in mathematics as heuristic search. In R. Davis & D. B. Lenat (Eds.), *Knowledge-based systems in artificial intelligence*. New York: McGraw-Hill.
- MacArthur, R. H., & Wilson, E. O. (1967). *The theory of island biogeography*. Princeton, NJ: Princeton University Press.
- Mitchell, T. M. (1977). Version spaces: A candidate elimination approach to rule learning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 305–310). Cambridge, MA: Morgan Kaufmann.
- Mitchell, T. M. (1979). An analysis of generalization as a search problem. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*. Tokyo, Japan: Morgan Kaufmann.
- Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18, 203–226.
- Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Pianka, E. R. (1970). On r- and k-selection. *American Naturalist*, 104, 592–597.
- Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Quinlan, J. R. (1986). The effect of noise on concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.

- Quinlan, J. R. (1988). An empirical comparison of genetic and decision-tree classifiers. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 135-141). Ann Arbor: Morgan Kaufmann.
- Rumelhart, D. E., & McClelland, J. L. (Eds.). (1986). *Parallel distributed processing*. Cambridge, MA: MIT Press.
- Schlumper, J. C. & Granger, R. H. (1986). Incremental learning from noisy data. *Machine Learning*, 1, 317-354.
- Scott, P. D., & Markovitch, S. (1988). *Learning through curiosity and conjecture* (Tech. Rep. No. 030988). Ann Arbor, MI: Center for Machine Intelligence.
- Scott, P. D., & Markovitch, S. (1989). Learning novel domains through curiosity and conjecture. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 669-674). Detroit, MI: Morgan Kaufmann.
- Scott, P. D., & Vogt, R. C. (1983). Knowledge oriented learning. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. Karlsruhe, Germany: Morgan Kaufmann.
- Shalin, V. L., Wisniewski, E. J., Levi, K. R., & Scott, P. D. (1987). A formal analysis of machine learning systems for knowledge acquisition. *Proceedings of the Second Knowledge Acquisition for Knowledge Based Systems Workshop*. Banff, Alberta, Canada.
- Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of communication*. Urbana, IL: University of Illinois Press.
- Shen, W. M. (1990). Complementary discrimination learning: A duality between generalization and discrimination. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 834-839). Cambridge, MA: AAAI Press.
- Simon, H. A., & Lea, G. (1974). Problem solving and rule induction: A unified view. In L. Gregg (Ed.), *Knowledge and cognition*. Hillsdale, NJ: Lawrence Erlbaum.
- Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning*, 4, 161-186.
- Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning*, 2, 199-288.
- Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.

CHAPTER 15

A Computational Account of Children's Learning About Number Conservation

TONY SIMON

ALLEN NEWELL

DAVID KLAHR

1. Introduction

This chapter focuses on the conceptual domain known in developmental psychology as “conservation of quantity”. Acquisition of quantity conservation constitutes a fundamental part of cognitive development, and hundreds of studies have been published in which children’s “naturally acquired” conservation concepts are assessed, or in which attempts are made to teach such concepts to children. We review some of the major regularities emerging from this literature, and then we describe one particular training study in detail. Following that, we describe Q-SOAR, a computational model that simulates the acquisition of number-conservation knowledge.

The particular study that we examine is Gelman's (1982) training of three and four year old children, and we use Q-SOAR to model the learning behavior exhibited by children in this study. This model is constrained by a unified theory of cognition, which is the SOAR architecture (Lewis et al., 1990; Newell, 1990). The version of Q-SOAR to be described in this chapter accounts only for the learning that occurs in Gelman's study. However, we will suggest that chunking, SOAR's learning mechanism, which is the only learning process included in Q-SOAR, could account for the acquisition of all conservation knowledge.

Our study of conservation learning and human development can be viewed as an instance of research in concept formation. The nature of knowledge acquisition in cognitive development is consistent with the concerns of work on incremental processing. In addition, the feedback that is naturally available during development rarely has the explicit nature that is assumed in supervised learning models (Fisher & Pazzani, Chapter 1, this volume). In fact, studies of conservation training (such as Gelman's) demonstrate that more explicit forms of feedback considerably accelerate the acquisition of conservation concepts. However, even under training conditions, the type of learning that is discussed here is in line with theory-driven approaches to concept formation (Fisher & Pazzani, Chapter 6, this volume), notably the construction of *ad hoc* concepts (Barsalou, 1983) and *explanation-based* methods (Mooney, this volume). We now turn our attention to conservation learning, and then to the particular system that we have developed to model it.

2. The Phenomenon of Conservation

A central tenet of Piagetian theory (Piaget, 1952, 1970) is that the acquisition of conservation knowledge is a crucial step in the child's development of mature conceptual capabilities. Piaget (1968, p. 978) defines conservation as follows:

We call "conservation" (and this is generally accepted) the invariance of a characteristic despite *transformations* of the object or of a collection of objects possessing this characteristic. Concerning number, a collection of objects "conserves" its number when the shape or disposition of the collection is modified, or when it is partitioned into subsets.

Children's knowledge about the effects of transformations must be empirically derived in the first instance because all transformations have different effects on different physical dimensions of the transformed material. For example, whether or not the *pouring* transformation conserves quantity depends on what is poured and what is measured:

If we pour a little sugar into red sugar water, we do not change temperature, amount, height, width, or redness, but we increase sweetness. If we add more of an identical concentration, we do not change temperature, redness or sweetness; however the

amount increases, as does liquid height, but not width (in a rigid container). On the other hand, if we add water, we increase two extensive quantities (amount, liquid height), reduce two intensive quantities (redness, sweetness), and leave one unchanged (temperature) (Klahr, 1982, pp. 68-69).

Therefore, a central component of what must be learned, either in training studies or "naturally acquired" by the child outside the laboratory, are the linkages between transformational attributes and their dimensional effects as measured in a wide variety of contexts.

The centrality of conservation concepts to most theories of cognitive development has produced a vast database of empirical results. Nevertheless, a computational model that can account for the regularities has yet to be fully specified. There have been both structural and processing accounts of the knowledge used by a child who "has" conservation, as well as global characterizations of the acquisition of that knowledge, such as Piaget's assimilation and accommodation processes, Klahr and Wallace's (1976) time-line processing, and Halford's (1982) levels of cognitive systems. However, neither these nor any other accounts have completely stated a set of operations and their interaction with a specified learning mechanism and shown this to produce the pattern of behavior observed in children acquiring conservation knowledge.

Q-SOAR is a model of the acquisition of conservation knowledge designed to meet several desiderata for computational models of developmental phenomena:

1. Such models should be based on a principled cognitive architecture, rather than as a set of arbitrary and ad hoc mechanisms. For Q-SOAR, the architecture is SOAR, to be described in Section 5.
2. Computational models should be constrained by the general regularities in the relevant empirical literature. There are a number of such regularities, i.e., findings that are consistently reported and for which there is little or no disconfirming evidence.
 - (a) Young children in the three to four year age range can consistently obtain accurate specific quantitative values for small sets of objects (up to four) (cf. Fuson, 1988).
 - (b) Young children in this age range cannot consistently obtain accurate specific quantitative values for large sets of objects (more than five or six) (cf. Fuson, 1988).

- (c) Children from three years of age are able to correctly produce a cardinal value for three or four items in under one second by the process of subitizing¹ (Campbell, Cooper, & Blevins-Knabe, 1988; Chi & Klahr, 1975; Svenson & Sjoberg, 1983).
 - (d) Subitizing has a limit of four items in these children (Campbell et al., 1988; Chi & Klahr, 1975; Svenson & Sjoberg, 1983).
 - (e) Children who have not fully acquired conservation knowledge can still correctly answer conservation questions when they can obtain a specific quantitative value for the objects concerned (cf. Siegler, 1981).
 - (f) Children who have not fully acquired conservation knowledge do not correctly answer the conservation question when they cannot obtain a specific quantitative value for objects concerned (cf. Halford & Boyle, 1985).
 - (g) Young children can respond correctly and provide explanations for tests of conservation of quantity when discrete materials are used (such as counters or cookies) before they can do this on tests involving materials with continuous properties, such as columns of water or areas of space (cf. Gelman & Baillargeon, 1983).
3. Computational models should generate the same behavior as do the children in the specific domain being modeled. More specifically, they should compute an approximation of subjects' final knowledge states, given an approximation of initial states and external inputs like those imposed by experimental and/or natural conditions.

Although more than 20 years have passed since Klahr and Wallace (1970) proposed an information-processing approach to cognitive development, as yet there are no computational models of any major developmental transitions that satisfy all of these criteria. The Klahr and Wallace work on the development of quantitative concepts (Klahr, 1973, 1984; Klahr & Wallace, 1973, 1976) consists of verbal descriptions, flow charts, and production-system models of distinct performance levels in the general domain of quantitative reasoning, including subitizing, counting, estimation, class inclusion, transitive reasoning, and quantity conservation. However, with respect to transition processes, their most

1. Subitizing is a fast and accurate process of determining the numerosity of small sets of entities.

fully developed model (Wallace, Klahr, & Bluff, 1987) goes only as far as a partially specified architecture for supporting developmental transitions.

More recent computational accounts of developmental phenomena have been of two kinds:

1. One type of account is highly constrained by data from empirical studies of children's acquisition of knowledge in a domain, but the computational model itself is not constrained by any theoretical principles. Instead, it is based on pragmatic decisions about how to implement a set of assumed mechanisms (e.g., Siegler, 1991).
2. The other type of account is based on a very broad set of theoretical assumptions that are consistent with a wide range of specific implementations, such as the adaptive production system used by Halford et al. (1991) to model the acquisition of transitive inference or the connectionist model used by McClelland (1991) to model the acquisition of balance scale rules. Computational models of this type, while suggesting interesting learning mechanisms, tend to be relatively unconstrained by any particular empirical results on children's knowledge acquisition.

The purpose of our approach is to formulate a model that is tightly constrained by both a general theory of the cognitive architecture and a specific set of empirical results. Q-SOAR's challenge is to demonstrate that it can model the learning reported in Gelman's (1982) training study, which we describe next. The further issue of explaining the general developmental regularities listed above is addressed in the final section of the chapter.

3. A Training Study

In constructing a simulation of training studies, we were faced with the choice of modeling either our own arbitrary view of the essential properties of a typical training situation, or one specific training situation chosen from the vast conservation training literature. The problem with the former choice is that there is no "typical" training study. Detailed examination of the literature on conservation training studies reveals that they vary along so many potentially relevant dimensions that it is nearly impossible to get agreement even on a prototypical training study,

let alone a set of defining properties. For example, Field's (1987) review organizes a collection of 25 recent conservation training studies with preschoolers along nine dimensions and three theoretical orientations.² Without any principled basis on which to construct a typical study, we chose to simulate a specific training study with well-defined procedures and clear quantitative outcomes.

3.1 Gelman's Training Procedure

As noted, we chose a training study reported by Gelman (1982) in which three and four year olds were trained in a brief session using small collections of discrete objects ($N = 3 - 4$) in both equivalence (two rows of equal number) and inequivalence (two rows of unequal number) relations, and in which the transfer test included both small ($N = 4 - 5$) and large ($N = 8 - 10$) collections. Gelman trained one group and used two types of control groups. Children in the Experimental group were trained with two types of collections in counterbalanced order. Half the children were first shown an equivalence relation (two rows of four items each), and the other half were first shown an inequivalence relation (one row of four and one row of three). In both equivalence and inequivalence collections, the items were initially placed in one-to-one correspondence.

For each type of collection there were nine steps, as illustrated in Figure 1: (1) The display was presented in one-to-one correspondence and the child was instructed to count the number of items in one of the rows. (2) That row was covered by the experimenter and the child was asked, "How many are under my hands?" (3) The child was instructed to count the number of items in the other row. (4) That row was covered by the experimenter and the child was asked, "How many are under my hands?" (5) The child was asked to judge whether the two uncovered rows contained "the same number or a different number" of items. (6) While the child watched, the length of one of the rows was spread or compressed. (7) The experimenter pointed to the altered (or unaltered) row and asked, "Are there still N here?" (8) The experimenter pointed to the other row and asked the same question. (9) The child was asked

2. The procedural dimensions were design, pretest, training, materials, reinforcements, verbal rule instruction, post-test, justifications, and delayed post-test. The theoretical orientations were specific experience, cognitive readiness, and perceptual readiness. Space does not permit an elaboration of these "guiding models", as Field calls them, but it is clear that training studies vary widely along both procedural and theoretical dimensions.

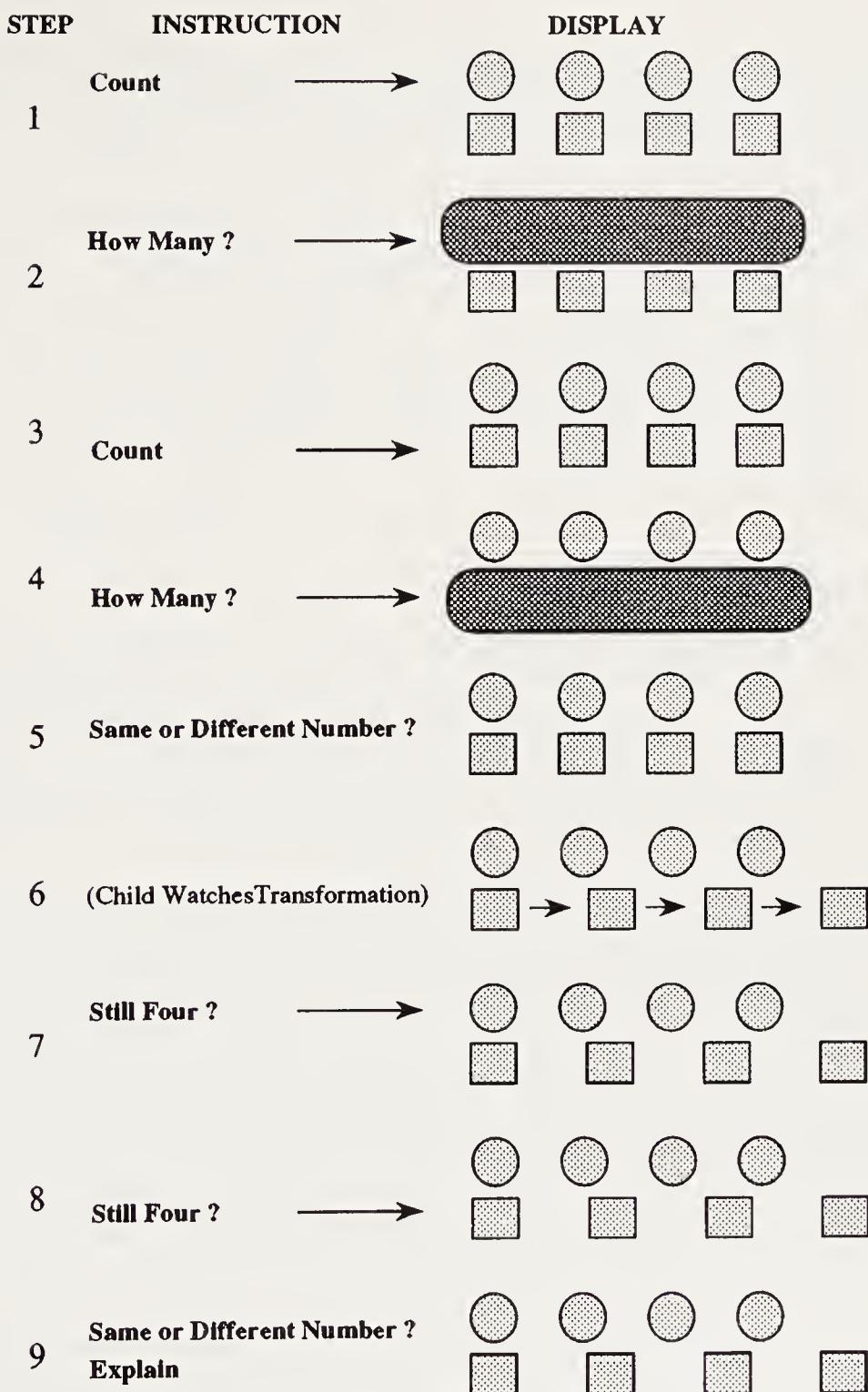


Figure 1. Graphical representation of the Experimental procedure.

whether the pair of rows had the same number or a different number of items, and to explain his/her judgment. All children answered the questions above correctly (except for one three year old who needed a slight extra prompt).

Gelman used two control groups. Children in the Cardinal-Once group were exposed to only one row (of three or four items). For that one row, they were exposed to steps 1-2 and 6-7 listed above. (Each row was altered four times to provide a comparable number of counting trials between the Experimental and control groups.) The other control group (No-Cardinal) simply counted single rows of three or four items, but the children in that group were not asked to "indicate the cardinal value rendered by the count".

3.2 Conservation Test

Immediately following the Experimental or other procedures, conservation tests were administered. Each child was given four different conservation tasks (large or small set size, and equal or unequal numbers of items in the two rows). Small sets included either four and five or five and five items, and large sets included either eight and ten or ten and ten items.

The order of presentation of large and small set sizes was counterbalanced as was the order of conservation of equality and inequality tasks within a set-size range. The equal arrays were equal in length prior to the transformation and unequal in length after being transformed. The reverse was true for the nonequivalent arrays: before being transformed they were unequal in length and then equal in length after the transformation. The conservation trials were run in the standard way older children are tested, and *included requests for explanations*. Likewise, children were discouraged from counting. (Gelman, 1982, p. 213)

Because children were discouraged from counting, and because one or both of the rows had at least five items, and because children of this age do not count beyond three or four items very reliably (Fuson, 1988), it is likely that the equivalence (or non-equivalence) of both large and small arrays was established by one-to-one correspondence.

Table 1. Proportion of correct conservation judgments (over all four judgments and all subjects) in each condition, derived from Table 1 in Gelman (1982).

AGE & SET SIZE	EXP'L (N=32)	CARD'L-ONCE (N=24)	NO-CARD'L (N=16)
3'S ON SMALL SET SIZE	71	11	13
4'S ON SMALL SET SIZE	75	58	16
3'S ON LARGE SET SIZE	72	8	0
4'S ON LARGE SET SIZE	65	34	13
OVERALL THREES	71	9	6
OVERALL FOURS	70	46	15

3.3 Results

For both the large and small sets, there was almost no difference in the equal and unequal set sizes, so those results will be collapsed in the following discussion. Table 1 shows the overall proportion of correct judgments on conservation tasks. The effect of condition is striking: overall, the Experimental groups passed about 70% of the conservation trials, compared to passing rates from 0% to 15% for the “untrained” (No-Cardinal) groups. Trained threes and fours did equally well on large and small sets.

The interesting difference between the threes and fours occurred in the Cardinal-Once groups. For threes, Cardinal-Once training had no effect, but for fours it had a substantial effect when tested on both small and large set sizes. This is important, because the children in the Cardinal-Once group were trained only in identity conservation (transforming a single row), rather than equivalence conservation (transforming one of a pair of rows).³ That is, they were never trained to notice that the relation between two different collections remained the same under a perceptual transformation, nor could they use one-to-one correspon-

3. See Klahr (1984) for a full discussion of the difference between identity conservation (IC) and equivalence conservation (EC). Note that Klahr's account of the acquisition of conservation rules is presented entirely in terms of the simple IC situation.

Table 2. Reference to transformation or one-to-one correspondence of Gelman's explanation categories.

	TRANSFORMATION	ONE-TO-ONE
IRRELEVANT TRANSFORMATION	×	
ADDITION/SUBTRACTION	×	
INITIAL EQUALITY/INEQUALITY	×	
ONE-TO-ONE CORRESPONDENCE		×

dence to reason about the effects of the transformations to which they were exposed. Instead, they could only learn that the initial and final cardinal value of a collection remained unchanged under certain kinds of transformations (i.e., spreading and compressing). Apparently, many four year olds, though few three year olds, were able to learn about transformations without the further help of one-to-one correspondence.

Gelman's categorization of children's explanations for their correct responses are presented in Table 2 in terms of (our interpretation of) whether the explanation makes reference to the transformation or to one-to-one correspondence. Gelman gives examples of two categories: *Irrelevant transformation* ("They just moved.") and *Addition/Subtraction* ("You need another one to make them the same.") explanations. *Initial equality/inequality of number* explanations presumably stated that the original value still held, while the content of *One-to-One Correspondence* explanations is obvious. The majority of the children's explanations referred to the transformation, and there were many more of these transformationally referenced explanations than there were explanations in terms of one-to-one correspondence. More specifically, for the Experimental threes, Experimental fours, and Cardinal-Once fours, the proportion of transformationally referenced explanations was 61%, 81%, and 65%, respectively, while for the same groups, one-to-one correspondence was used for only 21%, 9%, and 15% of the explanations.

Three year olds did not benefit from Cardinal-Once training, and four year olds in the Experimental group benefited more than did their age-mates in the Cardinal-Once group. Rather than attribute these differences to the role of one-to-one correspondence, we note that subjects

in the Experimental group, but not in the Cardinal-Once group, got repeated exposure to observations of the following transitive relation. When, for example, two rows of objects have the same number, and after spreading the transformed row has the same number as before, then the untouched row and the transformed row still have the same number of objects.

Note that it is not difficult for the child to compute the effect of the transformation. First, each set was counted before and after the transformation for the Experimental trials, rendering one-to-one matching redundant. Second, in Experimental trials, the pre- and post-transformation information is visible in the form of a transformed and an untransformed row after the transformation has taken place. However, in the Cardinal-Once trials, memory of the pre-transformation information is always required to compute the transformation's effect. All of the three year olds and some of the four year olds apparently needed this additional information (and reduction of processing) that was provided to the Experimental group.

4. A Theory of Number Conservation Knowledge

We can summarize our view of the important difference between the Experimental group and the Cardinal-Once group as follows. Subjects in the Experimental group were exposed to equivalence (or inequivalence) conservation trials in which they observed and encoded an initial quantitative relation between two collections, and then observed a quantity-preserving transformation on one of the collections. They then requantified both collections and noted that the relation had not changed. In contrast, subjects in the Cardinal-Once group, because they were dealing with only one collection, rather than two, were in an identity conservation situation. That is, they had to judge, after observing a spreading or compressing transformation, whether the quantity following the transformation was the same as the quantity preceding it; they could not simply requantify and compare the two rows.

In both situations, acquired knowledge stemmed primarily from the discovery that certain types of transformations have no effect on the numerosity of an object set, even though the transformations may affect other properties, like the spatial density or length of the set. This conclusion is independent of the number of objects in the set that was

measured when the new knowledge was created. In other words, what was learned was a characterization of the quantity-preserving aspects of the transformation in question.

Q-SOAR was built to model this piece of knowledge acquisition. In order to do this, the system must be able to specify:

1. The knowledge state prior to the training (i.e., a non-conserving child);
2. The encoding of the collection(s) prior to transformation. This will include salient features such as number, length, and density, as well as other features that may ultimately be irrelevant for the task at hand;
3. The encoding of the relation between collections (for the Experimental group);
4. The encoding of the collection(s) following transformation;
5. The encoding of the physical aspects of the transformation (e.g., salient motion, how objects were moved, how many were moved, direction of movement); and
6. New knowledge acquired from repeated trials of the kind presented to both the Experimental group and the Cardinal-Once group.

The model will have two variants, and each variant will be exposed to the three kinds of stimulus presentations (corresponding to the Experimental, Cardinal-Once, and No-Cardinal groups): Q-SOAR-4 will model the four year olds, who learn from both the Experimental manipulations and the Cardinal-Once manipulations. Q-SOAR-3 will model the three year olds, who learn only from the Experimental condition.

This set of general hypotheses about the essential mechanisms involved in the child's acquisition of number conservation can be called Q Theory, to distinguish it from Q-SOAR, which conjoins Q Theory with the assumptions of a particular cognitive architecture (SOAR) to form a more complete operational theory. A full theory of conservation will ultimately contain assumptions about the nature of the environments in which development takes place. Indeed, it is the lack of justifiable assumptions that can be made about naturally occurring conservation experiences that forces us to focus entirely on training studies.

5. The SOAR Architecture

This section describes the relevant aspects of the SOAR architecture. More detailed accounts of it exist elsewhere (Laird, Newell, & Rosenbloom, 1987; Laird, Swedlow, Altmann, & Congdon, 1989; Newell, 1990). Besides being an operational architecture, SOAR is also a theory of cognition that has been shown to explain a wide variety of psychological phenomena. No attempt can be made to describe that wider background here (cf. Lewis et al., 1990; Newell, 1990).

All tasks are formulated in SOAR as search in *problem spaces*, where operators are applied to states in an attempt to attain a goal state. Problem spaces can be thought of as packages of knowledge about different tasks. The operators within a given space (and knowledge about constraints on legal states) define the problem solver's competence for a task. For example, a complete problem space for the Missionaries and Cannibals puzzle contains the necessary operators to carry out moves, knowledge about the goal state, and knowledge about legal and illegal moves. Problem solving proceeds sequentially by decisions that select *problem spaces*, *states*, and *operators*. This processing gathers knowledge from a long-term recognition memory that is implemented as a production system. This memory matches structures in working memory and retrieves knowledge that elaborates the existing state and suggests preferences for the next step to take.

If SOAR cannot make a decision, an impasse occurs. In this case, SOAR automatically generates a subgoal in which a new problem space can be used to find the required knowledge. A major reason that SOAR exhibits the impasse and subgoal pattern is that not all of the knowledge required to carry out a task can be searched for within a single problem space. For example, should the goal arise in the Missionaries and Cannibals context to explain why the boat does not sink, there will be no knowledge in the problem space to implement that process. In response, an impasse will arise and in the resulting subgoal, SOAR will select a problem space for solving such an explanatory problem, since this is a different task requiring different knowledge. Once that knowledge is found, subgoals are resolved and processing continues where it left off (Newell, 1990).

SOAR has a single learning mechanism, called *chunking*, that learns new productions, or *chunks*, for resolved impasses. When similar situations are encountered, the knowledge generated by the previous subgoal-processing is retrieved automatically so that the impasse is not recre-

ated. The chunk will apply in a wider set of circumstances than the exact conditions under which it was created. This is because the chunking mechanism carries out an analysis that is a form of explanation-based learning (Mitchell, Keller, & Kedar-Cabelli, 1986; DeJong & Mooney, 1986; Rosenbloom & Laird, 1986; Mooney, this volume) to determine the critical features of the situation that led to the creation of the new knowledge. In future situations these act as cues to make the new knowledge available. The behavioral implication of chunking is that SOAR exhibits a shift from deliberate to automatic processing as the situations that it encounters become increasingly familiar. In other words, knowledge becomes compiled from search-based retrieval to recognition-based retrieval (Anderson, 1987; Rosenbloom & Newell, 1986).

6. The Acquisition of Number Conservation Knowledge

The knowledge and processes that enable Q-SOAR to acquire number conservation knowledge are implemented as a set of problem spaces that are depicted in Figure 2. The figure shows the problem spaces that are selected to carry out processing in response to given deficiencies in available knowledge; these deficiencies are stated as labels on the downward-pointing sides of the arrows. Once sufficient knowledge is returned (as depicted by the upward-pointing side of the arrows), the original processing can continue. The new knowledge becomes immediately accessible on later occasions in the form of chunks. The top panel depicts the knowledge required to interpret task instructions and to establish initial values before a transformation is applied. The lower panel depicts the knowledge involved in determining the quantitative effects of transformations.

The figure also distinguishes between task-motivated problem spaces (unshaded) and theory-motivated problem spaces (shaded). The unshaded spaces contain those operations that any task analysis of the training studies would deem to be necessary for its successful completion. These processes include the ability to understand instructions, create responses, and determine relative or absolute values for the objects used in training and testing. The shaded problem spaces contain operations that we, as theorists, assert are necessary to enable the cognitive architecture, SOAR, to achieve the behavior and learning that constitute the attainment of number conservation as shown by children in the three to four year old age range.

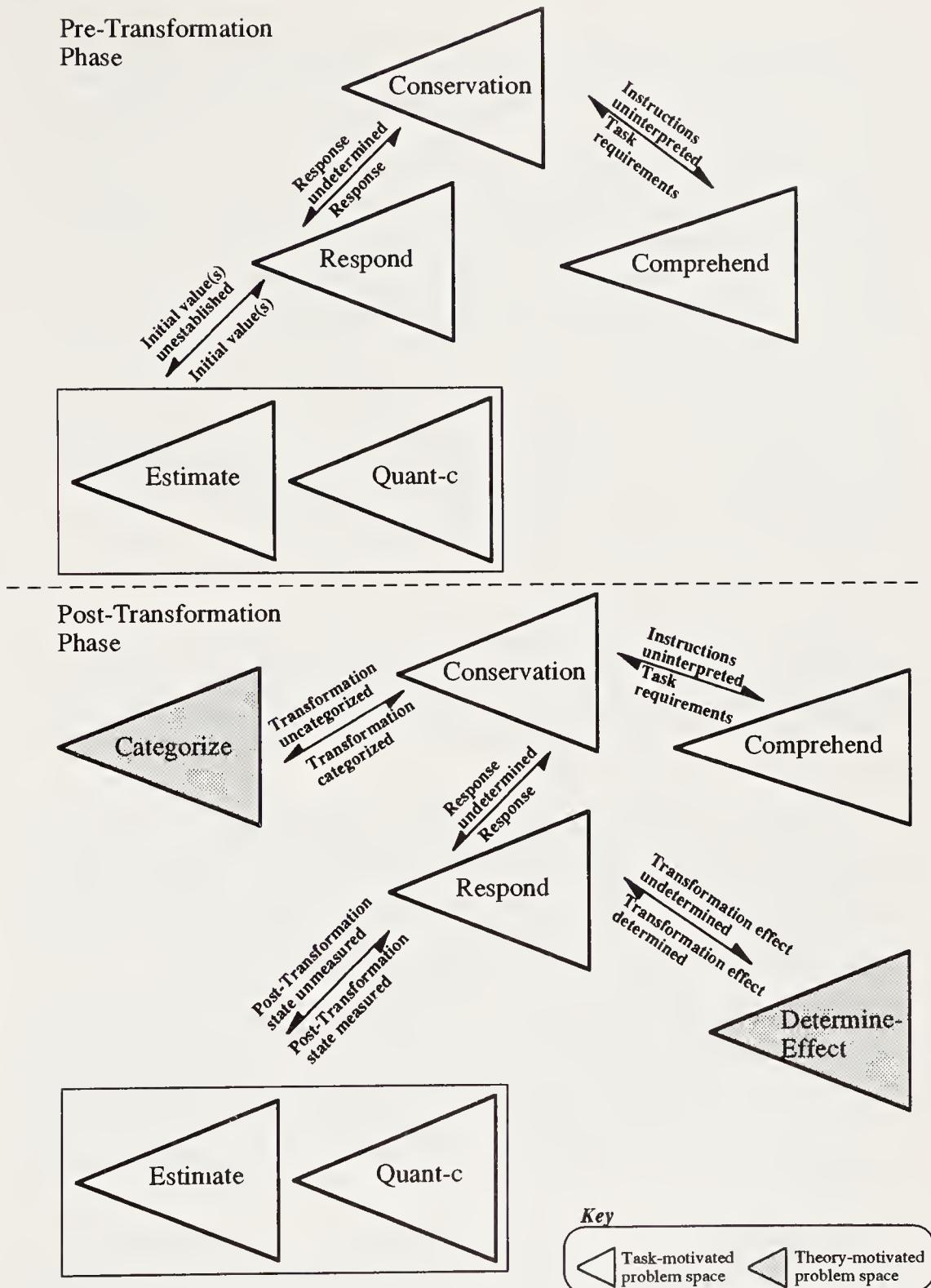


Figure 2. Problem spaces for the Q-SOAR model of conservation behavior.

Q-SOAR's design presumes that young children acquire number conservation knowledge by measurement and comparison of values to determine the effects of transformations on small collections of discrete objects. Having been shown a transformation to a set of objects, the child first categorizes the transformation and then initiates a conservation judgment about the transformation's effect. Ideally, categorization will identify the observed transformation as an instance of a larger class, with effects that are known to be associated (through chunking) with this class. If not, then pre- and post-transformation values created by measurement processes are compared to determine the effect of the transformation. The learning over this processing creates new knowledge about this kind of transformation, which will become available on future occurrences in similar contexts.⁴ Now the transformation's effects can be stated without the need for any empirical processing. In other words, the necessity of the effects is recognized.

6.1 Theoretical Assumptions

The behavior exhibited by Q-SOAR is determined, in part, by five assumptions that we make about the knowledge and strategies used by three and four year old children in the context of conservation experiments. The five assumptions are:

1. *The numerical value of a row of objects will not change if the row is not transformed.* Gelman (1977) has shown in her "magic studies" that children as young as two years of age operate with the assumption that the numerical value of a set of objects will remain constant in the absence of any observable manipulation of the set.
2. *The numerical value of a row of objects is very likely to change if the row is visibly transformed by an experimenter.* Many experiments, notably those of Donaldson (1978), have shown that the act of an experimenter making explicit physical changes to a set of objects suggests to young children that some significant change will result from the action.
3. *Four year olds have the knowledge that they can verify whether assumption 2 is true or false where measurement is possible.* By mea-
4. The notion of similarity involved is the occurrence in the new situation of the same essential features used in the prior situation. There is no similarity metric involved.

suring the numerical value of the objects before and after the transformation, a determination can be made as to whether the action changed the number of the objects.

4. *Three year olds do not have the knowledge that a transformation's effects can be verified but they may be motivated to measure a transformation's effects if faced with two conflicting sources of evidence.* If one source of information suggests that the value of the objects has changed while another suggests that it has not, three year olds will attempt to determine the true effect of the transformation where measurement is possible.
5. *Three and four year olds have the capacity to store and recall pre- and post-transformation values but only four year olds do so systematically.* Four year olds have the knowledge that it is important to do this in order to determine the effect of a transformation. There is a sizable literature on young children's strategic use of their mnemonic capacity which indicates that, even if three year olds have the same capacity for remembering as older children and adults, they have little knowledge about how to exploit that capacity and thus their untutored memory performance is poor (Brown, Bransford, Ferrara, & Campione, 1983).

Q-SOAR's behavior in simulating the training study is described below in sections that correspond to the problem spaces involved. References will be made to the example procedure in Figure 1 so that the reader can keep track of both the current subproblem that Q-SOAR is attempting to solve and the arrangement of the objects concerned. The description will be presented using the Experimental procedure since it is a superset of the other two control conditions. The actual behavior that Q-SOAR produces during this procedure, and its subsequent behavior on one of the test conditions, is presented in the appendix to this chapter. (Recall that the Cardinal-Once group only experienced steps 1-2 and 6-7 and the No-Cardinal group experienced only step 1.)

6.2 Conservation Task Operations

To carry out a conservation task, five basic processes are required. These correspond to SOAR operators in the CONSERVATION problem space: COMPREHEND-INSTRUCTION, CATEGORIZE-TRANSFORMATION, DETERMINE-RESPONSE, RETURN-RESPONSE, and WAIT.

All operators process internal representations in working memory. These representations correspond to aspects of the external situation to which the model is attending. There is a focus of attention, which is determined partly by external sources, such as the experimenter asking questions or drawing attention to the experimental materials. It can also be determined by internal processing, such as attention to individual items during counting. The internal representations are in the form of *annotated models* (Lewis, Newell, & Polk, 1989), which are descriptions of attended aspects of the external situation expressed symbolically as objects with parts, properties, and relations to other objects. These models are manipulated by operators and augmented with knowledge from memory and from the external situation via perception.

Each step in the Experimental (and other) procedures is carried out in response to an instruction or request from the experimenter; we shall refer to these as *instructions*. When an instruction is perceived, its meaning must be represented as an annotated model. This model is constructed by a COMPREHEND-INSTRUCTION operator in the manner of an existing system, called NL-SOAR, which is implemented in the COMPREHEND problem space; Lewis, Newell, and Polk (1989) give a description of this process. COMPREHEND-INSTRUCTION operators produce a representation of a request called a *behavior-model object*. In step 1 of Figure 1, for example, the child is requested to count the row of circles. In this case, the COMPREHEND-INSTRUCTION operator would produce a behavior-model object representing the operation "measure", the argument "circles", and the dimension "number". Thus a behavior-model object is a child's representation to himself/herself of what behavior should be carried out to achieve the task. In general, the child is perfectly capable of behaving without such a plan, but in the case of an experiment, he/she must represent (and remember) the instruction to be carried out. These representations also play a role in mediating the speed of acquisition of conservation knowledge.

Once the instruction has been comprehended and represented, Q-SOAR must still produce a response. To do so, it selects the DETERMINE-RESPONSE operator. Q-SOAR implements steps 1 and 2 (and then steps 3 and 4) in Figure 1 with a single DETERMINE-RESPONSE operator, which is augmented with the instructions represented on the behavior-model object. If the response is not immediately available, then there will be an impasse and other problem spaces will be selected to compute the response. When the system has created a response that satisfies the

instructions (such as the value “four” for steps 1 and 2), it selects the RETURN-RESPONSE operator to output the answer and then waits for the next instruction by selecting the WAIT operator.

Q-SOAR’s first response to observing a transformation (step 6) is to categorize it. This is done before responding to post-transformation questions by selecting the CATEGORIZE-TRANSFORMATION operator. The categorization process is described in Section 6.6.

6.3 Response Determination

If the DETERMINE-RESPONSE operator in the CONSERVATION space cannot immediately produce a response to the counting instructions of steps 1 and 2, there will be an impasse. The RESPOND space will always be selected when the DETERMINE-RESPONSE operator impasses, since it contains the three operators that are required to create responses in the conservation task: MEASURE, COMPARE, and RECALL. In the case where steps 1 and 2 have been comprehended as requiring a measurement, the MEASUREMENT operator is selected. Depending on the number of objects and their representation (see Section 6.5.1), either the QUANT-C or ESTIMATE space is selected to carry out this measurement. That measurement is returned as the result of the MEASURE operator and, in turn, it is also returned as the result of the top-level DETERMINE-RESPONSE operator.

Once Q-SOAR has created and returned a measurement for the circles and squares, it perceives and comprehends the question “same or different number?” in step 5. The resulting DETERMINE-RESPONSE operator will be augmented with the instruction “compare”, the arguments “circles” and “squares”, and the dimension “number”. Since this is the first time the instruction has been encountered, there will be no immediately available response and, in the resulting subgoal, the COMPARE operator will be selected in the RESPOND space. This operator tests whether the values for the measurements of the two rows match, since only a “same” or “different” response is required. In the current example, that processing will be carried out in the QUANT-C space if a value for that comparison is not immediately available.

Once the comparison has been created and returned, and the transformation has been observed and categorized (step 6 and lower panel of Figure 2), the experimenter asks, “Are there still N objects?” for

each row (where N is the number of objects in the row). These instructions are treated in a similar way to steps 2 and 4, by selecting a DETERMINE-RESPONSE operator. However, the operator is now augmented with the operation “recall”. If no answer to the question is immediately available, that operator will impasse and the RECALL operator will be selected in the RESPOND space. The implementation of this operator differs when it is applied to rows that have been transformed. The alternatives depend on the model variant (Q-SOAR-3 or Q-SOAR-4) and training condition (Experimental or Cardinal-Once), as discussed in Section 6.4. Responses to the “Are there still N ?” question with respect to an untransformed row (e.g., step 7 in Figure 1) are dealt with in the following way. Recall that assumption 1 (in Section 6.1) stated that the numerical value of a row of objects will not change if the row is not transformed. This means that the value for the row in question is assumed to be the same before and after the transformation of the other row. Q-SOAR-4 is able to recall the pre-transformation value and return it as the answer to this question. This is not the case for Q-SOAR-3 (see assumption 5 in Section 6.1). However, even without retrieving the correct value with the RECALL operator, Q-SOAR-3 can answer the question correctly by quantifying the objects in question. This can be done by subitizing since no row has more than four objects.

6.4 Effect Determination

The way that Q-SOAR responds to the “Are there still N ?” question for transformed rows is not only to produce a post-transformation value but also to determine the effect of the transformation. The DETERMINE-EFFECT problem space is selected if the RECALL operator impasses, because no effect of the transformation is immediately available. In order to learn about the effect of the transformation, the system must compare the pre- and post-transformation values for the row. Also, some role must be attributed to the transformation for the value of that comparative judgment. This can be as simple as identifying it as the action that created the post-transformation array. In other words, this process creates the knowledge to answer the implicit question, “What change did the transformation make to the number of objects?” This new knowledge states that, whenever such a transformation is applied, the relation between the pre- and post-transformation values that have

just been computed will hold for the dimension in question. For example, the response to step 9 in Figure 1 is that a spreading transformation causes no change because it produces an identical value to the one that existed before it was applied, namely “four squares” in both cases.

6.4.1 EFFECT DETERMINATION IN Q-SOAR-4

Q-SOAR-4 determines the effect of transformations in the same way for both the Experimental and Cardinal-Once conditions. Although assumption 2 in Section 6.1 stated that children believe that the value of a row will change when it is transformed, assumption 3 stated that four year olds have the knowledge that this can be verified when it is possible to measure the materials before and after a transformation is applied. This is the case in both of these conditions because of the small number of discrete objects. Thus, in all cases, Q-SOAR-4 makes a pre-and post-transformation value comparison to determine the effects of observed transformations.

6.4.2 EFFECT DETERMINATION IN Q-SOAR-3

Q-SOAR-3 behaves differently in the Experimental and Cardinal-Once conditions. In the Experimental condition its behavior is like that of Q-SOAR-4, due to assumption 4. This assumption stated that three year olds do not possess knowledge about verifying a transformation’s effect but may be induced to do so if they are faced with two conflicting sources of evidence. This is always the case with the Experimental condition. Before transformations, the two rows are in one-to-one correspondence so equal trials have equal-length rows and unequal trials have unequal-length rows. In other words, perceptual information and quantitative information are not in conflict. However, “transformations on unequal trials yielded rows of the same length; equal trials involved rows of different lengths” (Gelman, 1982, p. 212). In other words, after transformation, quantitative information, which was available via subitizing, and perceptual information were in conflict: unequal rows were the same length and equal rows were different lengths. This conflict leads Q-SOAR-3 to recall the pre-transformation value it measured and check it against the post-transformation value of the row.

In the Cardinal-Once condition, no such conflict exists. There is a single row of objects which, when transformed, takes on a new visual

appearance. There is nothing in the visual array to suggest that the assumed change in its numerical value should be doubted. Thus Q-SOAR-3 makes no attempt to compare pre- and post-transformation values of rows. As with the untransformed row, it answers the question "Are there still N ?" by requantifying. Since no comparison is made to the original value, no learning can take place as to whether the transformation has had any effect on the numerical value of the row.

6.4.3 EFFECT DETERMINATION OPERATIONS

There are two operators in the DETERMINE-EFFECT space. The RECALL operator recalls a pre-transformation value for comparison to the post-transformation value. The DETERMINE-EFFECT operator matches pre- and post-transformation values for the transformed row as described above. The process is a simple match that tests whether the values are the same or not. A requested determination of the magnitude or direction of the change will require accessing quantification knowledge. The result of this match is returned by the DETERMINE-EFFECT operator as the effect of the transformation, and the basis of its determination (e.g., that pre- and post-transformation values matched) constitutes an explanation. This will be the result of the RECALL operator in the RESPOND space and, ultimately, the DETERMINE-RESPONSE operator in the CONSERVATION space. The chunks that are built when this new knowledge is returned enable immediate retrieval of the effect of the current transformation. These chunks will fire in response to selection of the DETERMINE-RESPONSE operator, letting Q-SOAR immediately return the effect and explanation of the transformation. This demonstrates the shift in conservation performance from empirical examination of materials to direct explanation of the transformation's effects.

6.5 Quantification and Estimation

In the preceding sections we showed how the acquisition of number conservation knowledge in Q-SOAR is founded on empirical processing, whose results are then used by the effect determination process. In this section we examine the quantification and estimation abilities available to Q-SOAR that implement the measurement and comparison processes.

6.5.1 QUANTIFICATION

The primary measurement capability that is possessed by young children is quantification. This quantification subsystem, which we call *Quant-C*, for “quantification in conservation”, includes only capabilities that produce cardinal values for small sets of entities. Thus it includes subitizing and counting of small sets.

In cases where values are to be determined in terms of number, the QUANT-C problem space may be selected to implement the measurement. Selection of the space depends on two factors. The first is whether the conservation property represented on the MEASURE operator suggests the use of Quant-C processes (e.g., in the case of discrete objects but not liquid). The second is the suitability of the representation for the application of operators in the QUANT-C problem space.

Even in cases where the conservation property suggests Quant-C processes for determining equivalence, the problem solver may still be unable to use it. In order to select the QUANT-C space, the representation of objects to be measured must be in the form of symbols representing discrete objects that are in *one-onto-one* mapping (hereafter *onto*) with their external referents. We assume such representations are only possible for set sizes within the range that young children can subitize: a limit of four objects. Above this limit, a much looser *one-into-one* mapping (hereafter *into*) is used.

The process of subitizing in the QUANT-C space is not controlled by an operator: it is simply that of creating an *onto* representation of up to four external referents. This approach is based on the view that there is a special code for the representation of discrete quantities which is primitive to the architecture and which differs from the formal code used to communicate about numbers with words such as “three” or symbols like “3”. We call this primitive representation the *basic quantitative code* (Simon & Newell, 1990) and assume that it provides the agent with an ability to represent quantity in a primitive form.

There are six operators in the QUANT-C space: ATTEND, INITIALIZE, FETCH-NEXT, COUNT-NEXT, COMPARE, and MEMORIZE. The ATTEND operator attends to the objects specified in the behavior-model object and sets up an *onto* representation. All of the other operators are involved only if counting and not subitizing is to be carried out. The INITIALIZE operator selects a mark for identifying objects to be counted, selects an initial word from the counting string to be used, and selects

an initial object to be processed. The COUNT-NEXT operator assigns a selected count word to a marked object and, where cardinal responses (Fuson, 1988) are to be returned, assigns that label to the cardinality of the set. FETCH-NEXT obtains a next item to be counted, marks it, and obtains a next count word to be assigned. The COMPARE operator can be used to test either the relative similarity or difference of values created by MEASURE operators. The MEMORIZE operator carries out a deliberate act of memorization on the final response to a pre-transformation and post-transformation instruction, so that the results are stored in long-term memory and are available for the processes that determine the effect of the transformation.

None of these operations can be directly applied to *into* representations. However, one can count large collections of objects if perceptual and motor operations can be carried out to serially map individual items onto their external referents, thereby creating transitory *onto* representations for up to four items at a time. If this cannot be done or if a decision is made against doing so, the only recourse is to use estimation operations.

6.5.2 PERCEPTUAL ESTIMATION

Perceptual estimation in the children modeled by Q-SOAR is unidimensional — the relative number of two numerous rows of objects is determined either by length or by density, but not both. Siegler (1981) showed that a child's ability to integrate more than one dimension to solve a range of problems does not develop until around eight years of age. Thus, estimation in conservation settings is inaccurate, since one dimension is often inadequate for an accurate quantitative judgment.

The ESTIMATION problem space is selected to obtain values for materials under a number of conditions. As discussed in Section 6.5.1, Q-SOAR may be requested to create a relative quantity judgment where there are too many objects to create an *onto* representation. In this case the model uses perceptual estimation, in which the primary cue as to quantity is the length of the rows. A MATCH operator carries out a type of one-to-one matching called *end matching* (Klahr & Wallace, 1976), which tests whether the end items of each row are above or below the end items of the other row. If this is not the case, the longer row is assumed to be more numerous. A MEMORIZE operator stores the result of this processing, just as in the QUANT-C space.

6.6 Categorization

As mentioned earlier, Q-SOAR categorizes observed transformations.⁵ This means that it identifies critical features that are common to individual transformations, such as that all spreading actions move things further apart irrespective of the objects in question. Chunks created from processing in the DETERMINE-EFFECT problem space associate the new effect with the category of the transformation, not to the specific instance. As a result, invariance effects will be cued by any new transformation that can be identified as a member of that category. This enables novel situations to cue knowledge acquired about other members of the same category. Thus we assume that all novices, especially young children, form concepts to facilitate plausible generalizations about novel instances. Chunking models this desirable behavior, as do some other methods of explanation-based learning. As noted by Mooney (this volume), categorization need not be limited to a single dimension, but it should be sensitive to current goals. For example, when confronted with studies of number conservation like Gelman's, Q-SOAR may "CATEGORIZE" spreading, compressing, piling, and distributing together since they have no numeric effect. In contrast, if the concern is with spatial density, then compressing and piling would constitute a category with the opposite effect of spreading and distributing. The imposition of conceptual cohesiveness by goals or effects is related to *ad hoc* categorization introduced by Barsalou (1983) and discussed by Fisher and Pazzani (Chapter 6, this volume).

Q-SOAR selects the CATEGORIZE problem space when there is a transformation represented on the state and the CATEGORIZE-TRANSFORMATION operator in the CONSERVATION space cannot retrieve a type for it. The categorization process identifies in the representation of the transformation a set of features that are predictive of a certain classification. It is implemented as a recognition task. If a new instance is not immediately recognized as a member of a known class, features are progressively abstracted out of the instance description until the instance is recognized as a known class member. If no class is retrieved, then a new one is formed using the set of features in the new instance. For example, when all the features common to all spreading transfor-

5. The current version of Q-SOAR does not fully implement this process. Instead, the structures that would be created by an existing system called AL-SOAR (Miller & Laird, 1990) are fed into working memory.

mations are present, and none that are indicative of some other type of action (like compressing) are represented, the transformation will be treated the same way that other spreading transformations would be in the current context. This strategy is straightforward and sufficient to model Gelman's study, though a more general theory of conservation development will likely require more flexible principles of feature-discriminating power, like those found in Chapters 1 through 5 of this volume.

6.7 Learning Conservation Knowledge in Q-SOAR

The above subsections presented the problem spaces and operators that comprise Q-SOAR. How then do these components combine to create the number conservation knowledge that is the result of the effective training procedures? The answer is that they are called upon to contribute knowledge as Q-SOAR experiences impasses during problem solving. These impasses arise dynamically from the particular task that Q-SOAR is working on and the knowledge it brings to bear on each task at a given time. Thus, the conservation knowledge that the system has depends of what problems it has tried to solve and what knowledge it had available when it tried to solve them.

For example, if the knowledge required to respond to the question about the relative quantity of the two rows of objects in the initial array is not available, an impasse will arise. Q-SOAR will have already quantified the two values, but no comparative value will exist. The resulting series of impasses ground out in the selection of the COMPARE operator in the QUANT-C space. The successful creation of that comparative value resolves the impasse and creates a new piece of information that is available for later instances of the same problem. This kind of processing is repeated for every impasse that the system encounters. Some of these chunks simply reduce the amount of search that Q-SOAR engages in on subsequent trials (such as chunks that implement the instruction comprehension operators). Other chunks, such as those that arise from compare operators in the QUANT-C space, not only reduce search but also directly contribute to the ultimate conservation judgments that the system makes.

The result of one such impasse is the chunk (hereafter the *conservation chunk*) marked in the appendix that produces the conservation response. Due to the explanation-based nature of chunking, some gen-

eralization will occur with respect to the applicability of the chunk. Specifically, only features that existed before the impasse arose can become conditions for chunks. This is to ensure that an impasse for the same problem will not recur. However, not all of the pre-existing features will become conditions; only those that are used to compute the result in the subgoal will be selected. This means that the chunked result will be retrieved in a wider set of circumstances than the one in which it was formed. However, it does not mean that Q-SOAR exhibits conserving responses after one trial. When simulating human cognition, SOAR builds chunks only for the results created from the lowest goal in a subgoal stack. This *bottom-up* chunking causes the architecture to exhibit a gradual progression from deliberate (search-based) to automatic (recognition-based) behavior.

In the case of Q-SOAR, the conservation chunk at first only implements the response to the DETERMINE-EFFECT operator because that was the operator that led to the final impasse from which the chunk was created. Only after a series of trials is there a single subgoal caused by the top-level DETERMINE-RESPONSE operator. Then the information in the original chunk becomes available to implement that operator and thus enable a recognitional response to the effect of an observed transformation, as in the case of a conserving child.

By acquiring conservation knowledge in this way, Q-SOAR does not create any single knowledge structure that represents a "conservation concept". Instead, it builds a series of chunks that, when appropriately cued, enable the system to exhibit number conservation. In other words, rather than learning concepts that define the features of conserving transformations, Q-SOAR acquires generalized knowledge about the effects of observed transformations that is cued by other, similar transformations in similar contexts. As already described, these pieces of knowledge are acquired incrementally as problems are solved by the system with different amounts of available knowledge. In this particular modeling study, Q-SOAR was led to acquire its conservation knowledge by the use of a training regime. However, this was not a supervised concept-learning situation in which pre-classified examples of concepts are presented for the system to learn. Indeed, Q-SOAR is never presented with the concept of conservation; it is merely asked to solve a series of problems that were experimentally demonstrated to result in the acquisition of conservation knowledge. These kinds of problems can be encountered and solved without supervision and, as can be seen in

the next section, we claim that Q-SOAR should be capable of learning conservation knowledge without training but at a slower speed than demonstrated here.

7. Toward a Full Theory Of Conservation

Q-SOAR successfully models the acquisition of conservation knowledge attained by subjects in Gelman's training study in an implementation within SOAR's unified cognitive theory. In this final section, we describe the behavior of Q-SOAR before and after training. We also describe what we anticipate as the necessary steps toward a full theory of conservation in other domains.

Ultimately, one can evaluate an enterprise such as the one presented here in terms of Piaget's (1964) well-known criteria for "real" conservation:

But when I am faced with these facts [that *learning* of structures seems to obey the same laws as the *natural development* of these structures], I always have three questions which I want to have answered before I am convinced.

The first question is, "Is this learning lasting? What remains two weeks or a month later?" If a structure develops spontaneously, once it has reached a state of equilibrium, it is lasting, it will continue throughout the child's entire life. When you achieve the learning by external reinforcement, is the result lasting or not and what are the conditions for it to be lasting?

The second question is, "How much generalization is possible?" ... When you have brought about some learning, you can always ask whether this is an isolated piece in the midst of the child's mental life, or if it is really a dynamic structure which can lead to generalizations.

Then there is the third question, "In the case of each learning experience what was the operational level of the subject before the experience and what more complex structures has this learning succeeded in achieving?"

To these three questions, we add a fourth: How can subjects (and Q-SOAR) learn so rapidly from a brief training study, when untrained subjects take several years to acquire the same knowledge?

7.1 Durability and Robustness of Learning

With respect to Piaget's first question, Q-SOAR makes a specific theoretical claim: a chunk, once learned, is always available, and will be evoked whenever the context-specific information that was included in the original chunk is recognized and encoded. For the SOAR architecture, chunking is an automatic acquisition mechanism that is applied to all processing that takes place. Thus, by undertaking the processing that is induced by the externally driven training procedure, the learning of conservation knowledge will occur.

The empirical prediction associated with this claim is not straightforward. The general pattern of results with increasingly remote post-tests is that, for a while, performance declines as a function of intervening time between training and testing, but then performance improves as one would expect with the "natural" acquisition of conservation. At present we have no principled explanation of this in terms of chunking.

7.2 Generalization

The second question refers to the specificity of learning from experience. This is a well-established empirical fact and is predicted by the chunking mechanism (Laird, Rosenbloom, & Newell, 1986). In the context of Q-SOAR, chunking predicts little generalization from learning about certain transformations of discrete objects to other transformations of different materials (e.g., the pouring of water). Indeed, this is what one usually finds from conservation training studies: very little generalization to other kinds of quantity conservation.

Transfer from small to large number tasks is achieved by the generalization inherent in SOAR's chunking mechanism. The actual objects that are measured in determining a conservation judgment are not tested when it is retrieved from memory. There are tests for the kind of transformation and the conservation property (in this case, number) and these delimit the scope of transfer. If that were not so, Q-SOAR would predict unrealistically fast learning: to transformations of quantities that are not affected in the same way as the one measured.

In addition, transfer is also limited with respect to continuous quantities such as volumes of liquid. Acquiring knowledge about continuous quantity is not addressed by Q-SOAR. Nevertheless, having acquired conservation knowledge based on small number measurement, a prob-

lem solver must come to appreciate what is common to transformations like lengthening and the pouring of liquids. This requires that these transformations be represented as actions that neither add nor remove any of the materials that they manipulate.

Finally, we suggest that the problem solving that enables the identification of the common features of different transformations and materials can best be described as a discovery process. The learner operates with a set of expectations based on current knowledge. This will at some point create a violation of the expected effects of a transformation. The learner's task then is to generate an hypothesis of what caused that violation, to devise ways of testing that hypothesis, and to integrate the results either into new hypotheses or modified knowledge. Research on scientific reasoning (Klahr & Dunbar, 1988), instructionless learning (Shrager, 1987), and analogy (Gentner, 1983) provides good explanations of the nature of such processes. Mediating factors in the effectiveness of that problem solving are the selection and combination of features that are considered for inclusion in the analysis (Bransford, Stein, Shelton, & Owings, 1981).

7.3 Operational Level and Structural Change

With respect to Piaget's third question, Q-SOAR makes explicit statements about the "complex structures" arising from the training of conservation responses. These can be seen by examinations of Q-SOAR-3 and Q-SOAR-4 before and after training.

7.3.1 Q-SOAR-3 AND Q-SOAR-4 BEFORE TRAINING

Before experiencing the three conditions of Gelman's training study, both versions of Q-SOAR are able to execute all the steps of the three experimental conditions. The only difference between the two versions is that Q-SOAR-3 does not start out with the knowledge that the effects of transformations can be verified by comparing pre- and post-transformation values. Apart from this difference, both versions have all the capabilities described in Section 6.

However, since neither variant has undergone any training and has not learned about the effects of any transformations, both versions of Q-SOAR fail all of the conservation tests that Gelman used. Neither system can accurately measure the large number of objects in the tests to yield

the correct comparative answers. They must use estimation, a process that results in the assertion that a longer row contains more objects than a shorter row. Finally, both untrained variants of Q-SOAR-3 and Q-SOAR-4 are unable to determine the effects of the transformations and so cannot state an explanation. Before training, then, both are true non-conservers. We will now examine their behavior after training. Since the No-Cardinal condition is not expected to induce any change in behavior, we will discuss only the results of the other two conditions.

7.3.2 Q-SOAR-3 AFTER CARDINAL-ONCE TRIALS

Without employing its memorization capability to recall and compare pre- and post-transformation values, Q-SOAR-3 cannot learn anything about the numerical effect of observed transformations. Thus, based on assumption 2 in Section 6.1, it always assumes that the value of the row changes. Since this is never the case in the experiment, Q-SOAR-3 is always wrong and it fails the conservation tests. As can be seen in Table 1, three year olds produced few correct responses.

7.3.3 Q-SOAR-3 AFTER EXPERIMENTAL TRIALS

As explained in Section 6.4, the conflicting information in Experimental trials after a transformation induces Q-SOAR-3 to recall and compare values in the same way as Q-SOAR-4. Thus, Q-SOAR-3 can construct a correct comparison and explanation from such trials. These can then be recalled later, enabling it to pass the conservation tests. The behavior of Q-SOAR-3 after the Experimental condition produces correct responses and explanations, and thus is consistent with the pattern of results in Table 1. It seems likely that the experience of this conflict and the resulting recall and comparison of values provide the means by which three year olds acquire the effect-verification knowledge that we have assumed to be available to four year olds and which we provided for Q-SOAR-4.

7.3.4 Q-SOAR-4 AFTER BOTH TRIALS

Having produced a quantitative response before a transformation (e.g., step 4 in Figure 1), Q-SOAR-4 selects the MEMORIZE operator to store the computed values in long-term memory. Then, in the DETERMINE-EFFECT problem space, it selects the RECALL operator to enable it to

compare pre- and post-transformation values to determine the effect of the transformation and create an explanation. This knowledge can then be recalled in the tests, enabling Q-SOAR-4 to pass the conservation tests after experiencing both the Experimental and Cardinal-Once procedures. This pattern of results is also consistent with that in Table 1. The higher proportion of correct responses in the Experimental group may reflect the fact that not all four year old children had acquired the effect-verification knowledge that we assumed for Q-SOAR-4. Those that had not would be expected to perform less well in the Cardinal-Once condition, just as was the case for three year olds.

7.4 Learning Speed

In Section 4 we stated that Q Theory is designed to account for the natural development of conservation, while Q-SOAR simulates only conservation learning in a single training study. Therefore, we should explain how the same processes can learn very quickly under experimental situations and yet take a few years to reach the same point during natural development. Two obvious factors are the differences in exposure and the availability of feedback. Intensive exposure to important features and informative feedback are characteristic of training studies, but neither of these is the case in unsupervised everyday activity.

However, we suggest that the greatest influence on learning speed is what we shall call the goal versus encoding interaction. A learner may activate the goal of measuring the effects of transformations. Alternatively, that learner's processing may be in the service of some other goal, such as building towers out of blocks. Even if the measurement goal has been activated, the learner may not attend to a property of the transformed materials that will reveal any number-invariance knowledge, such as the spatial density of a pile of blocks. Only if the child simultaneously has the goal of measurement and the encoding of number as the feature to be measured will he/she acquire number conservation. Well-designed training studies, such as Gelman's, foster just such optimal conditions, and in Q-SOAR these aspects are explicit in the representation of comprehended instructions. Similar directiveness appears to be provided for the child in relatively natural mother-child interactions, as set up by Saxe, Gearhart, and Guberman (1984). We

know of no evidence to suggest that the goal and property combination optimal for number conservation learning would be chosen by the child any more or less often than any other, although it is evident that children often set themselves the goal of counting things. Thus, three of the four types of opportunities for learning number conservation knowledge would not produce conservation learning in Q-SOAR.

8. Conclusion

In this chapter we presented Q-SOAR, a computational model of the acquisition of conservation knowledge as reported in a single experimental training study. This is the first such account to present a set of mechanisms, constrained by a unified theory of cognition, that can be shown to acquire conservation knowledge. Furthermore, we demonstrated not only that Q-SOAR can account for the rapid learning observed in the Gelman (1982) training study but also that, without modification, it may also be able to explain the slower, more opportunistic acquisition of invariance knowledge that is characteristic of a young child's everyday unsupervised learning experiences. Much remains to be done before we can claim that Q-SOAR gives a complete account of the acquisition of conservation knowledge. There exist many other training studies (Field, 1987) whose results should also be explicable by the mechanisms of Q Theory. The transfer to conservation of continuous quantity remains to be explained, and an account of "natural" conservation development is still an important goal. Nevertheless, we believe that the work reported in this chapter represents real progress in the creation of computational theories of conceptual development.

Acknowledgements

We thank Bob Siegler for comments and access to his experimental data and Rochel Gelman for further explication of her experimental procedures. In addition, the first author wishes to thank members of the SOAR group for invaluable help, Richard Young for comments, discussions, and support, and Doug Fisher for helpful suggestions on earlier drafts. This work was supported by Contract N00014-86-K-0678 from the Computer Science Division of the Office of Naval Research.

References

- Anderson, J. R. (1987). Skill acquisition: Compiling weak method problem solutions. *Psychological Review*, 94, 194-210.
- Barsalou, L. W. (1983). Ad hoc categories. *Memory and Cognition*, 11, 211-227.
- Bransford, J. D., Stein, B. S., Shelton, T. S., & Owings, R. A. (1981). Cognition and adaptation: The importance of learning to learn. In J. Harvey (Ed.), *Cognition, social behavior, and the environment*. Hillsdale, NJ: Lawrence Erlbaum.
- Brown, A. L., Bransford, J. D., Ferrara, R. A., & Campione, J. C. (1983). Learning, remembering and understanding. In J. H. Flavell & E. M. Markman (Eds.), *Handbook of child psychology: Cognitive development* (Vol. 3). New York: John Wiley.
- Campbell, R. L., Cooper, R. G., & Blevins-Knabe, B. (1988). *The development of subitizing in preschool and early elementary school children* (Tech. Rep. No. 63254). Yorktown Heights, NY: IBM, T. J. Watson Research Center.
- Chi, M. T. H., & Klahr, D. (1975). Span and rate of apprehension in children and adults. *Journal of Experimental Child Psychology*, 19, 434-439.
- DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1, 145-176.
- Donaldson, M. (1978). *Children's minds*. Glasgow, Scotland: Fontana.
- Field, D. (1987). A review of preschool conservation training: An analysis of an analysis. *Developmental Review*, 7, 210-251.
- Fuson, K. C. (1988). *Children's counting and concepts of number*. New York: Springer Verlag.
- Gelman, R. (1977). How young children reason about small numbers. In N. J. Castellan, D. B. Pisoni, & G. R. Potts (Eds.), *Cognitive theory* (Vol. 2). Hillsdale, NJ: Lawrence Erlbaum.
- Gelman, R. (1982). Accessing one-to-one correspondence: Still another paper about conservation. *British Journal of Psychology*, 73, 209-220.
- Gelman, R., & Baillargeon, R. (1983). A review of some Piagetian concepts. In J. H. Flavell & E. M. Markman (Eds.), *Handbook of child psychology: Cognitive development* (Vol. 3). New York: John Wiley.

- Gentner, D. (1983). Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7, 155-170.
- Halford, G. S. (1982). *The development of thought*. Hillsdale, NJ: Lawrence Erlbaum.
- Halford, G. S., & Boyle, F. M. (1985). Do young children understand conservation of number? *Child Development*, 56, 165-176.
- Halford, G. S., Smith, S., Mayberry, M., Stewart, J., & Dickson, J. C. (1991). *A computer simulation model of acquisition of transitive inference*. Paper presented at the Biennial Meeting of the Society for Research in Child Development, Seattle, WA.
- Klahr, D. (1973). Quantification processes. In W. Chase (Ed.), *Visual information processing*. New York: Academic Press.
- Klahr, D. (1982). Nonmonotone assessment of monotone development: An information processing analysis. In S. Strauss & R. Stavy (Eds.), *U-shaped behavioral growth*. New York: Academic Press.
- Klahr, D. (1984). Transition processes in quantitative development. In R. J. Sternberg (Ed.), *Mechanisms of cognitive development*. New York: W. H. Freeman.
- Klahr, D., & Dunbar, K. (1988). Dual space search during scientific reasoning. *Cognitive Science*, 12, 1-48.
- Klahr, D., & Wallace, J. G. (1970). An information processing analysis of some Piagetian experimental tasks. *Cognitive Psychology*, 1, 358-387.
- Klahr, D., & Wallace, J. G. (1973). The role of quantification operators in the development of the conservation of quantity. *Cognitive Psychology*, 4, 301-327.
- Klahr, D., & Wallace, J. G. (1976). *Cognitive development: An information processing view*. Hillsdale, NJ: Lawrence Erlbaum.
- Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33, 1-64.
- Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine learning*, 1, 11-46.
- Laird, J. E., Swedlow, K. R., Altmann, E. M., & Congdon, C. B. (1989). *SOAR 5 User's Manual* (Tech. Rep.). Ann Arbor: University of Michigan, Department of Electrical Engineering and Computer Science.

- Lewis, R. L., Huffman, S. B., John, B. E., Laird, J. E., Lehman, J. F., Newell, A., Rosenbloom, P. S., Simon, T., & Tessler, S. G. (1990). SOAR as a unified theory of cognition. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 1035–1042). Cambridge, MA: Lawrence Erlbaum.
- Lewis, R. L., Newell, A., & Polk, T. A. (1989). Toward a SOAR theory of taking instructions for immediate reasoning tasks. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 514–521). Ann Arbor, MI: Lawrence Erlbaum.
- McClelland, J. L. (1991). *Connectionist models of developmental change*. Paper presented at Biennial Meeting of the Society for Research in Child Development, Seattle, WA.
- Miller, C. S., & Laird, J. E. (1990). *A simple, symbolic model for associative learning and retrieval*. Unpublished manuscript, Artificial Intelligence Laboratory, University of Michigan, Ann Arbor.
- Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1, 47–80.
- Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.
- Piaget, J. (1952). *The child's conception of number*. New York: W. W. Norton.
- Piaget, J. (1964). Development and learning. In R. E. Ripple & V. N. Rockastle (Eds.), *Piaget rediscovered*. Ithaca, NY: Cornell University Press.
- Piaget, J. (1968). Quantification, conservatism and nativism. *Science*, 162, 976–979.
- Piaget, J. (1970). *Structuralism*. New York: Basic Books.
- Rosenbloom, P. S., & Laird, J. E. (1986). Mapping explanation-based generalization into SOAR. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 561–567). Philadelphia, PA: Morgan Kaufmann.
- Saxe, G. B., Gearhart, M., & Guberman, S. R. (1984). The social organization of early number development. In B. Rogoff & J. V. Wertsch (Eds.), *Children's learning in the zone of proximal development*. San Francisco, CA: Jossey-Bass.
- Shrager, J. (1987). Theory change via view application. *Machine Learning*, 2, 1–30.

- Siegler, R. S. (1981). Developmental sequences within and between concepts. *Monographs of the Society for Research in Child Development*, 46, 1–74.
- Siegler, R. S. (1991). *Variation and selection as cognitive transition mechanisms*. Paper presented at the Biennial Meeting of the Society for Research in Child Development, Seattle, WA.
- Siegler, R. S. (1989). Mechanisms of cognitive development. *Annual Review of Psychology*, 40, 353–379.
- Simon, T., & Newell, A. (1990). *Subitizing is faster than you think — If you don't ask 'How Many?'* Unpublished manuscript, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.
- Svenson, O., & Sjoberg, K. (1983). Speeds of subitizing and counting process in different age groups. *Journal of Genetic Psychology*, 142, 203–211.
- Wallace, J. G., Klahr, D., & Bluff, K. (1987). A self-modifying production system model of cognitive development. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.

Appendix: Sample Q-SOAR Run

Here we illustrate how the problem spaces described in Section 6 generate behavior when Q-SOAR is presented with a task. The following trace is an abstracted version of the steps presented in Figure 1, which show Q-SOAR in operation for the first time. The second trace shows the model's successful performance on a conservation test.

The traces retain only the critical information, showing the problem spaces (denoted by P) and operators (denoted by O) that are selected in response to the impasses that arise. An impasse is shown by processing in a subgoal (G) being indented under the operator that produced the impasse. When an impasse is resolved, processing continues at the highest level at which an operator can be selected. The operators are augmented with the instruction that led to their initiation or by the objects on which they are focused.

External arrays and instructions are depicted to the right of the trace in lower case and Q-SOAR's output is given in the center in upper case. The trace is marked with '**' at the points in the run where the key

conservation chunk is acquired and where it is evoked. Chunks are created continually throughout the run (one or more when returning from each impasse), but these are not shown.

ABSTRACTED RUN OF Q-SOAR DURING ITS FIRST OPERATION

()	()	()	()
]]]]]]]]

P: (CONSERVATION)

How many circles?

O: (COMPREHEND-INSTRUCTIONS)

O: (DETERMINE-RESPONSE)

P: (RESPOND)

O: (MEASURE)

P: (QUANT-C)

O: ((CIRCLE) ATTEND)

O: (INITIALIZE)

O: ((CIRCLE) COUNT-NEXT) Counting item: ONE

O: ((CIRCLE) FETCH-NEXT)

O: ((CIRCLE) COUNT-NEXT) Counting item: TWO

O: ((CIRCLE) FETCH-NEXT)

O: ((CIRCLE) COUNT-NEXT) Counting item: THREE

O: ((CIRCLE) FETCH-NEXT)

O: ((CIRCLE) COUNT-NEXT) Counting item: FOUR

O: (MEMORIZE)

O: (RETURN-RESPONSE)

Answer FOUR

How many squares?

O: (COMPREHEND-INSTRUCTIONS)

O: (DETERMINE-RESPONSE)

P: (RESPOND)

O: (MEASURE)

P: (QUANT-C)

O: ((SQUARE) ATTEND)

O: (INITIALIZE)

O: ((SQUARE) COUNT-NEXT) Counting item: ONE

O: ((SQUARE) FETCH-NEXT)

O: ((SQUARE) COUNT-NEXT) Counting item: TWO

O: ((SQUARE) FETCH-NEXT)

O: ((SQUARE) COUNT-NEXT) Counting item: THREE

O: ((SQUARE) FETCH-NEXT)

O: ((SQUARE) COUNT-NEXT) Counting item: FOUR
 O: (MEMORIZE)

O: (RETURN-RESPONSE) Answer FOUR
 Same or different number?

O: (COMPREHEND-INSTRUCTIONS)
 O: (DETERMINE-RESPONSE)
 ==>G: (OPERATOR NO-CHANGE)
 P: (RESPOND)
 O: (COMPARE)
 ==>G: (OPERATOR NO-CHANGE)
 P: (QUANT-C)
 O: (COMPARE)

O: (RETURN-RESPONSE) Answer SAME
 () () () ()
 □ □ □ □
 Still four circles?

O: (CATEGORIZE-TRANSFORMATION)
 O: (COMPREHEND-INSTRUCTIONS)
 O: (DETERMINE-RESPONSE)
 ==>G: (OPERATOR NO-CHANGE)
 P: (RESPOND)
 O: (RECALL)

O: (RETURN-RESPONSE) Answer FOUR
 Still four squares?

O: (COMPREHEND-INSTRUCTIONS)
 O: (DETERMINE-RESPONSE)
 ==>G: (OPERATOR NO-CHANGE)
 P: (RESPOND)
 O: (RECALL)
 ==>G: (OPERATOR NO-CHANGE)
 P: (DET-EFFECT)
 O: (RECALL)
 O: (DETERMINE-EFFECT)**

O: (RETURN-RESPONSE) Answer FOUR
 Same or different number?

O: (COMPREHEND-INSTRUCTIONS)
 O: (DETERMINE-RESPONSE)
 ==>G: (OPERATOR NO-CHANGE)
 P: (RESPOND)
 O: (COMPARE)

O: (RETURN-RESPONSE) Answer SAME
 VALUES MATCH BEFORE AND AFTER THIS TRANSFORMATION.

End -- Explicit Halt

The following rule paraphrases the chunk that Q-SOAR learns at the point marked '**' in the run above and that applies at the point marked '**' in the following trace. Pattern-match variables are preceded by question marks.

If Goal ?G1 has State ?S1 and Operator ?O1,
 and State ?S1 has Transformation ?T1 marked on it,
 and Transformation ?T1 is Spreading,
 and Operator ?O1 is Determine-Response,
 Then mark State ?S1 with Effect ?E1 of Transformation ?T1,
 where ?E1 states that ?T1 has the effect NONE on the
 property Number, because pre- and post-transformation
 numerical values matched.

Q-SOAR RUN ON A CONSERVATION TEST AFTER LEARNING

```

          () () () () ()
          [] [] [] [] []

P: (CONSERVATION)                                Same or different number?
O: (COMPREHEND-INSTRUCTIONS)
O: (DETERMINE-RESPONSE)
==>G: (OPERATOR NO-CHANGE)
      P: (RESPOND)
      O: (COMPARE)
      ==>G: (OPERATOR NO-CHANGE)
          P: (ESTIMATE)
          O: (MATCH-1-TO-1)           One-to-one end-match: SAME
          O: (MEMORIZE)
O: (RETURN-RESPONSE)                            Answer SAME
                                              () () () () ()
                                              [] [] [] [] []
                                              Same or different number?
                                              Explain
O: (CATEGORIZE-TRANSFORM)
O: (COMPREHEND-INSTRUCTIONS)
O: (DETERMINE-RESPONSE)**
O: (RETURN-RESPONSE)                            Answer SAME

VALUES MATCH BEFORE AND AFTER THIS TRANSFORMATION.

End -- Explicit Halt

```

Index

- Abelson, R. P., 193, 198, 205
ACLS, 105
ACT, 55, 225
ADECLU, 28, 30
Adelman, L., 244, 273
aggregation, 65, 66, 329
Agogino, A. M., 348, 350
Aha, D., 6, 35, 343, 350
Ahn, W., 17, 18, 21, 31, 34–36, 59,
 68, 166, 173, 174, 239, 273
AIR-CYL, 348
Albert, M. K., 6, 35, 343, 350
Allen, J. A., 124, 157, 158, 160,
 311, 318, 320, 382, 385–
 387
Allen, S. W., 223, 224, 231
Altmann, E. M., 435, 457
Altom, M. W., 57, 69
AM, 310
analogy, 155, 225, 232, 275, 452,
 457
Anderson, J. R., 6, 7, 11, 28, 29,
 31, 32, 36, 45, 48, 55, 56,
 58, 59, 67, 68, 80, 98, 100,
 120, 122, 124, 182, 184,
 202, 209, 210, 212, 214,
 224, 225, 227, 229, 231,
 232, 234, 235, 237, 238,
 245, 279, 298, 300, 334,
 357, 363, 436, 456
Anderson, J. S., 312, 318
Annenberg, M. R., 64, 68
ARACHNE, 32, 33
Arango, G., 313, 318
Arciszewski, T., 328, 350, 352
attention, 28, 30, 34, 35, 70, 111–
 113, 139, 156, 166, 170,
 201, 440
Austin, G. A., 4, 12, 36, 40–43, 75,
 100, 123, 160, 235, 240,
 273, 301, 302, 318
AUTOCALSS, 17, 29, 58, 59, 308,
 319
Baddeley, A. D., 118, 122
Baillargeon, R., 426, 456
Banerji, R. B., 154, 160, 281, 299,
 303, 311, 321
Bareiss E. R., 34, 36, 170, 176, 228,
 235, 321
Barsalou, L. W., 73, 100, 166, 174,
 214, 222, 230–232, 424, 447,
 456
Bauer, M., 16, 31, 38, 139, 159, 208,
 233, 245, 274
Beasley, C. M., 55, 68, 212, 231
Belkin, N., 321
Benjamin, D. P., 388, 420
Bennett, S. W., 181, 204
Bergadano, F., 299, 300
Berge, C., 47, 68
Berger, J. O., 50, 65, 68
Bezdek, J., 17, 22, 36
Biederman, I., 269, 273
Billman, D., 30, 33, 71, 208, 226,
 227, 231, 232
Binford, T. O., 181, 205
Biswas, G., 22, 32, 35, 36, 308, 318
Blevins-Knabe, B., 426, 456
Blockley, D., 327, 353
Bluff, K., 427, 459
Bobick, A. F., 80, 100
Bourne, L. E., 240, 274
Bower, G. H., 57, 68, 69, 123, 159,
 209, 231, 233, 385, 387
Boyes-Braem, P., 26, 41, 70, 101
Boyle, F. M., 426, 457
Bradshaw, G. L., 22, 167, 175, 195,
 309, 320
Brand, M., 177
Bransford, J. D., 439, 452, 456
Braverman, M. S., 169, 170, 174,
 295, 297, 301

- Breiman, L., 11, 36, 343, 350
 Brewer, W. F., 239, 273
BRIDGER, 313, 323, 324, 334, 337–
 349
 Brooks, L. R., 209, 210, 222–224,
 231–233
 Brown, A. L., 439, 456
 Brown, D. C., 348, 350
 Bruneau, M., 327, 351
 Bruner, J. S., 4, 5, 12, 36, 75, 100,
 123, 124, 240, 273
 Burke, R., 177
 Cagan, J., 348, 350
 Campbell, R. L., 426, 456
 Campione, J. C., 439, 456
 CAQ, 322
 Carbonell, J. G., 37, 40–42, 158,
 160, 175–177, 184, 202, 203,
 205, 212, 231, 232, 275–
 278, 303, 320–322, 351, 421
 Carey, S., 65, 68, 239, 273
 Carlson, B., 279, 294, 299, 301, 312,
 318
 case-based reasoning, 72, 170, 174,
 175, 214, 215, 224–226, 233,
 234, 279, 280, 302, 316,
 319, 320, 338
 category utility, 26–29, 31, 38, 75,
 86, 140, 151, 256, 289, 290,
 294, 297, 335, 337, 339,
 340, 345, 347, 368, 369,
 373–375, 384
 Chalnick, A., 227, 232
 Chan, P. K., 30, 38, 79, 96, 101,
 175, 292, 294, 299, 301,
 416, 420
 Chandrasekaran, B., 348, 350
 Charlton, T. H., 308, 318
 Charniak, E., 200, 202
 Cheeseman, P., 17, 19, 22, 32, 36,
 58, 59, 64, 68, 301, 308,
 319, 363, 385, 386
 Chen, J. J., 314, 319
 Chen, K., 333, 351
 Cheng, Y., 19, 36, 312, 314, 319
 Chi, M. T., 168, 175, 213, 226, 232,
 280, 298, 301, 426, 456
 Chilausky, R. L., 53, 70, 327, 351
 Clark, S. E., 11, 36, 57, 70
 Clarkson, K., 3, 37, 199, 203, 238,
 274
CLASSIT, 28–30, 337, 343, 344, 358,
 367, 370, 372, 373, 375,
 385
CLS, 5–8, 12, 104–107, 115, 116
 clumping, 15, 21, 22, 33, 174
CLUSTER, 136
CLUSTER/2, 19, 31, 131–134, 138,
 141, 308, 333
COBWEB, 19, 20, 26–34, 74, 75, 80,
 98, 121, 127, 138–143, 146–
 148, 151, 152, 157, 279,
 280, 286, 313, 334–338, 340,
 349, 367–370, 372, 375, 385,
 391, 397
 Collins, A., 243, 273, 274
 Collins, G. C., 238, 278
 Congdon, C. B., 435, 457
 connectionist models, 20, 40, 67, 98,
 99, 102, 112, 114, 121, 397,
 427
 constructive induction, 277, 339, 344,
 345, 347, 352
 contradiction, 33, 283, 292
CORA, 30
 Corter, J. E., 26, 38, 57, 69, 85, 86,
 101, 139, 159, 255, 289,
 301, 334, 350, 368, 386
 Croft, W. B., 321
 Crossman, E., 123
 Cultice, J., 57, 69
CYRUS, 24–28, 30, 31, 33, 34, 121,
 127, 138, 317
DÆDALUS, 157, 311–313
 Dale, M. B., 16, 37
 Decaestecker, C., 28, 30, 34, 37
 DeGroot, A. D., 279, 301

- DeJong, G. F., 169, 179, 180, 202, 208, 224, 232, 236, 239, 274, 436, 456
Dewey, G. I., 209, 234, 244, 278
DIDO, 309, 310, 312, 387, 388, 390–392, 394, 395, 397–404, 406–409, 411–420
Dietterich, T. G., 3, 7, 9, 10, 18, 37, 129, 131, 134, 158, 184, 186, 190, 199, 201–203, 238, 240, 253, 267, 268, 274, 283, 284, 286, 292, 301, 314, 387, 390, 391, 420
discriminating, 172, 298, 448
Domeshek, E., 177
Donaldson, M., 438, 456
Dryburgh, R. B., 327, 351
Dubes, R. C., 13, 19, 38, 307, 319
Duda, R. O., 19, 22, 37
Dudycha, A. L., 244, 276
Dunbar, K., 452, 457
Dyer, M., 238, 277

EBG, 281–283, 290–292
Edelson, D., 177
Edelson, S. M., 57, 69, 214, 224, 225, 234
EGGS, 293
Elio, R., 58, 68, 209, 225, 232
Elkan, C., 290, 303
Ellery, I., 299, 303
Ellman, T., 239, 274
EPAM, 23–25, 30, 103, 104, 107–123, 170, 319, 329
Estes, W. K., 74, 100
Etzioni, O., 292, 301, 397, 420
Everitt, B., 12–14, 22, 25, 37, 131, 158, 307, 319
EXOR, 280, 281, 286–292, 294–300, 314, 315, 347

Falkenhainer, B., 155, 158
familiarization, 23, 24, 108, 111, 125
Faries, J. M., 168, 175, 212, 232, 298, 301
Farley, A. M., 312, 318
Farrell, R., 210, 231
Feigenbaum, E. A., 23, 37, 103, 107, 108, 111, 113, 121, 123, 125, 141, 158, 203, 274, 319
Feltovich, P., 168, 175, 213, 232, 280, 301
Fenves, S. J., 33, 167, 313, 314, 323, 338, 352
Ferguson, W., 177
Ferrara, R. A., 439, 456
Field, D., 73, 160, 174, 428, 455, 456
Fikes, R. E., 184, 203
Fisher, D. H., 6–8, 10, 15, 16, 18–21, 26, 28–32, 34, 37, 42, 47, 59, 60, 68, 74, 75, 79, 80, 85, 86, 88, 96, 100, 101, 103, 121–123, 127, 131, 138, 139, 147, 154, 156, 158, 159, 165–167, 169, 171, 173–175, 179, 185, 198, 201–203, 205, 207, 224, 228, 229, 231, 233, 236–238, 240, 242, 243, 245, 253, 255, 267, 268, 273, 274, 279, 284–286, 289, 292, 294, 298, 299, 301, 307, 309–315, 317–319, 322, 329, 334, 336, 337, 345, 347, 349, 350, 357, 358, 363, 367–372, 385–387, 390, 391, 397, 416, 420, 424, 447, 455
Fisher, R. A., 64, 68
Flann, N. S., 186, 190, 201, 203, 239, 253, 267, 268, 274, 283, 284, 286, 292, 301
Flowers, M., 238, 277
FOIL, 315
Forbus, K. D., 155, 158
Freed, M., 177
Freeman, D., 17, 36, 58, 68, 122, 160, 276, 308, 319, 385,

- 386, 457
 Freko, D., 57, 69
 Fried, L. S., 363, 385, 386
 Friedman, J. H., 36, 343, 350
 Fu, K., 19, 36, 314, 319
 Fuson, K. C., 425, 430, 446, 456
 Gearhart, M., 454, 458
 Gelman, R., 426, 428, 430–432, 438, 443, 452, 455, 456
 Gelman, S. A., 66, 68
GENESIS, 191, 193–195, 197–199, 201, 347
 Gennari, J., 10, 23, 28–30, 34, 38, 39, 103, 121, 124, 127, 138, 151, 156–160, 179, 203, 207, 214, 228, 233, 301, 312, 320, 337, 350, 355, 358, 363, 364, 367–370, 372, 382, 385–387
 Gentner, D., 155, 158, 210, 212, 233, 452, 457
 Gero, J. S., 328, 351
 Gilmartin, K. J., 108, 125
 Gilovich, T., 210, 223, 233
 Giordana, A., 299, 300
 Glaser, R., 168, 175, 213, 232, 280, 301
GLAUBER, 22, 167, 169, 309, 315
 Gluck, M. A., 26, 38, 57, 69, 85, 86, 101, 139, 159, 209, 233, 255, 289, 301, 334, 350, 368, 386, 387
 goals, 45, 170, 180, 193, 195, 197, 207, 214, 229, 230, 311, 313, 347, 355, 420, 435, 449, 454
 Goebel, J., 319
 Goodenough, F. L., 244, 274
 Goodnow, J. J., 4, 12, 36, 75, 100, 123, 240, 273
 Granger, R. H., 8, 11, 42, 75, 94, 99, 102, 194, 203, 419, 422
 Gray, W. D., 26, 41, 70, 101
 Gregg, L. W., 42, 108, 112, 123, 422
 Gross, K. P., 390, 421
 Grove, D. C., 308, 318
 Guberman, S. R., 454, 458
 Hadzikadic, M., 28, 30, 34, 38
 Halford, G. S., 425–427, 457
 Hall, R., 38, 125, 161, 299, 302, 319, 352
 Hammond, K. J., 170, 175, 225, 233, 319
 Hampson, S. E., 12, 17, 40, 166, 176
 Handa, K., 156, 159, 167, 175
 Hanson, S. J., 16, 31, 38, 139, 159, 208, 233, 245, 274
 Hardy, J. K., 142, 160
 Harris, D. B., 244, 274
 Hart, P. E., 19, 22, 37, 121, 184, 203
 Hayes-Roth, B., 56, 69
 Hayes-Roth, F., 38, 43, 56, 69, 129–131, 155, 159, 177, 223, 233, 322
 Hayes-Roth, J. R., 223, 233
 Haygood, R. C., 240, 274
 Heit, E., 208, 227, 232, 273
 HIERARCH, 29, 32
 Hinsley, D. A., 223, 224, 233
 Hinton, G. E., 67, 69, 99, 102
 Hintzman, D. L., 74, 101, 120, 123, 209, 233, 238, 245, 274
 Hirsh, H., 283, 299, 302, 303
 Hirtle, S. C., 142, 160
 Hoff, W., 134, 159
 Hoffman, J., 57, 69
 Holder, L. B., 18, 38, 294, 302, 319
 Holland, J. H., 39, 160, 320, 353, 386, 387, 397, 421
 Holte, R., 170, 176, 228, 235, 321
 Holyoak, K. J., 213, 215, 233, 363, 385, 386
 Homa, D., 57, 69
 Hong, J., 351
 Hopke, P. K., 308, 318
 Hovland, C. I., 104, 123

- Huffman S. B., 458
Hunt, E. B., 5, 8, 38, 104, 123
Hunter, L. E., 238, 278, 321
- Iba, G., 292, 302
Iba, W., 10, 39, 124, 157, 158, 160,
312, 320, 355, 357, 358,
382, 385–387
- IBP, 343, 344
- ICARUS, 157
- ID3, 7, 8, 11, 19, 20, 74, 99, 105,
107, 241, 397, 399, 406
- ID4, 8, 10, 12, 23, 24, 31
- ID5, 8, 10, 23, 31
- ill-structured problems, 323–327, 349,
352
- INC, 28, 30
- indexing, 3, 24–26, 34, 107, 135,
139, 144, 148, 193, 200,
201
- INDUCE, 242, 254
- INDUCE/2, 134
- inference, 37, 43, 51, 102, 159, 168,
169, 172, 174, 177, 203,
208, 227, 274, 281, 291,
296–298, 302, 317, 322, 427,
457
- inheritance, 27, 98, 137, 317
- instantiation, 152
- inter-category relationships, 14, 114,
139, 285, 368
- intra-category relationships, 14, 114,
115, 139, 285, 368
- IOE, 283, 284, 286, 287, 297
- IOU, 267, 268
- IPP, 24, 315, 316
- Jacoby, L. L., 209, 210, 233
- Jain, A. K., 13, 19, 38, 307, 319
- Jepson, D., 66, 70
- Johnson, D. M., 26, 41, 70, 101
- Johnston, J. C., 113, 123
- Jona, M., 177
- Jones, G., 26, 38
- Jones, R., 319
- Kahneman, D., 230, 234
- Katz, B., 181, 205
- Kedar-Cabelli, S., 169, 170, 179–
181, 195, 203, 204, 240,
267, 275, 276, 281, 294,
303, 311, 321, 351, 436,
458
- Keele, S. W., 56, 70, 209, 235
- Keil, F. C., 239, 275
- Keller, R. M., 169, 170, 179, 204,
239, 276, 281, 294, 297,
302, 303, 311, 321, 351,
436, 458
- Kelly, J., 17, 36, 58, 68, 308, 319,
385, 386
- Kibler, D., 6, 35, 299, 302, 343, 350
- Kim, I., 299, 303
- Klahr, D., 39, 101, 312, 351, 421,
423, 425–427, 431, 446, 452,
456, 457, 459
- Kline, P., 7, 11, 36, 39, 55, 68, 155,
159, 212, 231, 279, 300
- Knapp, A. G., 209, 234
- Knoblock, C. A., 311, 319
- knowledge-level learning, 184, 186,
194, 202
- Knuth, D. E., 31, 39
- Koh, K., 167, 213, 215, 233
- Koller, G. R., 22, 36, 308, 318
- Kolodner, J. L., 19, 24, 25, 34, 39,
123, 127, 138, 159, 170,
175, 210, 215, 225, 226,
234, 279, 280, 285, 302,
316, 319, 320, 336, 351
- Koppitz, E. M., 244, 255, 275
- Krantz, D. H., 66, 70
- Krulwich, B., 177
- Kubo, S., 332, 353
- Kunda, Z., 66, 70
- LABYRINTH, 122, 127–129, 138, 140–
144, 147, 148, 152–158
- Laird, J. E., 119, 123, 182, 184, 203,
435, 436, 447, 451, 457,
458

- Landers, R., 212, 233
 Langley, P., 6, 7, 9–11, 15, 16, 21–
 23, 28–30, 32–36, 38–40,
 43, 94, 100, 101, 103, 109,
 121–124, 127, 135, 138, 157–
 161, 165, 167, 170, 175,
 179, 195, 202, 203, 207,
 229, 231, 233, 238, 246,
 273–275, 279, 281, 299–302,
 309, 311, 314, 315, 318,
 320, 322, 337, 339, 350,
 357, 358, 363, 364, 369,
 382, 384–387, 397, 415–417,
 419, 421
 Langrana, N. A., 341, 351
 Larson, J. B., 8, 39
 Lassaline, M. E., 26, 39
 LASSY, 293, 294
 latency, 53, 120, 121
 Lavrac, N., 351
 Lea, G., 7, 42, 389, 422
 Lebowitz, M., 19, 24, 25, 28, 29,
 31, 34, 39, 47, 69, 98, 101,
 127, 137, 138, 160, 171,
 175, 201, 203, 208, 234,
 240, 267, 268, 275, 285,
 286, 302, 316, 320
 Lee, P. M., 51, 69
 Lehman, J. F., 458
 Lenat, D. B., 309, 320, 387, 421
 Levi, K. R., 389, 422
 Levinson, R. A., 29, 33, 40, 134–
 136, 141, 160, 316, 322
 Lewis, D. D., 321
 Lewis, M. W., 224, 234
 Lewis, R. L., 423, 435, 440, 458
 Lowry, M., 181, 205
 Lu, S., 314, 322, 333, 351

 MÆANDER, 157, 358, 382, 383
 MacArthur, R. H., 396, 421
 MAGGIE, 382, 383
 Magued, M. H., 327, 351
 Mahadevan, S., 184, 204
 Malasky, S., 300, 302

 Malt, B. C., 209, 222, 234
 Marin, J., 5, 38, 104, 123
 Markovitch, S., 33, 94, 167, 282,
 292, 293, 302, 309, 312,
 387, 388, 390, 396, 420,
 422
 Marr, D., 142, 160, 269, 275
 Martin, J. D., 30, 33, 40, 71, 226,
 227
 Matessa, M., 6, 29, 31, 32, 45, 58,
 80, 98, 214, 229, 237, 245,
 298, 334, 357, 363
 Mayberry, M., 457
 Mayer, R., 281, 286, 297, 302
 McCarty, L. T., 181, 203
 McClelland, J. L., 67, 69, 102, 112,
 114, 121, 124, 125, 209,
 234, 397, 422, 427, 458
 McCormick, W. T., 310, 311, 320
 McDermott, J., 7, 38, 129–131, 155,
 159
 McKusick, K. B., 32, 35, 40, 138,
 158, 160, 337, 385
 McLaughlin, S., 328, 351
 McNamara, T. P., 142, 160
 McNulty, D., 6, 35
 Medin, D. L., 5, 6, 9, 12, 14, 17,
 18, 21, 26, 31, 33, 34, 36,
 39, 40, 42, 53–57, 59–63,
 67–69, 72, 73, 75, 98, 101,
 138, 161, 166, 168, 172,
 174, 176, 208–210, 213, 214,
 224–226, 231, 234, 235, 237–
 239, 241, 242, 244, 245,
 257, 275, 276, 278, 285,
 302, 312, 362, 386, 387
 Mervis, C. B., 5, 17, 26, 40, 41, 70,
 101, 103, 114–117, 125, 160
 Michalski, R. S., 7–12, 16, 18, 37,
 39, 40, 42, 53, 70, 129, 131–
 134, 154, 158–161, 169, 172,
 175–177, 179, 203, 205, 231,
 232, 238, 241–243, 245, 254,
 274–278, 303, 308, 310, 314,

- 320–322, 327, 333, 345, 347, 351, 353, 384, 387, 390, 391, 420, 421
Minsky, M., 269, 276
Minton, S., 182, 184, 203, 204, 282, 289, 292, 302
Mitchell, T. M., 7, 37, 40–42, 72, 76, 101, 154, 158, 160, 169, 170, 175–177, 179–181, 183, 184, 193, 203–205, 231, 232, 238, 239, 243, 245, 275–278, 281, 294, 299, 303, 311, 320–322, 327, 351, 387, 389–391, 397, 414, 421, 436, 458
Mogensen, B. N., 169, 176, 201, 204, 321
Mooney, R. J., 20, 40, 169, 172, 179–181, 187, 192, 193, 202, 204, 208, 224, 232, 235, 239, 240, 242, 267, 274, 276, 280, 285, 293, 303, 327, 347, 349, 424, 436, 447, 456
Morris, S., 308, 321
Mostow, J., 184, 205, 253, 276
Mozetic, I., 351
Muchinsky, P. M., 244, 276
Murphy, G. L., 57, 70, 208, 231, 234, 235, 239, 244, 276, 278
Murphy, T. D., 209, 278
Mustafa, M., 328, 350
Neisser, U., 106, 124, 231, 232
Nelson, K., 166, 176
Neumann, P. G., 74, 75, 101, 209, 235
Newell, A., 118, 119, 123, 124, 182, 203, 312, 347, 351, 423, 435, 436, 440, 445, 451, 457–459
NGLAUBER, 309, 310
Niblett, T., 11, 36, 105, 124
Nilsson, N. J., 184, 202, 203
Nisbett, R. E., 66, 70
Nishihara, H. K., 269, 275
Noordewier, M., 300, 303
Nordhausen, B., 160, 167, 176, 310, 321
Norman, D. A., 269, 276
Nosofsky, R. M., 57, 67, 70, 209, 235, 238, 245, 276
OCCAM, 170–173, 201, 267, 269, 284
Ohmaye, E., 177
Olshen, R. A., 11, 36, 343, 350
operationality, 169, 180–182, 297–299, 301, 302
operationalization, 183, 253, 257, 272
Ortony, A., 213, 226, 233, 234, 236, 239, 275, 277
Osgood, R., 177
Osherson, D. N., 98, 101
Ourston, D., 208, 235, 239, 240, 267, 276
Owens, C., 170, 176, 317, 321
Owings, R. A., 452, 456
OXBOW, 312, 356, 358, 359, 362–383, 385
Palmer, S. E., 176, 269, 276
Papademetriou, C., 155, 161
Patterson, A., 105, 124
Pazzani, M. J., 7, 18, 21, 41, 86, 103, 121, 131, 165, 166, 170, 171, 174, 176, 185, 198, 201, 202, 204, 208, 231, 235, 238, 240, 242, 255, 267, 269, 277, 283, 284, 289, 298, 300, 303, 307, 309–311, 313, 314, 317, 329, 336, 345, 369, 371, 385, 424, 447
perception, 30, 40, 69, 103, 104, 107, 108, 111–113, 122–125, 233, 275, 276, 301, 359, 440
Piaget, J., 424, 425, 451, 452, 458

- Pianka, E. R., 396, 421
 PLANEREUS, 312
 PLOT, 311–313
 PLS, 312, 313
 Polk, T. A., 440, 458
 Porter, B., 170, 176, 228, 235, 321
 Posner, M. I., 56, 70, 209, 235
 Postman, L. P., 124
 predictability, 26, 28, 317
 predictiveness, 26, 28, 289, 317
 Prieditidis, A. E., 184, 205
 PRISM, 397, 415, 419
 Pryor, L., 177
 PURFORM, 191, 192, 195–198, 201
 Q-SOAR, 312, 423, 425, 427, 434, 436–444, 446–455, 459, 462
 Quinlan, J. R., 6, 7, 9, 41, 74, 99, 101, 105, 107, 124, 128, 161, 176, 238, 241, 245, 277, 285, 294, 303, 315, 321, 328, 330, 351, 387, 390, 391, 397, 416, 417, 421, 422
 RA, 317, 322
 Rajamoney, S. A., 239, 240, 277
 Ramachandran, N., 341, 351
 rationality, 9, 49, 298
 Reed, S. K., 56, 70, 210, 235
 Reich, Y., 33, 167, 313, 314, 323, 337–340, 349, 352
 Reiser, B. J., 168, 175, 212, 232, 298, 301
 Reitman, W. R., 324, 352
 relevance, 21, 166, 168, 220, 293, 294, 311
 Rendell, L., 155, 161, 277, 312, 321
 RESEARCHER, 29, 315, 316
 Richards, W. A., 41, 161, 366, 386, 387
 Richman, H. B., 5, 23, 30, 103, 108, 109, 111, 113, 114, 124, 329
 Riesbeck, C., 198, 205
 Rips, L. J., 114, 124, 239, 271, 277
 Romesburg, H. C., 19, 41, 308, 310, 321
 Rosch, E., 5, 17, 26, 40, 41, 57, 70, 98, 101, 103, 114–117, 124, 125, 160, 209, 232, 236
 Rosenbloom, P. S., 119, 123, 124, 182, 203, 435, 436, 451, 457, 458
 Ross, B. H., 166, 168, 176, 207, 209, 210, 213–218, 220, 224, 226, 232, 234, 236, 280, 298, 303
 Rubin, J. M., 142, 161, 366, 386, 387
 Rumelhart, D. E., 20, 41, 67, 69, 99, 102, 112, 114, 121, 124, 125, 209, 234, 269, 276, 397, 422
 Russell, A., 169, 290, 295, 297, 301, 303
 Sacerdoti, E. D., 311, 321
 Sage, S., 21, 39
 SAL, 120
 Sauers, R., 210, 231
 Saxe, G. B., 454, 458
 Schaffer, M. M., 6, 40, 53–57, 59–63, 67, 69, 72, 73, 75, 101, 209, 210, 224, 234, 238
 Schank, R. C., 170, 176, 177, 193, 198, 205, 210, 226, 236, 238, 278
 Schlimmer, J. C., 7, 8, 10, 11, 42, 75, 94, 99, 102, 167, 177, 342, 419, 422
 Schmidt, R. A., 359, 386, 387
 Schulenburg, D., 208, 235, 238, 277, 308, 321
 Schwanenflugal, P. L., 238, 275
 Schweitzer, P. J., 310, 320
 scientific discovery, 22, 36, 160, 167, 175, 307–309, 319, 320, 322, 452, 457

- Scott, P. D., 33, 94, 167, 238, 277, 282, 292, 302, 309, 312, 345, 387–390, 396, 420, 422
Segre, A. M., 184, 205, 290, 303
Seibel, R., 119, 125
Seifert, C. M., 168, 170, 172, 173, 177, 273, 298, 303
Self, M., 17, 36, 40, 58, 68, 134, 160, 308, 319, 322, 326, 363, 385, 386
Seshu, R., 312, 321
Shaffer, M. M., 275
Shah, A., 341, 351
Shalin, V. L., 389, 422
Shannon, C. E., 121, 125, 394, 418, 422
Shastry, L., 98, 102
Shavlik, J. W., 37, 184, 205, 224, 236, 240, 267, 278, 300, 303
Shelton, T. S., 452, 456
Shen, W. M., 310, 321, 391, 422
Shiavi, R. G., 314, 319
Shoben, E. J., 98, 101, 115, 124, 222, 236
Shrager, J., 36, 160, 177, 452, 458
Siebler, R. S., 426, 427, 446, 455, 459
Sigleo, A. C., 308, 322
Siklóssy, L., 125
Simon, H. A., 3, 7–9, 22, 23, 31, 37, 42, 107–109, 111–114, 118, 121, 123–125, 167, 175, 195, 223, 233, 298, 303, 309, 319, 320, 324, 325, 351, 352, 389, 422, 445
Simon, T., 312, 423, 458, 459
Simpson, R. L., 210, 215, 225, 234
Sjoberg, K., 426, 459
Sleeman, D., 299, 303
SME, 156
Smith, E. E., 5, 6, 33, 42, 57, 70, 98, 101, 115, 124, 138, 161, 209, 222, 234, 238, 273, 278, 362, 386, 387
Smith, R. G., 240, 278
Smith, S., 457
SNPR, 22, 315
SOAR, 423, 425, 434–436, 439, 449–451, 455, 457, 458
Spalding, T., 166, 168, 217, 218, 236, 280, 298
SPROUTER, 129–131, 133, 136, 140, 148, 155
STAGGER, 99
Steels, L., 323, 352
Steiglitz, K., 155, 161
Stein, B. S., 452, 456
Steinberg, L. I., 184, 204, 341, 352
Stepp, R. E., 15, 16, 18, 19, 41, 42, 131, 133, 134, 159–161, 169, 172, 176, 177, 179, 201, 203, 205, 308, 310, 314, 320, 322, 333, 345, 347, 351, 353, 384, 387, 390, 391, 421
Stone, C. J., 11, 36, 343, 350
Stone, J., 327, 353
Stone, P., 38, 104, 123
STRIPS, 311
Stutz, J., 17, 36, 58, 68, 308, 319, 385, 386
Subramanian, D., 167, 177
Svenson, O., 426, 459
Swaminathan, K., 317, 322
Swedlow, K. R., 435, 457
Taura, T., 332, 353
Taylor, W., 17, 36, 58, 68, 308, 319, 385, 386
Tcheng, D., 312, 321
Tessler, S. G., 458
Thau, R., 226, 234
Thompson, K., 29, 109, 122, 124, 127, 138, 157, 160, 165, 167, 314, 320, 337, 339, 364, 369, 382, 384–387
THOTH, 18, 19, 312

- Towell, G. G., 240, 267, 278, 300, 303
- Turnbull, W., 271, 277
- typicality, 5, 6, 12, 17, 28, 34, 103, 113–115, 117, 119, 121, 122, 274, 277
- uncertainty, 38, 69, 101, 159, 301, 350, 386, 387, 390, 393, 394, 396, 398–400, 417–419
- UNIMEM, 24–31, 33, 34, 39, 98, 127, 137–139, 171, 175, 201, 267, 286, 317, 320, 340
- UNIVERSE, 316
- Utgoff, P. E., 6, 8, 10, 42, 43, 154, 160, 191, 205, 238, 278, 281, 299, 303, 311, 321, 416, 422
- Vere, S. A., 7, 18, 19, 21, 43, 129, 130, 155, 161, 165, 177, 312, 322
- Vogt, R. C., 389, 390, 396, 422
- Volk, K., 319
- Waldinger, R., 196, 205
- Wallace, J. G., 425–427, 446, 457, 459
- Waltz, D., 321
- Warren, D., 289, 303
- Wasserman, K., 29, 43, 136–138, 141, 142, 161
- Watanabe, L., 155, 161
- Wattenmaker, W. D., 9, 17, 40, 166, 176, 238, 239, 241, 242, 244, 245, 275, 278
- Weaver, W., 394, 422
- Weene, P., 106, 124
- Weinberg, J. B., 22, 32, 36, 279, 294, 299, 301, 308, 312, 318
- White, T. W., 81, 88, 239, 241, 252, 310, 320, 410
- Whitehall, B. L., 314, 322
- WHOLIST, 5, 7
- Wilcox, C. S., 316, 322
- Wilensky, R., 170, 174
- Wilkins, D., 279, 303
- Williams, R. J., 99, 102
- Wilson, E. O., 396, 421
- Wilson, S. W., 416, 422
- Winston, P. H., 10, 43, 129, 155, 161, 181, 205, 240, 245, 276, 278, 387, 390, 391, 422
- Wisniewski, E. J., 26, 39, 168, 177, 208, 235, 237, 240, 244, 278, 280, 312, 389, 422
- WITT, 16, 17, 22
- Wogulis, J., 29, 33, 43, 135, 161, 322
- Wolff, J. G., 22, 43, 315, 322
- Wu, S., 208, 235
- Wu, T. D., 314, 322
- Yang, H., 170, 175, 224, 311, 322
- Yang, Q., 22, 36, 308, 318
- Yoo, J. P., 169, 170, 173, 175, 201, 205, 224, 229, 243, 253, 267, 268, 279, 314, 315, 347, 349
- Yoshikawa, H., 332, 341, 353
- Yun, D., 30, 34, 38
- Zhang, G., 118, 125
- Ziarko, W., 328, 350
- Ziessler, C., 57, 69
- Zytkow, J. M., 22, 167, 175, 309, 320

DATE DUE / DATE DE RETOUR

MAR 25 1996
MAR 25 1996

OCT 16 1997

JUN 21 2002

JUN 11 2002

JAN 15 2004

CARR MCLEAN

38-297

TRENT UNIVERSITY



0 1164 0127642 7

