AD-A239 320

DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		THESIS/~~DISSERTATION~~

4. TITLE AND SUBTITLE
Opportunistic Constructive Induction: Using Fragments of Domain Knowledge to Guide Construction

5. FUNDING NUMBERS

6. AUTHOR(S)
Gregg H. Gunsch, Major

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT Student Attending: University of North Dakota

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/CI/CIA- 91-005D

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFIT/CI
Wright-Patterson AFB OH 45433-6583

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release IAW 190-1
Distributed Unlimited
ERNEST A. HAYGOOD, 1st Lt, USAF
Executive Officer

12b. DISTRIBUTION CODE

DTIC
ELECTE
AUG 0 9 1991
S D
D

13. ABSTRACT *(Maximum 200 words)*

91-07349

91 8 08 069

14. SUBJECT TERMS	15. NUMBER OF PAGES 227
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

PII Redacted

OPPORTUNISTIC CONSTRUCTIVE INDUCTION: USING FRAGMENTS OF DOMAIN KNOWLEDGE TO GUIDE CONSTRUCTION

BY

GREGG HAROLD GUNSCH

B.S., University of North Dakota, 1979
M.S., Air Force Institute of Technology, 1983

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1991

Urbana, Illinois

ABSTRACT

One subfield of machine learning is the induction of a representation of a concept from positive and negative examples of the concept. Given a set of training examples, the goal of the inductive system is to create a description capable of classifying the training examples, yet general enough to accurately predict the classification of unseen examples. Often the original attributes describing the instances are inadequate to capture important regularities in the concept. New descriptors, constructed through the application of operators to the original attributes, can provide the proper vocabulary to create concise concept representations at the right level of generalization to be highly predictive. Constructive induction is the process of generating and applying new descriptors during inductive learning.

The large number of possible constructive operators and combinations of attributes defines an enormous search space for the inductive process. Knowledge about the concept or problem domain can be used to guide the construction of new descriptors. This thesis lays the foundation of *opportunistic constructive induction* in the context of decision-tree assembly, providing a framework for dynamically applying fragments of knowledge to produce potentially useful descriptors or *hypotheses*. A two-staged process of generating candidate descriptors (hypothesis generation) and focusing the induction on the most

25 * Construction, * Fragments, * Learning, * Test Construction (Partially) Theses.

promising (hypothesis ordering) has been developed and partially implemented. This thesis concentrates on the development of a hypothesis ordering mechanism that incorporates the evaluation of multiple objectives to identify the most promising descriptors. Experiments in four test domains demonstrate the hypothesis ordering mechanism to be a robust, effective method of significantly reducing the potential computational burden created by prolific hypothesis generation. In addition, preliminary investigation of hypothesis generation indicates that small amounts of knowledge can provide substantial increases in the predictive accuracy of the induced decision-trees.

DEDICATION

To my father, Harold Gunsch, for having so much faith in
me he would sacrifice all those years to get me started.
I hope I turned out all right, Dad.

ACKNOWLEDGMENTS

In an endeavor such as this, many people contribute to its success in ways which they are often unaware of and, unfortunately, which go by unacknowledged. At the risk of omitting some people I am indebted to, I would like to take this opportunity to extend my appreciation to those who helped me the most.

I would first like to thank Professor Larry Rendell for being my advisor, mentor, friend and most gentle critic. He was a constant source of encouragement and assistance, always available and willing to help. He and I covered a lot of interesting ground over the short time we worked together, all of it stimulating. He also provided the wealth of computing resources for this research, supported in part by grant IRI 8822031 from the National Science Foundation.

I would also like to thank my committee members Professors Narendra Ahuja, Mehdi Harandi, and Benjamin Wah for their insights, recommendations, support, and tolerance of my scheduling constraints. A candid committee makes the job tougher, but the final product is well worth the effort.

Thanks also to the members of the Inductive Learning Group who formed a dynamic, intelligent and refreshingly controversial sounding board for the discussion of my ideas. The group includes Powell Benedict, Gunnar Blix, Daniel Dannenberg, Anita Govindjee,

Chris Matheus, Eduardo Perez, Harish Ragavan, Jay Scott, Raj Seshu, Larry Watanabe, and Der-Shung Yang. Gunnar Blix deserves a special thanks for his constant involvement as a critic, reader, and friend.

No acknowledgment would be complete without recognizing the importance of support and encouragement from one's family. My wife, Cherry, and children, Jason and Ginnie, were constant sources of joy: refreshing oases to return to after hard days (and nights) of work. The children deserve special citations of excellence for their patience and cooperation during this time. Somehow they knew when to be quiet and stay away from the office. The debt of gratitude I owe Cherry goes far beyond words: I would just like to thank her now for holding me together through this program.

This opportunity to pursue my doctoral education was made possible through the sponsorship of the Air Force Institute of Technology. Although there are not specific people I can single out to thank for offering this program to me, I am grateful to the United States Air Force in general for the financial and physical security it has provided my family over the years, and the constant exposure to cutting-edge technologies it has given me.

Finally, my deepest appreciation goes to my Lord God. Father, You have given me so much these past few years. You gave me the opportunity to pursue this education even when it looked like the Air Force wanted me elsewhere. You handed me a great university on a silver platter, and timed it so I could be a part of the excitement of the new Beckman Institute. You surrounded me with the caring, supportive congregation of Windsor Road Christian Church, and then You led me to a good advisor and friend, Larry. You were always faithful, even when I was unfaithful to You, became distracted, procrastinated, or was overwhelmed. At those times You never failed to shower me with people who had more confidence in me than I had in myself. I noticed, Lord, and I thank You. Yes, Lord, Your grace is sufficient for me (II Corinthians 12:9).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

One of the hallmarks of an intelligent system, whether a person, animal or machine, is the ability to learn and adapt. Without learning, even the most clever system is doomed to repeat mistakes or perform inefficient processes over and over. Learning is a broad topic, encompassing many methods and objectives. This research focuses on one popular aspect of automated symbolic learning: the acquisition of new and useful concepts by computer through induction over examples. *Inductive concept learning*, herein known simply as *inductive learning*, seeks to find or create a description of a concept represented by positive versus negative examples, and to generalize that description to attempt to correctly predict the classification of examples not seen in the training set.

A *concept* learned under the conditions noted above is an intensional description of a class of objects [Hunt *et al.*, 1966, Matheus, 1989], i.e., a condensed, nonenumerated description of the members of a particular class. This thesis concentrates on *discriminant* concept learning: the creation of a description intended to identify members of the concept and discriminate them from nonmembers (as opposed to *characteristic* concept learning which describes the commonalities of members within the class [Dietterich and Michalski, 1983]). The form of concept representation used in this thesis is the decision tree, and the concepts considered will be limited to those capable of being represented by a decision-tree.

Definitions: Appendix A provides definitions of many of the more important and specialized words appearing throughout this thesis. These words typically appear in italics in the text and are usually accompanied by short elaborations.

1.1 Inductive Learning

In its simplest form, inductive learning formulates a concept description from the attributes used to describe the training examples or *instances*. Several authors freely interchange the terms *attribute* and *feature*, meaning, in both cases, the variables with which the learning situation is described. In this thesis I maintain a strict distinction between the two, limiting a *feature* to be a Boolean attribute whose possible values can be only "true" or "false." (See Appendices A and B.)

A common form of inductive learning, similarity-based learning, incorporates the assumption that instances sharing similar attributes are likely to be members of the same class; i.e., the positive examples of the concept are similar to each other and distinct from the negative examples. *Selective induction* uses the attributes as the dimensions delineating an instance space (the space of all possible examples), and seeks to find a boundary capable of separating the positive from the negative examples. Often, however, the examples cannot be so cleanly partitioned. Instead, subregions of the instance space are isolated and described, and these partial descriptions are disjunctively joined to create the full concept description [Hunt *et al.*, 1966, Dietterich and Michalski, 1983, Breiman *et al.*, 1984].

Even with this divide-and-conquer approach, many interesting problems provide examples so scattered in the instance space that the resultant disjunctive description (if found at all) is very large and unwieldy. Although such descriptions provide good accuracy on the training examples, they often do not provide a communicable and understandable description, and can be highly inaccurate on the unseen examples. In these problems, the current attributes are inadequate to represent the concept. New attributes must be created to provide a language in which accurate, concise concept descriptions may be obtained [Soloway and Risemen, 1977, Dietterich and Michalski, 1983, Breiman *et al.*, 1984, Rendell and Seshu, 1990]. One approach to creating new attributes is to construct them

from the existing attributes. When these new attributes are constructed with the intent of using them to redescribe the instance space and perform induction over the examples in this modified space, the selecti induction approach takes on a new level of complexity and is known as *constructive induction* [Dietterich and Michalski, 1983].

One example of the need for constructive induction can be illustrated with a physics problem of the kind solved by the system BACON [Langley *et al.*, 1986]: pressure times volur e equals a constant ($P \times V = C$). Figure 1.1(a) shows a two-dimensional projection of instance space, highlighting the two attributes of interest and displaying a set of training examples. Selective induction would produce a desc iption like that shown in Figure 1.1(b) if high accuracy was desired, or Figure 1.1(c) if conciseness was i nportant. Yet neither description captures the essence of the true concep' BACON provides an iterative method capable of constructing the form ($attributeA^X \times attributeB^Y$) and is able to discover the required attribute ($P \times V$). When this new attribute is added to the instance space as a new dimension, the discovery of the compact concept description

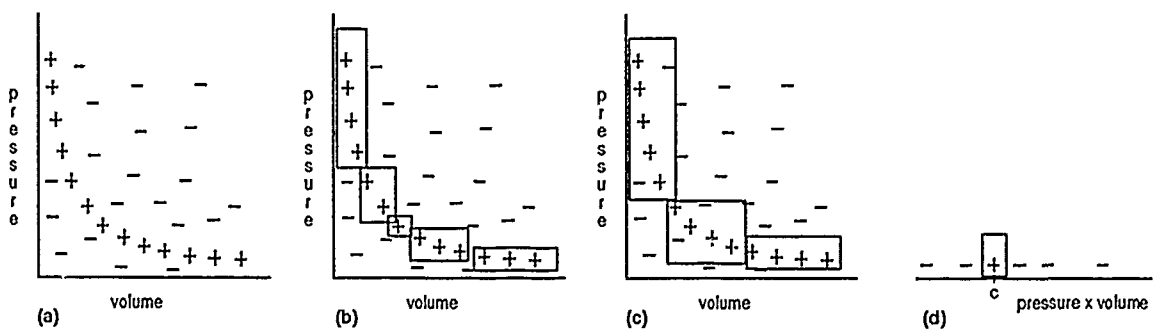


Figure 1.1 Induction of $P \times V = C$. Example physics problem where pressure times volume is a constant: (a) shows the two-dimensional projection of the examples onto the interesting attributes, (b) illustrates the partitioning for high accuracy, (c) illustrates the partitioning for a concise description, and (d) shows the one-dimensional projection over the constructed attribute, providing both conciseness and accuracy.

becomes a trivial selective induction exercise. Figure 1.1(d) shows the one-dimensional projection of the instance space over the new attribute.

Constructive induction generally proceeds as follows: first, the need for new attributes is determined (typically, when the given attributes are deemed inadequate [Matheus, 1989]), and second, new attributes are constructed and tested. This process is repeated as needed, possibly constructing new attributes from previously constructed attributes. Generating only the most useful or promising new attributes is a formidable task. The number of constructive operators available for potential use is large and their types are varied; for example, Boolean (AND, OR, XOR), mathematical ($+$, $-$, $\times$, $\div$, $\sqrt{\ }$), relational (on-top-of, a-part-of, greater-than), and generalizing (drop condition, close interval, spatially transform). Given the large number of operators and the arity of the construction (combining two, or three, or five attributes for example), the number of possible constructed attributes is huge. Selecting the most promising constructions from the space of possible combinations is difficult at best.

Many researchers approach the task of constructing new attributes by imposing strict limits on the allowable constructive operations and then restricting the class of problems made available to the system. While this approach has recognizable research value, these restrictions must eventually be lifted for intelligent systems to learn effectively in the real world; therein lies the motivation of this thesis. Its central theme is to establish a mechanism for managing nearly unrestricted access to potential new attributes through the *opportunistic* application of any information available to the system, particularly knowledge about the domain.

1.2 Thesis Overview

This thesis is concerned with the induction and generalization of decision-trees during single concept, supervised learning; i.e., given a set of training instances labeled by a teacher as positive or negative examples of the concept being learned, a decision tree is assembled to distinguish the positive examples of the concept from the negative. The decision-tree is also generalized during induction in an attempt to predict the classification of previously unseen instances.

The original description language used to describe the training instances is not always adequate to construct a sufficiently accurate, compact, or understandable decision-tree. This thesis proposes and studies a mechanism for making use of available knowledge to suggest constructions from the original attributes to promote the assembly of better decision-trees. *Opportunistic constructive induction* applies fragments of knowledge to dynamically suggest constructions, whenever it appears the knowledge might be useful based on the current state of concept induction. The knowledge may be suggested by the user, compiled in specialized procedures, stored as a domain theory, or learned automatically from previous inductive tasks.

1.2.1 Importance of the problem

A significant trend in current and computer technology is the transition of increasingly complex computer-based applications (e.g., robotics) from the laboratory to the real world. This trend requires a much greater amount of autonomy in those systems, since the real world is rife with novel situations and exceptions to every rule. The most powerful systems will be those that can analyze previously unfamiliar situations, elicit the important patterns, and adapt for future encounters. The process of extracting important or useful patterns from the input data (whether the data are sensory, retrieved

from a database, or input by a human as a problem) is one of manipulating the original data description language in such a way as to form combinations of the language's terms to provide significant portions of the sought after patterns. These combinations of terms are descriptive extensions to the language; hence, the process is one of discovering and incorporating *new terms* into the description language.

The problem of selecting new terms has been recognized since the early days of machine learning research. Referring to the selection of terms to use for evaluating board positions in his checkers playing program, the machine learning pioneer Arthur Samuel [1959] wrote:

> It might be argued that this procedure of having the program select new terms for the evaluation polynomial from a supplied list is much too simple and that the program should generate terms for itself. Unfortunately, no satisfactory scheme for doing this has yet been devised.

After over three decades, the problem stills exists. Although mechanisms have been developed to guide the creation and selection of new terms in very specific applications, the approaches are all limited in scope and explore only small segments of the potential space of new terms. Dietterich and Michalski [1983] wrote:

> An important problem is the development of efficient mechanisms for guiding the process of constructive induction through the potentially immense space of possible derived descriptors.

More recently, Matheus [1989] reiterated what all researchers in constructive induction know regarding the enormity and complexity of the problem of finding new terms:

> The overall result is that the search for an appropriate set of features is intractable in the general case. Solutions to this problem therefore must rely on powerful heuristics.

The primary contribution of this thesis is the development of an approach to allow the treatment of available fragments of domain knowledge as heuristics in the search for new terms, and to apply them in a manner that utilizes the best ones and prevents the poor ones from impeding the process.

1.2.2 Objectives

This research effort provides the framework for incorporating fragments of domain knowledge to propose potentially useful new terms or *hypotheses* during constructive induction. The use of domain knowledge promises to be an effective means of tractably guiding the search through a vast hypothesis space in a nearly best-first fashion. The primary objective of this research effort is to develop a mechanism to allow the use of domain knowledge in an unrestricted, dynamic fashion while maintaining a manageable computational load. Achievement of this primary objective entails the accomplishment of the following secondary objectives:

1. Penetrate the mechanics of induction to incorporate flexible search guidance that responds to the needs of the problem at hand (versus the imposed inflexible biases usually found in inductive systems). This includes an interleaving of deductive and inductive mechanisms to provide the necessary *opportunism*, i.e., to allow the search guidance to be suggested during concept induction as promising opportunities arise. (Chapter 2)

2. Develop a modular system architecture to implement the four components of the opportunistic constructive induction process: hypothesis generation, hypothesis ordering (focusing), hypothesis evaluation, and hypothesis incorporation. An overview of the process is shown in Figure 1.2. The implemented system is intended

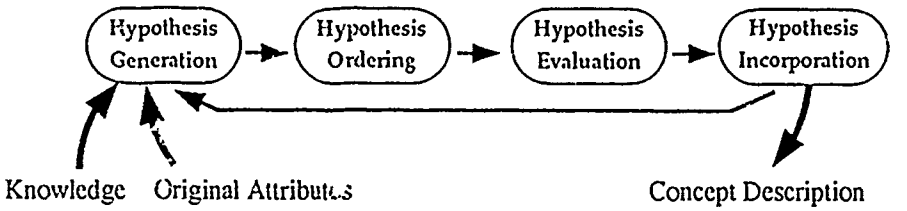


Figure 1.2 Overview of the Opportunistic Constructive Induction Process.

to provide a domain-independent development and testbed environment for the application of domain knowledge to guiding decision-tree construction. (Chapter 3)

3. Develop a robust hypothesis ordering mechanism to manage the potentially enormous computational burden produced by the uninhibited exploration of hypothesis space. (Chapter 4)

4. Develop a robust hypothesis generation mechanism that uses fragments of domain knowledge and an assessment of the current state of the inductive process to explicitly create new hypotheses intended to enrich the concept description language for further induction. (Chapters 3 and 6)

This thesis accomplishes secondary objectives 1 through 3 and explores several aspects of secondary objective 4. The novel machine learning aspects and contributions of this research are:

- The creation of a conceptual framework for the inductive process that encourages the incorporation of deductive processes using background knowledge to suggest or provide fragments of the concept description, taking much of the mystery out of the *abductive* process, i.e., the process of creating good hypotheses [Watanabe, 1985] (Chapter 2). This framework maps directly into an implementation architecture integrating inductive and deductive mechanisms.

- The investigation of the use of a *hypothesis ordering* mechanism to act as a filter between the generation and test phases of new term creation. Included in the investigation is the development of a competitive mechanism that uses small samples of the training data to focus the system's attention on the most promising hypotheses and to reject the most useless.

- Exploration of several *multiple-objective evaluation* functions as the basis of hypothesis ordering, and establishment of the *non-dominance* method as a robust and computationally practical approach.

- The establishment of the groundwork for a diverse and flexible *hypothesis generation* mechanism, including the specification of a baseline capability, and experimentation with several applications of knowledge (described below).

1.2.3 Limitations of this research

Since the phase of research described in this thesis is a portion of a larger research project, the current implementation and completed experiments are limited. Chapter 7 discusses several areas of continuation and expansion of this research.

The four-part inductive process shown in Figure 1.2 also represents the degrees of difficulty of the components and the implementation dependencies. The leftmost components, although more interesting, are more difficult and require the implementation of the rightmost components before they can be properly investigated. Hypothesis evaluation and hypothesis incorporation are relatively well understood and readily implemented. Therefore, this thesis focuses on the development of the hypothesis ordering mechanism to provide a foundation for future development of the hypothesis generation capability. The current implementation consists of the overall system architecture, baseline hypothesis ordering and hypothesis evaluation modules, and partially developed hypothesis

incorporation and central blackboard mechanisms. The hypothesis generation component has been implemented only to the extent necessary to perform controlled experimentation on the effectiveness of certain pieces of domain knowledge. The generator has not yet been developed to the degree necessary to become an autonomous and integral part of the system.

The opportunistic capabilities of a completed system architecture were not fully exercised in the experiments : that is, knowledge was not invoked to generate additional hypotheses based on a partial or tentative concept description. However, the experiments did examine several variations on the type of knowledge used for hypothesis generation and the stage of induction in which the knowledge was applied. In Chapter 6, contextual knowledge (knowledge about content) was used by a post-processing procedure to constrain the conclusions drawn from a previously learned decision-tree, i.e., the knowledge guided learning from experience. This combination of expectation and experience produced a small set of useful hypotheses for subsequent learning sessions. In other experiments (Chapters 5 and 6), syntactic (structural) and contextual knowledge was used by procedural mechanisms to generate hypotheses prior to decision-tree induction. Hypotheses generated with proper knowledge provided the means for extremely rapid convergence to the concept description with small sample sizes. These experiments demonstrate the feasibility and flexibility of applying domain knowledge for hypothesis generation.

1.2.4 Thesis organization

Chapter 2 provides the background and motivation of this research, focusing on the need for incorporating knowledge in the constructive induction process. Chapter 3 presents an overview of the implementation of OXG.le, a testbed for opportunistically utilizing knowledge during constructive induction. The most critical component

of OXGate in this phase of research, hypothesis ordering, is described in Chapter 4, followed by the experimental support and analysis of its viability in Chapter 5. Experiments with selected applications of knowledge are presented in Chapter 6, providing inroads into hypothesis generation, the most novel component of OXGate and next logical area for continued research. Chapter 7 summarizes the important results and contributions of this thesis, and suggests areas of future research. Appendix A provides the definitions of many of the terms used throughout this thesis. Appendix B presents a formalism for describing hypotheses, decision-trees, feature and hypothesis construction, and the application of knowledge. Descriptions of the concepts and data used in the experiments are presented in Appendices C through F.

CHAPTER 2

BACKGROUND AND MOTIVATION

This chapter presents a four-part model of the inductive process and an approach to guiding induction toward a concept description through the application of knowledge. An overview of opportunistic constructive induction is presented next, followed by descriptions of several established constructive induction systems that form the foundation for this research.

2.1 A Framework for Analyzing Induction

The end result of inductive learning is a description known as a *hypothesis*, indicating it is an approximation of the true concept[1] and subject to change should future examples indicate it is incorrect. A hypothesis is a statement relating descriptive attributes of a concept to values those attributes assume in examples of the concept (Section B.2). The simplest form of a hypothesis is an attribute-value pair: "attribute A has value V."

The common definition of hypothesis, the end product, can be extended to include any candidate testable statement that may be used as a component in the formation of the concept description. Generally, a hypothesis may be a relatively complex description constructed from previously established hypotheses, which may themselves be constructed from still simpler hypotheses, allowing for the incremental or piecewise instantiation of the internal attribute-value pairs.

[1] A hypothesis is an approximation of the true concept except when the training set is exhaustive or otherwise completely represents the set of all possible instances. In such cases, the hypothesis is provably correct and, therefore, no longer a hypothesis.

The framework for viewing inductive learning is derived from Satosi Watanabe [1985]. Watanabe's term *inductive learning* can be interpreted to encompass constructive induction, with the distinction between the common usage of these terms to be only the manner of generating candidate hypotheses.

Watanabe's definition:

> Inductive Process = Abduction + Induction Proper
>
> where *Abduction* is the process of creating hypotheses,
> and *Induction Proper* is the experimental assessment of
> those hypotheses, i.e., confidence-building, not validation.

Expanding this definition, I refine the abductive process into two components: the generation of hypotheses and the ordering of hypotheses for testing.

> Abduction = Hypothesis Generation + Hypothesis Ordering

Hypothesis generation could be termed *abduction proper* in the manner of Watanabe. It is the creation of candidate hypotheses by whatever means available. In the abstract, where abduction is distinct from deductive processes [Watanabe, 1985], the creation of useful hypotheses is *extralogical* and *extra-evidential*: it has a mystical quality where good hypotheses are simply "pulled out of thin air." A more pragmatic view allows the systematic generation of candidate hypotheses through controlled relaxation of a restricted description language (a typical constructive induction approach), augmented by the opportunistic utilization of available domain knowledge to suggest and retract candidate hypotheses.[2]

[2]Watanabe describes this and all mechanized processes as purely deductive operations, guided by heuristics provided by humans. He uses this distinction to support his claim that computers cannot perform induction, since the creation of the heuristics is itself an extralogical task, a computer is incapable of induction because it is incapable of true abduction. Philosophically, he may be correct, but such a discrimination does not aid in endowing the computer with the ability to learn, even if "only" a mechanized approximation of human inductive ability.

Any system that grows the set of candidate hypotheses has the potential of becoming overwhelmed by the expense of testing them. Previous machine learning systems avoid this problem by incorporating biases into hypothesis generation that severely constrain the space of possible hypotheses, thereby limiting the number of hypotheses the system must entertain. My approach is not to limit the growth of the set of candidate hypotheses produced through hypothesis generation, but rather to contain the effects of this growth through *hypothesis ordering*. Hypothesis ordering is used to identify the most promising of the candidate hypotheses for testing against the instance data (induction proper), and to reject or suppress those deemed useless. Only the most promising of the hypotheses are tested, thereby reducing the expense of evaluation. While hypothesis ordering is a heuristic, beam-search approach, and therefore does not guarantee perfect filtering (the correct hypothesis may not be in the set of most promising hypotheses), it is a necessary counterpart to the opportunistic hypothesis generation method. A major contribution of this research shows that hypothesis ordering is an effective means of managing uninhibited hypothesis generation.

Continuing to refine Watanabe's definition, I also split induction proper into two components: hypothesis evaluation and hypothesis incorporation.

Induction Proper = Hypothesis Evaluation + Hypothesis Incorporation

Hypothesis evaluation is the testing of a hypothesis against the set of classified training instances. Any of several evaluation functions may be used, the goal being to rate the hypothesis by some measure of credibility or "goodness." After all the most promising hypotheses have been tested, the best one is selected for incorporation. *Hypothesis incorporation* involves propagating the effects of hypothesis selection through the training data and assessing whether the learning task has been satisfactorily completed. For example, in decision-tree learning, incorporating a hypothesis means splitting the set of training instances into two subsets: those covered by the hypothesis and those not.

Associated with the two data subsets are the applicable subsets of the original set of candidate hypotheses. For instance, if the hypot is (*color* = *red*) were chosen, any other hypothesis regarding color need not be considered for the covered subset of instances: these can be removed. In conventional decision-tree learning, the effects of hypothesis incorporation end here, and the program simply begins induction over the subsets. In the *opportunistic constructive induction* system, however, hypothesis incorporation initiates the process in which domain knowledge is invoked in response to a change in the state of the system. It "closes the loop," providing a means of affecting the next stage of hypothesis generation. The four-step process of hypothesis generation, ordering, evaluation and incorporation (Figure 2.1) proceeds cyclically until a satisfactory concept description has been obtained.

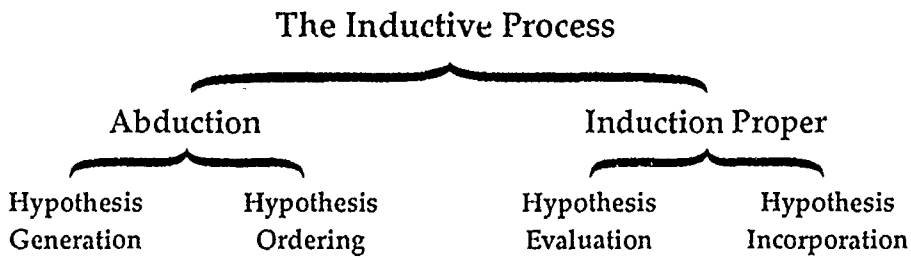


Figure 2.1 The Inductive Process.

2.2 Bias and the Use of Domain Knowledge

The inductive learning problem is to search through hypothesis space for a description capable of distinguishing the positive from negative examples of a concept, and to generalize that description to predict correctly the classification of unseen examples. When the positive and negative examples are sufficiently intermixed in instance space,

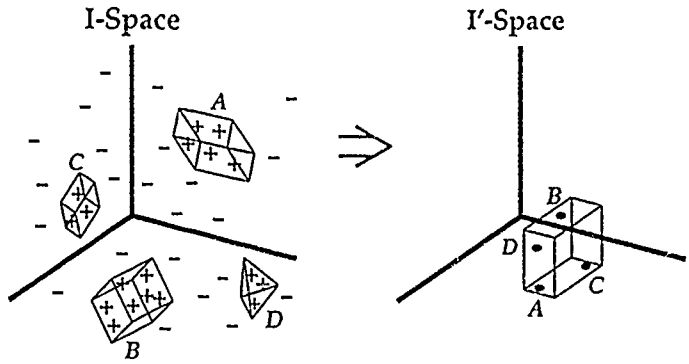


Figure 2.2 Transforming Instance Space. The original instance space (I-Space) is pictured here with a disjunctive collection of islands (conjunctive components or other structures) separating positive from negative instances. Constructive induction transforms I-Space into I'-Space by finding the relationships between the islands and defining new descriptive terms (dimensions in I'-Space), allowing easier concept induction and a more compact description.

a complex description is needed to isolate the islands[3] of positive examples from the negative background (or vice versa) and relate them in a more global fashion to each other (Figure 2.2). This building up of a complex description from simpler island descriptions, constructive induction, can be viewed as the transformation of the original instance space into a simpler space in which inductive learning can be accomplished over the islands themselves rather than the examples. Generalization of the description relating the islands provides the prediction of unseen examples, as well as the prediction of unseen islands of examples. In such a view, the hypothesis space becomes one where not only are all possible islands describable, but all combinations and transformations of those descriptions are representable as well. The mechanism needed to tame this space and make it amenable to productive search is called *bias* [Mitchell, 1980, Utgoff, 1986].

[3]Closely related to the notion of *peaks*, which are regions of similar class membership. [Rendell, 1989]

2.2.1 The need for bias

Bias is the preference of certain areas of hypothesis space over others. In other words, given the set $\mathcal{H}$ of all possible hypotheses, a bias restricts the set of hypotheses accessible to the learning system to a subset $\mathcal{H}_b$. The purpose behind using biases in inductive learning is to regulate the search through hypothesis space with the aim of finding a sufficiently correct hypothesis early in the search. Often, biases are viewed as the application of *extra-evidential* information: information not contained in the set of training examples. A more flexible definition of bias does not require extra-evidentiality, but includes any mechanisms for establishing a preference in hypothesis space, regardless of the means of managing or invoking such mechanisms. Biases are indispensable components of any inductive learning approach, simply because of the vast size of hypothesis space. For example, in an instance space where examples are describable by five Boolean attributes, there are only 2^5 or 32 distinct instances. Yet these instances can be combined to represent 2^{2^5} or 4.3 billion possible concepts. In an instance space with nominal, integer or real dimensions, the number of possible hypotheses becomes astronomical. Biases provide the means of focusing the inductive search to look first to hypotheses expected to be characteristic of the particular domain or problem at hand.

Induction over the instance space, while initially viewed as a search in hypothesis space, can also be described as a search through bias space. Through the regulated application of proper biases, the inductive learning task becomes mechanical, almost easy: once proper biases are selected, the fruitful areas of hypothesis space are identified, and the search for an adequate concept description in hypothesis space is thus constrained. It is the determination of proper biases which becomes the difficult, yet crucial aspect of inductive learning.

2.2.2 Manifestations of bias

Researchers always build biases into the inductive systems they implement as a matter of practicality. One common bias is the use of a restricted description language such as allowing only Boolean attributes (features) in the instance space. Often the hypothesis/concept description language is limited to conjuncts of the instance space attributes, with disjunctions and negations prohibited. Systems using the version space method, such as the heuristic generalization component of LEX [Mitchell *et al.*, 1983], operate with this restriction. Some other systems allow disjunctions, but only within the clauses of the higher-level conjunctive description: these are known as internal disjunctive forms. The a priori imposition of the syntactic form of the hypotheses limits the kinds of problems or domains such systems can address. When the demands of the concept description exceed the limitations of the system, the system is left with no mechanism for modifying the bias and must categorize the problem as insolvable. The use of built-in biases which are inflexible and unresponsive to the demands of the domain at hand can be classified as *context-insensitive*.

Regardless of the method used to restrict hypothesis space, another type of bias is needed to determine the order in which the circumscribed hypotheses are examined. *Simplicity* is one such bias, preferring to entertain simple hypotheses over more complex ones. Simplicity has wide domain applicability and is a pervasive real-world heuristic. The reason for its success as a heuristic is straightforward: there are relatively few simple hypotheses; therefore, a simple hypothesis is unlikely to be consistent with the data by chance [Dietterich, 1990]. Simplicity could be classified as a context-insensitive bias since there is no information in the domain or problem to alter its behavior, yet it does not interfere with the ultimate discovery of the correct concept description as context-insensitive biases have the potential to do. Simplicity establishes a preference only within

the confines of the restrictions imposed by other biases. Hence, simplicity and other related preference biases can be classified as common sense or *context-independent* biases.

Context-independent and context-insensitive biases are general purpose problem solving approaches and are useful to consider when stronger, context-sensitive biases are unavailable. However, any available knowledge about the domain or particular problem at hand should be used whenever possible, since the biases derived from this knowledge have a strong likelihood of guiding the inductive search appropriately. These *context-sensitive* biases provide a means of exploring specific areas of hypothesis space known or suspected to be relevant to domains and problems similar to the current one. Within these areas of hypothesis space, context-sensitive biases also provide a means of intelligently selecting the most promising hypotheses first.

One context-sensitive bias useful for establishing hypothesis preference is a data-driven approach I have termed the *Quick-Look*: the evaluation of candidate hypotheses on a small subset of the training data (Section 4.2.1). The purpose of the Quick-Look is to identify the hypotheses that appear to be promising and to suppress consideration of those appearing useless. Although the Quick-Look is not extra-evidential, it qualifies as a bias since it regulates the search in hypothesis space. When used in conjunction with simplicity or other preference biases, it forms the basis of the hypothesis ordering mechanism needed to complement the aggressive, domain-knowledge-driven exploration of hypothesis space.

2.2.3 Knowledge as a powerful form of bias

Context-sensitive biases can be sensitive to the current data or the current domain. The Quick-Look is an example of a bias sensitive to the current data. Another bias sensitive to the data is the use of "outlier" instances to serve as the basis for suggesting new hypotheses. Biases sensitive to the current domain can use domain knowledge

in both hypothesis generation and hypothesis ordering. In hypothesis generation, domain knowledge may be used to explicitly suggest hypotheses or to constrain the generation of hypotheses through the restriction of the concept description language. Domain knowledge can also be used to intelligently relax previous language restrictions or to extend the language through the explicit addition of useful operators (such as extending the set of candidate mathematical operators) or constructors [Matheus, 1989, Seshu *et al.*, 1989]. Domain knowledge can also be used in the hypothesis ordering mechanism: knowledge of the past performance of the hypothesis (or class of hypotheses the current one belongs to) can be used to predict the utility of the hypothesis. Context-sensitive biases provide the only means of intelligently exploring diverse areas of hypothesis space without excessive sampling, and at the same time provide the means to tame the potential explosion of hypotheses.

One example of the use of domain knowledge to extend the space of possible hypotheses can be found in the system STABB (Shift To A Better Bias) [Utgoff, 1986]. In STABB, Utgoff uses a knowledge-based means of relaxing the initial language restriction bias through the use of information contained in the grammar of the description language and a set of backward problem-solving operators. This analytical approach produces new attributes that are refinements of the existing representation language as well as having been tested against a portion of the training instances. As such, they have a greater likelihood of being useful attributes than those suggested by context-insensitive or context-independent biases.

Work is also being done to use domain knowledge to explicitly guide the selection of generalization heuristics. In PREDICTOR [Gordon and Perlis, 1989], domain knowledge is used to determine the applicability of three heuristics for screening out potentially inappropriate hypotheses: cohesion, irrelevance, and independence. The *cohesion* heuristic indicates when it is appropriate to climb the generalization tree of a structured attribute.

The *irrelevance* heuristic indicates when attributes appear useless as discriminators and should be dropped. The *independence* heuristic is used in converting a disjunctive normal form to a conjunctive normal form with internal disjunction, thereby generalizing and compacting the description. One example of the application of these heuristics is the use of the irrelevance heuristic in the recognition of color as a useless attribute when describing something graspable by a robotic manipulator. When knowledge indicates that only structural or textural properties such as size, shape, hardness and surface slickness are applicable, any hypothesis containing the attribute *color* need not be generated.

In addition to using domain knowledge for generating hypotheses, it may also be advantageous to use domain knowledge to retract previously generated hypotheses. For example, in the robotic manipulator scenario, the system may have already proposed hypotheses using color as a component. At some point, when the system recognizes that the concept involves the graspability of an object, the proper domain knowledge can be invoked indicating the uselessness of the color as an attribute. The system can then discard or modify all hypotheses relating to color. An alternative approach would be to have the hypothesis ordering mechanism severely penalize those "retractable" hypotheses. This way, the hypotheses may not be completely eliminated from future consideration in case the domain knowledge turns out to be inapplicable.

Four primary approaches to using domain knowledge are considered in this thesis. They are discussed in greater detail in Section 3.1.1. In the first approach, domain knowledge is used to explicitly propose potentially useful hypotheses. This use of domain knowledge is illustrated in Sections 2.3 and 3.2, and is applied in some of the experiments of Chapters 5 and 6. The second approach involves using domain knowledge to propose operators and create new attributes; these operators and attributes provide avenues for the generation of new families of hypotheses. One example is the proposition of the XOR or parity operators in a Boolean domain to overcome the *parity problem* [Seshu, 1989].

Other examples are the proposition of the multiplication operator in mathematical or physical systems domains, and the use of symmetry in spatial domains such as board games. The proposition of Boolean operators AND, OR, and NOT is applied in some of the experiments of Chapters 5 and 6.

The third approach to applying knowledge for hypothesis generation uses domain knowledge to screen out or retract hypotheses, such as described in the robotic manipulator scenario. Knowledge is used in Section 6.1 to screen hypotheses generated through experiential learning. Finally, domain knowledge might be used to retract groups of hypotheses by the elimination of previously proposed operators. An example of this process is the rejection of the multiplication operator, and all hypotheses using it, when the system discovers that only nominal-valued attributes are applicable to the domain under consideration. This last approach is not used in any experiment of this thesis.

2.3 Opportunistic Constructive Induction in Action

Discussing the need for continued research in the area of constructive induction, Kerber [1988] wrote:

> A key limitation of current systems that perform constructive induction is that many of the descriptors "constructed" are either predefined, constructed prior to examining a single training example, or constructed regardless of which training examples are encountered. More valuable, are learning systems capable of extending the description language during run-time by creating new descriptors whose necessity was not pre-anticipated but whose creation is triggered by the needs of the current situation.

The *opportunistic constructive induction* process embodies this latter type of system: the hypothesis generator proposes hypotheses relevant to the state of induction. This

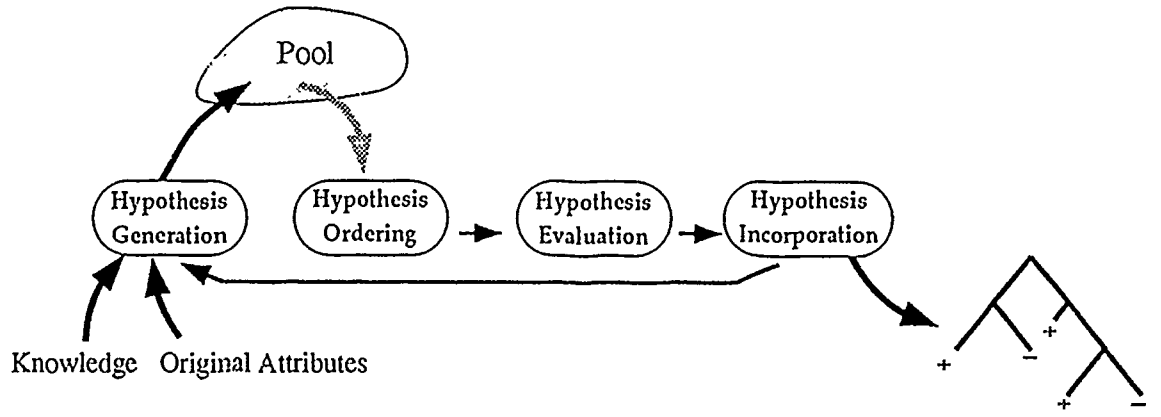
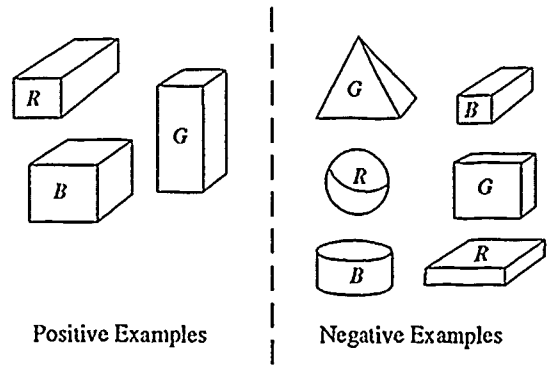


Figure 2.3 Opportunistic Constructive Induction of Decision-Trees.

section illustrates the interactions of the components of the opportunistic constructive induction process with a simple example. Figure 2.3 provides a pictorial representation of the pattern of operation. Knowledge and the original attributes are used by the hypothesis generator to produce a pool of potentially useful hypotheses. The hypothesis ordering component extracts hypotheses from the pool and provides the most promising ones to the hypothesis evaluation mechanism. The best hypothesis is selected and passed to the hypothesis incorporation mechanism to begin assembly of the decision-tree; i.e., the hypothesis is inserted as a decision point in the tree and the set of training data is divided among the branches. The incorporation of this hypothesis represents a change in the state of the system: it knows more about the concept. This state change can be used by the hypothesis generator to trigger more knowledge and produce additional hypotheses for the pool. The operation proceeds cyclically until a satisfactory tree has been assembled.

Figure 2.4 provides a simple example of a concept requiring constructive induction. A selective induction system could determine that the concept had something to do with the geometry of the objects (i.e., separate the objects into blocks and non-blocks),



Attributes: color (R, G, B), height, width, depth,
geometry (block, disk, sphere, pyramid)

Figure 2.4 A Difficult Concept: Large Blocks.

but would not be able to resolve the remainder of the concept given the listed attributes: *volume* is simply not in its vocabulary. For instance, in one experiment using the simplified values of *small, medium*, and *large* as the possible values for each of the attributes *height, width*, and *depth*, the selective induction system ID3 (Section 2.4.1) determined the concept had to be at least a block. It then overspecialized the description, requiring the blocks to be either tall (*height* = *large*), medium height and blue ((*height* = *medium*) AND (*color* = *blue*)), or short of height and of medium width ((*height* = *short*) AND (*width* = *medium*)). This description does not capture the essence of the intended concept, and is difficult to understand and convey.

Figure 2.5 illustrates the operation of the opportunistic constructive induction on the same concept. Initially, in part (a), nothing is known about the concept so the hypothesis generator proposes a set of *primitive* hypotheses based on the original attributes. At this point, the only knowledge applied would be the method used to form the intervals to make hypotheses from the real-valued attributes.[4] The hypothesis ordering mechanism culls

[4]The mechanism for converting real- or integer-valued attributes into nominal attributes by producing discrete ranges of values is not addressed in this thesis. Statistical partitioning approaches such as those

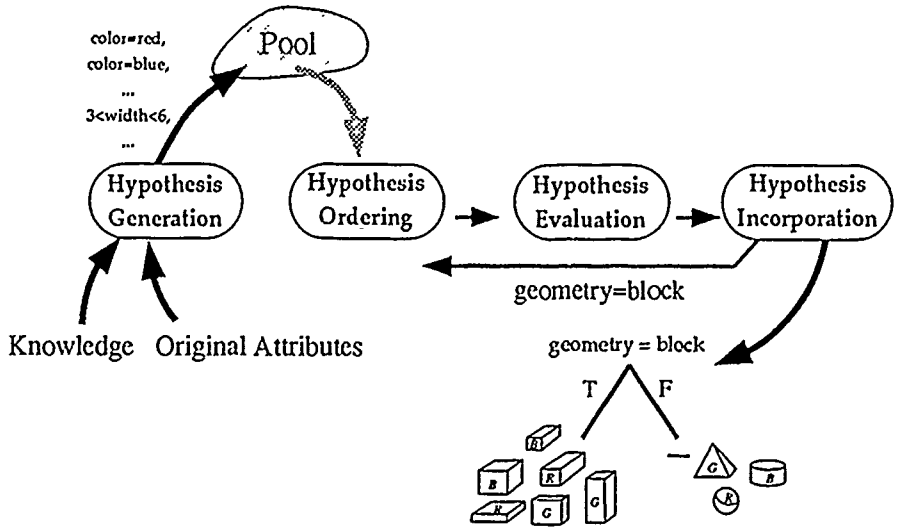


Figure 2.5 Using Knowledge Opportunistically. In part (a), the knowledge about the type of problem is used in conjunction with the original attributes to propose a set of primitive hypotheses to begin induction. One of the hypotheses, (*geometry* = *block*), is selected to begin the decision-tree. In part (b), this initial selection triggers the proposition of more hypotheses, based on domain knowledge about blocks (e.g., relationships among dimensions, construction of *volume*, etc.). The opportunistic constructive induction system can then discover the proper description: ((*geometry* = *block*) AND (*volume* > 45)).

from the pool of hypotheses and passes the most promising to the hypothesis evaluation component, based on a heuristic estimate of their potential utility. Full evaluation determines the hypothesis (*geometry* = *block*) to be the best choice to begin the decision-tree. The hypothesis incorporation component establishes the hypothesis in the decision-tree and partitions the training data accordingly.

In part (b) the state change of the system (i.e., from knowing nothing about the concept to believing the geometry is a block) is made available to the hypothesis generator. This triggers the application of domain knowledge which proposes *volume* as a potentially useful attribute when dealing with geometric objects. Once the new set of hypotheses has been proposed, hypothesis ordering is again used to select the most promising and pass them along for evaluation. In this example, (*volume* > 45) is found to be the best available hypothesis and is incorporated into the tree. At this point the decision-tree perfectly discriminates the training examples, but without overspecializing the tree as was the case with selective induction.

2.4 Relationship to Other Work

This section provides descriptions of several constructive induction systems in the context of the opportunistic constructive induction model presented in Figure 2.3. The number of extant constructive induction systems is large; thus, this section is intended to provide an overview of a broad range of approaches and capabilities, not to be exhaustive. Before the presentation of the constructive induction systems, the well-established selective induction system ID3 is discussed as a frame of reference.

used by STAGGER [Schlimmer, 1987] or the PLS series of programs [Rendell, 1985] could be incorporated into the hypothesis generation component to provide a preliminary set of ranges.

2.4.1 ID3

Figure 2.6 depicts the fundamental operation of ID3 for single concept learning [Quinlan, 1986]. The system ID3 typifies the selective induction process for decision-tree creation. The attributes in the original description language are tested against the training data to determine which attribute is the most informative for separating the positive examples from the negative. The chosen attribute is established as the decision-point in the tree, and for each branch created by a particular value of the attribute, the relevant training examples are collected. For each branch containing a mixture of positive and negative examples, the splitting process is repeated using the remaining attributes. Induction is complete when each leaf is pure (contains either positive or negative examples, but not both), or the set of original attributes is exhausted.

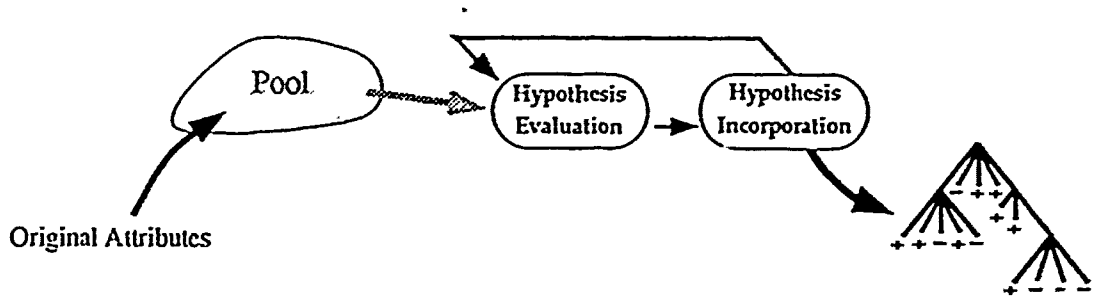


Figure 2.6 Selective Induction in ID3.

Hypothesis generation: The set of original (ground) attributes forms the pool. Hypotheses in the sense defined in Appendix B (i.e., testable statements that are either "true" or "false") are not used, except when the attributes happen to be Boolean. Each attribute is considered across its complete space of possible values, rather than in attribute-value pairs as implemented in OXGate. Real and integer-valued attributes must be partitioned into a finite number of discrete ranges by an external agent such as the user before induction can begin.

Hypothesis ordering: No hypothesis ordering is used to reduce the cost of evaluation. An ordering on attributes is imposed by the domain definition that can affect the outcome of decision-tree induction since it specifies the order in which the attributes are evaluated. In the basic implementation, if two attributes prove equally good, the first one tested is selected.

Hypothesis evaluation: An estimate of the information-gain (Section 4.2.1.2) is used to select the best attribute.

Hypothesis incorporation: The decision-tree is constructed by incorporating the attribute as the decision-point, and creating one branch for each of the values. The training examples are allocated to the branches according to the values they represent of the attribute. Branches containing purely positive or purely negative examples are labeled as such. The end of the branch is declared to be a leaf node. Branches related to values not represented in the training data are terminated with leaf nodes labeled "unknown" in the basic implementation; probabilistic approaches are used in more powerful versions. The remaining branches undergo the evaluation-incorporation process on their respective subsets of training examples until all branches terminate in leaf nodes, or the set of attributes is exhausted.

Extensions: The basic implementation of ID3 suffers several limitations, many of which are addressed in more sophisticated versions and derivative systems [Quinlan, 1985, Quinlan, 1986, Cheng *et al.*, 1988, Wirth and Catlett, 1988, Utgoff, 1988, Norton, 1989]. One limitation is that the simple system tends to overfit the decision trees to the training data. A statistical (approximately chi-squared) *pre-pruning* approach is used in later versions to make the decision-trees more general and less susceptible to noise [Quinlan, 1986]. Another limitation is that the basic version unduly prefers attributes with

large numbers of values over those with few values. To offset this preference, a modified evaluation measure is used, i.e., the *gain-ratio*. The gain-ratio employs an estimate of the intrinsic value of the attribute in addition to the information-gain [Quinlan, 1985, Quinlan, 1986].

The system ID3 has been used for several commercial applications with very large amounts of training data. To reduce the memory storage requirements, the technique of *windowing* was developed. A subset (window) of the training data is used to develop the decision-tree, then the tree is tested against the remaining data. If the decision-tree is inadequate to correctly classify the examples, a subset of the exceptions are added to the window, and the decision-tree is relearned. This process is repeated until the decision-tree satisfactorily represents the full set of training data. The use of windowing reduces memory requirements at the cost of greater processing time, with no apparent loss of accuracy [Wirth and Catlett, 1988].

2.4.2 FRINGE

Figure 2.7 depicts the operation of the constructive induction system FRINGE, an extension of the selective induction system ID3 [Pagallo and Haussler, 1989, Pagallo,

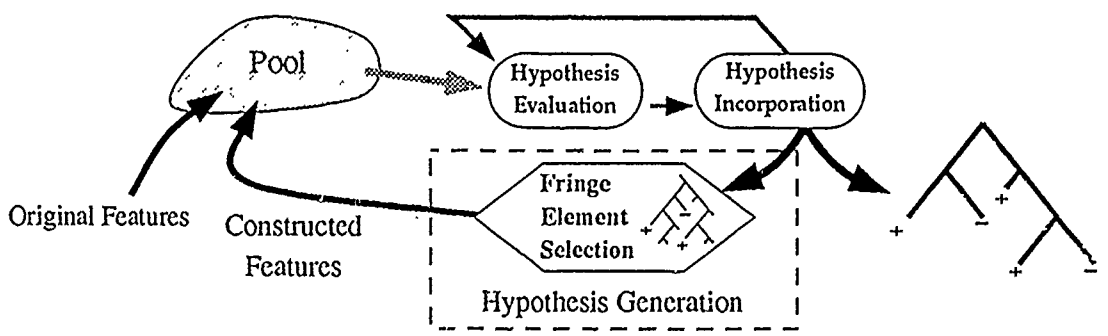


Figure 2.7 Constructive Induction in FRINGE.

1989]. FRINGE typifies a class of *experiential learning systems* that use the results of one learning session to create new descriptive terms for subsequent sessions. The motivation behind the development of FRINGE was to solve the *replication problem* in decision-tree induction: the occurrence of duplicated subtrees in the concept description. FRINGE constructs new terms using the decision points at the fringes of the positive branches of the decision-tree (described in detail in Chapter 6). These new terms are added to the description language and the decision-tree is relearned. Eventually, the replicated subtrees are represented as single complex terms and become incorporated in a simpler decision-tree as depicted in Figure 2.8. This approach to feature construction (hypothesis generation) is effective in improving the accuracy on random Boolean functions, even in the presence of noisy data [Pagallo, 1989].

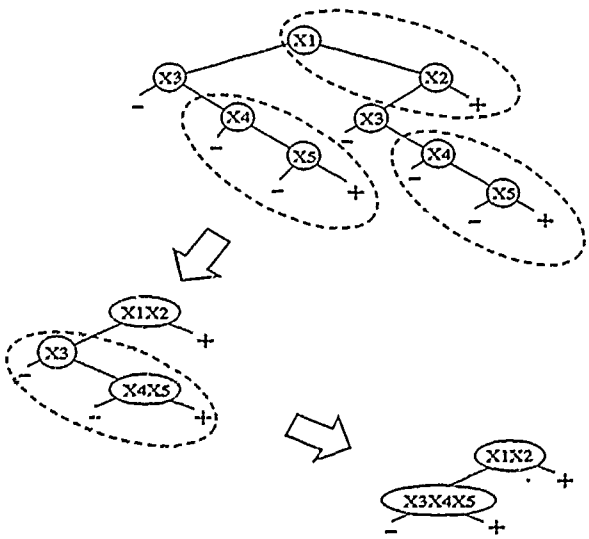


Figure 2.8 Solving the Replication Problem.

Hypothesis generation: FRINGE operates in the domain of Boolean functions; therefore, all descriptors are Boolean attributes, or features. The initial pool of hypotheses consists of the original features. As discussed in Appendix B, a feature is a simple form

of hypothesis, and the features found in the original description language are equivalent to primitive hypotheses. After a decision-tree is created, the features used as decision points nearest the positive leaves are conjoined to create new features. These constructed features are added to the pool for the next iteration of the process.

Hypothesis ordering: No hypothesis ordering is used to reduce the cost of evaluation. Hypothesis generation is terminated, however, if the number of new hypotheses exceeds a user-defined threshold. As in ID3, the order in which the features are evaluated may affect the outcome of decision-tree induction. This effect is ameliorated by the introduction of the new, and presumably better, features for subsequent iterations.

Hypothesis evaluation: As in ID3, an estimate of the information-gain is used to select the best feature.

Hypothesis incorporation: The decision-tree is assembled in the typical fashion. Induction is declared complete when no new features are constructed, or when the number of new features reaches a predefined threshold.

2.4.3 CITRE

Figure 2.9 depicts the operation of the constructive induction system CITRE, an extension of the experiential learning approach found in FRINGE [Matheus, 1989]. CITRE provides a variety of biases for feature selection, can filter out undesirable features with domain knowledge, has the potential to generalize constructed features, and can control the number of features made available for decision-tree assembly.

Hypothesis generation: Like FRINGE, CITRE uses Boolean attributes (features) only for decision-tree induction: nominal-, real-, and integer-valued attributes must be

converted to Boolean before induction can begin (Appendix B). The set of original features forms the initial pool of primitive hypotheses. After a decision-tree is created, pairs of features appearing along paths to positive leaves are used as operands for the construction of new features, limited to certain locations along the paths by a user-selectable bias (Chapter 6). Domain knowledge can also be applied to filter out feature pairs deemed inappropriate by the knowledge for new feature construction. New features are constructed from the remaining feature pairs. If domain-dependent generalization knowledge is available, the new features are generalized before being added to the pool.

Hypothesis ordering: CITRE is able to use one of two approaches to limit the number of constructed features retained for decision-tree induction. One method evaluates each new feature against the entire set of training data and keeps the N features with the highest values of information-gain. The value of N is a fixed threshold equal to the number of internal nodes present in the original decision-tree. The other method ages the hypotheses, discarding all hypotheses created in previous iterations but not appearing in the latest decision-tree. In both cases, the primitive hypotheses always remain in the pool.

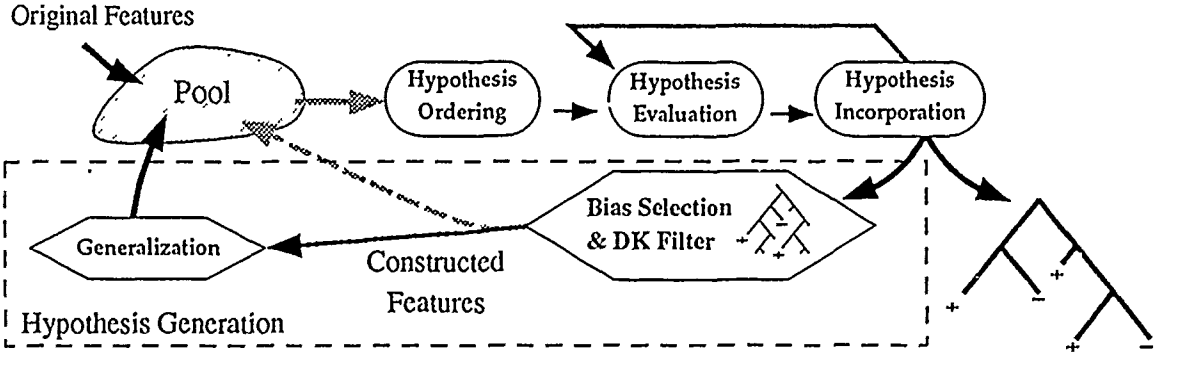


Figure 2.9 Constructive Induction in CITRE.

Hypothesis evaluation: An estimate of the information-gain is used to select the best feature to incorporate in the decision-tree.

Hypothesis incorporation: The decision-tree is assembled in the typical fashion. Induction is declared complete when no new features are constructed.

2.4.4 MIRO

The constructive induction system MIRO uniquely integrates the deductive use of knowledge with induction over an abstraction of the training examples [Drastal and Raatz, 1989, Drastal *et al.*, 1989]. Processing occurs in several stages: a preprocessing stage to create an *abstraction space*, an induction stage to formulate the concept description, and a postprocessing stage to convert the discriminant representation into a characteristic one. The resultant concept description is in *disjunctive normal* form, a representation readily convertible to a decision-tree.

In the first stage, the deductive component of MIRO uses the domain knowledge to create an abstraction space from which to derive the concept description language. Each training example is applied to the domain knowledge to instantiate/prove aspects of the knowledge set and create proof structures. The most abstract (general) descriptors in the proof structures become the set of hypotheses composing the initial concept description language.

The second stage uses the abstract description language and the training examples to induce a discriminant concept representation, separating the positive from the negative examples. If all of the training examples cannot be adequately represented with the set of most abstract hypotheses, less abstract hypotheses are selected from the proof trees and added to the description language. This process repeats until all of the positive examples

are distinguished from the negative examples, or the entire set of abstract and primitive hypotheses is exhausted.

The postprocessing stage specializes the discriminant concept description to create a characteristic description. For each term of the discriminant disjunctive normal form description, hypotheses that are common to all of the positive examples covered by the term are conjoined to the term. The additional hypotheses are selected from the unused hypotheses in the combined set of abstract and primitive hypotheses. The augmented terms are then disjoined together to create the complete characteristic description of the concept. Figure 2.10 depicts the operation of MIRO in the context of the opportunistic constructive induction model.

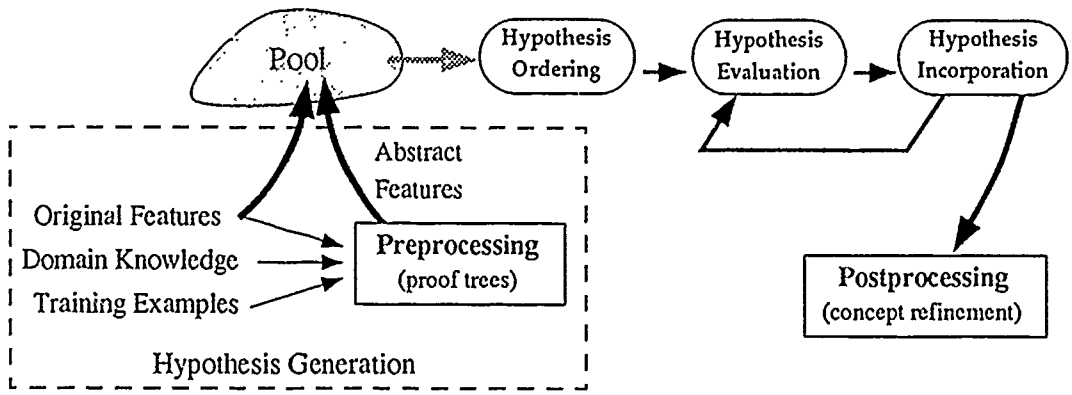


Figure 2.10 Constructive Induction in MIRO.

Hypothesis generation: The hypothesis pool is filled during the preprocessing stage, and contains the full set of abstract hypotheses appearing in the proof trees plus the primitive hypotheses appearing in the descriptions of the examples.

Hypothesis ordering: Only the most abstract hypotheses are selected from the pool for induction. If the hypothesis evaluation component is not able to distinguish the

positive from negative examples, less abstract hypotheses are heuristically selected from the pool and made available for induction.

Hypothesis evaluation: Concept induction in MIRO is performed using the one-sided variant of the candidate elimination algorithm [Mitchell, 1978, Haussler, 1987]. First, a positive example is heuristically chosen as the *seed*. Next, the most general descriptions of that seed that exclude all the negative examples are developed from the set of hypotheses provided by the hypothesis ordering mechanism. These descriptions are conjunctions of hypotheses. When multiple descriptions are produced, a heuristic measure is used to select the one with the best balance between two criteria: the number of (previously uncovered) positive examples covered by the description, and an extra-evidential measure of the amount of knowledge collectively entailed in the proof structures of the covered examples.

Hypothesis incorporation: Once the best description has been selected for the seed, all positive examples covered by that description are removed from the training set. The seed description is disjoined to the developing concept description, and the hypotheses used in the seed description are returned to the pool for possible use with other seeds. The evaluation-incorporation cycle continues until all of the positive examples have been removed, or all of the hypotheses in the pool have been considered and determined to be inadequate.

2.4.5 STAGGER

STAGGER is another novel approach to constructive induction [Schlimmer, 1987, Schlimmer and Granger, 1986]. Concepts are represented through a collection of hypotheses as depicted in Figure 2.11. Each hypothesis contributes to the overall concept

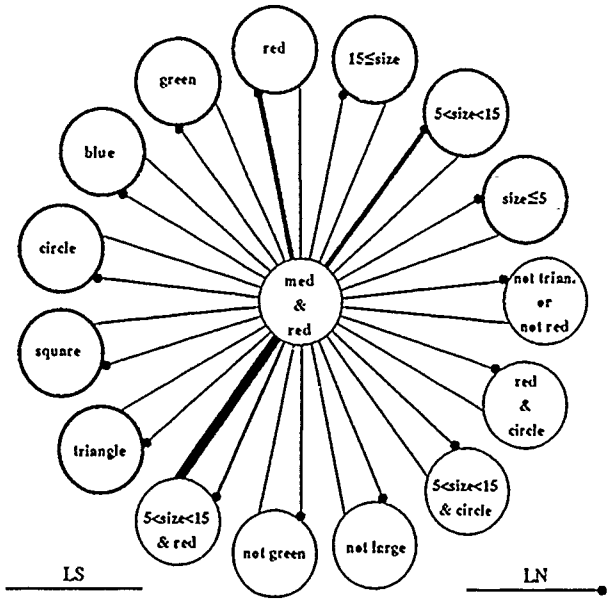


Figure 2.11 Concept Description Support in STAGGER. Pictorial representation of the support for the concept (*medium* AND *red*). The lines terminated with small dots represent the logical sufficiency (LS), the others represent logical necessity (LN). The thickness of a line is indicative of the degree of logical necessity or sufficiency the associated hypothesis provides for the target concept.

description in accordance with the strengths of two weights associated with the hypothesis. The logical sufficiency (LS) approximates the degree with which the presence of the hypothesis increases the expectation of an outcome. The logical necessity estimates the degree with which the absence of the hypothesis decreases the expectation of the outcome. In the example, the unknown target concept (*medium* AND *red*) is best supported by the constructed hypothesis ($(5 < size < 15)$ AND *red*), while the primitive (original) hypotheses *red* and $(5 < size < 15)$ are the most necessary.

STAGGER is an incremental learning system, meaning examples are introduced one at a time and the concept representation is adjusted with each example. Learning occurs in three ways. In the first, the LS and LN weights are adjusted for each hypothesis by

$$\text{LS} = \frac{p(matched|example)}{p(matched|\neg example)} \qquad \text{LN} = \frac{p(\neg matched|example)}{p(\neg matched|\neg example)}$$

With the weighted learning component, STAGGER behaves as a single layer connectionist network. It is capable of discovering only linearly separable concepts in terms of the existing descriptive elements. The second method of learning uses a Boolean learning component that proposes more complex hypotheses, thereby allowing the STAGGER to behave as a multilayered network. Conceptually, STAGGER begins with a strong bias of using only primitive hypotheses and then relaxes the bias as required to expand the representation language and construct more complex hypotheses.

The third learning method uses a numerical learning component to handle real- and integer-valued attributes by partitioning the attributes into discrete Boolean units. For example, the real-valued attribute *size* is recast in the example as three Boolean features: small ($size \leq 5$), medium ($5 < size < 15$), and large ($15 \leq size$).

Hypothesis generation: The Boolean learning component is failure-driven: it is activated when the concept description mispredicts the classification of a new example. Constructed hypotheses are proposed according to the following heuristics:

1. If a negative example is predicted to be positive (error of commission), then the system is behaving too generally: some necessary hypothesis was unmatched.
    - Conjoin two strongly necessary hypotheses,
    - Negate a hypothesis that is matched but is a poor predictor.
2. If a positive example is predicted to be negative (error of omission), then the system is behaving too specifically: some sufficient hypothesis was unmatched.
    - Disjoin two strongly sufficient hypotheses,
    - Negate a hypothesis that is unmatched and is a poor predictor.

The numerical learning component determines the end-points of the discrete intervals by using statistics collected from the positive and negative examples to estimate

the utility of proposed boundaries. The number of partitions is an externally supplied parameter. First, STAGGER applies a local smoothing function to remove noise. Then, the locally maximal end-points are found using the utility formula

$$U(end\text{-}points) = \prod_{i=1}^{|classes|} odds(class_i) \times \frac{p(class_i| < end\text{-}point)}{p(class_i| > end\text{-}point)}$$

The conditional probabilities are computed from the number of positive and negative examples of each class with values less and greater than the proposed end-point. The prior probability is also statistically derived. The best end-points are retained and used to divide the real-valued attribute into discrete units for use by the Boolean and weight learning components for concept construction. As more examples are obtained, the end-points are adjusted accordingly, and any resultant changes to the discrete representation are propagated through.

Hypothesis ordering and evaluation: In practice, STAGGER generates more hypotheses than it can use; therefore, a method of pruning is required. A set of heuristics is applied to determine the potential value of hypotheses as they are constructed and to discard the least promising ones. Simultaneously, the hypotheses already in the pool are also assessed to remove the least useful ones. A bookkeeping mechanism is incorporated to allow backtracking should a hypothesis prove to perform more poorly than in the past. By constantly removing hypotheses from the pool as better ones are added, the system can extend its search frontier while maintaining a limited number of hypotheses. This form of beam search allows STAGGER to maintain a manageable memory size and linear search time, even as the expressive power of the representation language increases.

Hypothesis incorporation: All of the hypotheses in the pool are considered part of the concept description. A new hypothesis is incorporated by establishing the values for

LS and LN as described above. In addition, each time a new example is introduced, the LN and LS weights are incrementally recalculated for each hypothesis.

2.4.6 ML-SMART/ENIGMA

Up to this point, the discussion has focused on similarity-based learning systems, also known as *empirical* learning systems, that use relatively large numbers of training examples to develop the concept descriptions. In the more sophisticated systems, domain knowledge is added to help guide the inductive process. At the other end of the spectrum are *explanation-based*, or *analytic*, learning systems which rely primarily on complete sets of domain knowledge (domain theory) and perform induction using a small number of training examples, one at a time. The more sophisticated of these systems are able to use several training examples simultaneously to overcome and correct deficiencies in the knowledge base, such as incompleteness or inconsistency. The system ENIGMA, an extension of ML-SMART, is one example of a hybrid empirical/analytic learning system that interleaves deduction over the domain knowledge with induction using all of the training examples simultaneously [Bergadano and Giordana, 1988, Bergadano *et al.*, 1988, Bergadano *et al.*, 1990].

In analytical learning, the system is given the high-level description of the concept (often its name) and the set of training examples. The goal of the system is to *operationalize* the description: reformulate it in terms of the testable attributes of the examples by splicing together the appropriate pieces of knowledge from the domain theory. For instance, the concept shown in Figure 2.4 could be initially described as *monolith*. Operationalizing this concept first involves transforming the description to ((*geometry* = *block*) AND (*volume* > 45)). The first condition is operational (directly testable), but the second is not: *volume* must be reformulated before the concept is completely operationalized. This process "proves" the training examples. If a contradiction

is reached, the example is either assumed to be noisy, or the domain theory is assumed to be flawed.

The inadequacies of the domain theory are handled statistically in ENIGMA by utilizing the full set of training examples simultaneously to identify the deficient pieces of knowledge and avoid them in the operationalization process [Bergadano and Giordana, 1988]. Statistics over the examples also contribute to a heuristic evaluation criterion used to select the best partial explanations from a set of alternatives and guide further operationalization. ENIGMA also handles the acquisition of concepts in noisy domains by attaching a measure of uncertainty to each piece of knowledge and updating the measure through statistical analysis of the examples [Bergadano *et al.*, 1988].

If the training examples are not noisy and the domain theory is correct, operationalization will proceed smoothly. ENIGMA will develop the concept description in a completely top-down fashion, behaving as a pure explanation-based learning system, with the exception that multiple examples will be used to provide a better evaluation criterion for selection among competing solutions. If operationalization fails completely, ENIGMA defaults to the other end of the spectrum and performs purely inductive, similarity-based learning.

In a sense, ENIGMA performs a type of constructive induction: a new term (the high-level concept) is added to the description language and is defined by the operational (primitive) attributes. In addition, intermediate concepts are also developed as the operationalization of the given concept unfolds. In the case of noiseless data and a perfect domain theory, the operation of ENIGMA is strictly deductive; however, this is considered a special case by its developers. The principal mode of operation combines induction over the examples with the deductive use of knowledge.

Hypothesis generation: At each step of the operationalization process, one or more hypotheses are proposed. Each hypothesis equates the non-operational intermediate (or high-level) concept with a specialization of that concept: a more refined definition described with lower-level terms. Ultimately, the lowest-level terms must be operational, i.e., primitive hypotheses.

Hypothesis ordering and evaluation: Simple examination of the training examples determines if the operational components of a hypothesis are correct. If the operational components are not completely correct, or if multiple hypotheses are being considered, a heuristic evaluation criterion is applied to select the best hypotheses, if any, for further consideration. Multiple chains of reasoning may be maintained: the heuristic criterion is used to limit the scope of the search to the most promising hypothesis. The criterion may be based on three kinds of heuristics: statistical (a hypothesis is "good" if supported by enough examples), domain-independent (e.g., hypothesis simplicity or understandability), and domain-specific (a priori knowledge for guiding the search). ENIGMA uses the statistical and domain-independent heuristics.

Hypothesis incorporation: Multiple chains of reasoning are maintained as long as a chain appears worthwhile given the examples and heuristic criteria. Accepted hypotheses are incorporated into the chains to create a *tree of formulas* or *specialization tree* [Bergadano and Giordana, 1988], and will form the basis for the next attempt at hypothesis generation. Terminal nodes of this tree represent either dead ends or positive leaves. A dead end occurs when positive and negative examples cannot be distinguished with the operational terms present at that node. The positive leaves are partial concept descriptions that correctly cover some of the positive examples and do not cover any negative example. When all of the positive examples have been correctly covered, the concept description is the disjunction of the positive leaves.

2.4.7 Summary: Relationship to other work

This section provided a description of several notable constructive induction systems in terms of the opportunistic constructive induction model. Many of the ingredients of these systems can be used to create a generalized approach to constructive induction, described in Chapter 3. For instance, the experiential learning mechanisms of FRINGE and CITRE, the domain knowledge filtering approach in CITRE, the numerical learning algorithm of STAGGER, and the deductive applications of domain knowledge in MIRO and ENIGMA could all be developed as coexisting components of a flexible hypothesis generation mechanism. Hypothesis ordering embodies the notion of limiting the number of hypotheses to be considered (CITRE), but could do so using a competitive method based on heuristic evaluation criteria such as evidential support and simplicity (ENIGMA). Hypothesis evaluation could use any number of evaluation criteria to determine the "goodness" of the desired representation. The choice of representation also affects the method of hypothesis incorporation.

This section was not intended to be exhaustive: the number of extant constructive induction systems and approaches to representation reformulation is large and increasing rapidly. Matheus [1989] lists thirty-two systems and approaches, ranging from the well-established to the most recent advances of 1989. Since that time, other systems and approaches have been introduced, including ENIGMA [Bergadano *et al.*, 1990], CAP [Hume, 1990], the "boot-strap" learning of Flann [1990], the ILS framework [Silver *et al.*, 1990], IR1 [Wu *et al.*, 1990], the E* function-finding algorithm of Schaffer [1990], the bounded generalization algorithm of Smith and Rosenbloom [1990], and pattern-based approaches to constructive induction [Rendell and Seshu, 1990, Yang *et al.*, 1991]. The approaches presented in this section span a broad spectrum of capabilities. They provide a basis for the discussion of the opportunistic constructive induction framework, and are solid starting points for the development of the prototype implementation OXGate.

CHAPTER 3

THE OXGATE FRAMEWORK

This chapter describes the architecture of OXGate, the prototype opportunistic constructive induction system. OXGate is intended to provide a domain-independent development and testbed environment for the application of domain knowledge to guiding decision-tree construction. The major components of OXGate are described in this chapter, along with justifications for the relative depths of implementation and investigation each component received during the current phase of research.

3.1 Major OXGate Components

The inductive process presented in Section 2.1 consists of four activities: hypothesis generation, hypothesis ordering, hypothesis evaluation and hypothesis incorporation. These activities map directly into the four major components of the OXGate architecture as shown in Figure 3.1. The components interrelate through the sharing of information posted on a central blackboard. The blackboard itself is not necessary for opportunistic constructive induction, yet it provides a convenient mechanism for the modular development, expansion and testing of OXGate.

The current phase of research described in this thesis is a portion of a larger envisioned research effort. The thesis is a snapshot of an ongoing program, and OXGate is currently a partially implemented prototype of the opportunistic constructive induction system. The current implementation consists of the overall system architecture, baseline hypothesis ordering and hypothesis evaluation modules, and partially developed hypothesis incorporation and central blackboard mechanisms. These components provide a foundation

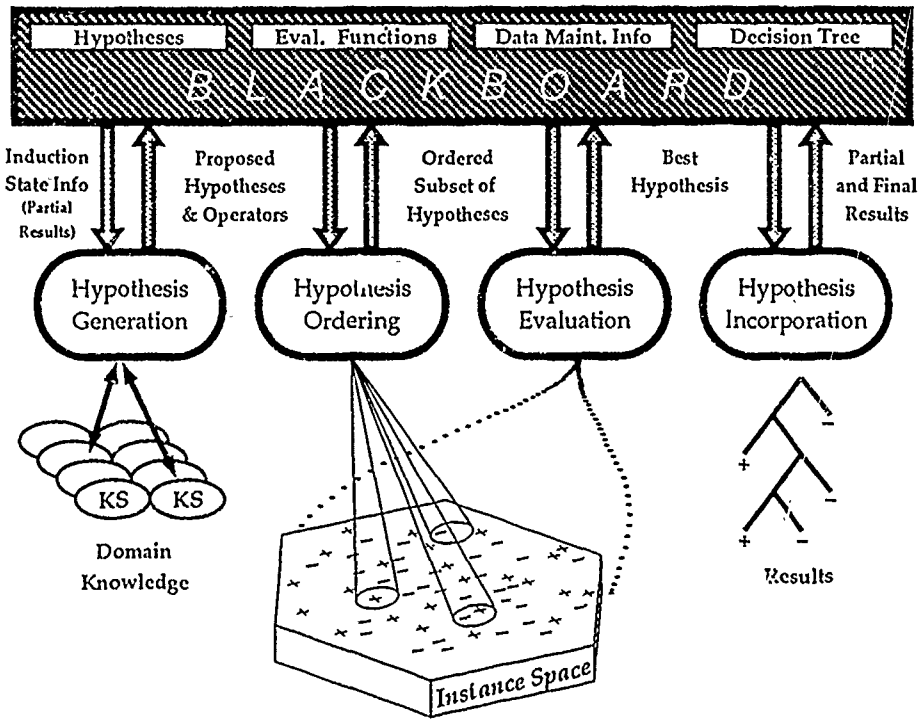


Figure 3.1 The OXGate System Architecture. The four components of the inductive process communicate with each other through the blackboard. The blackboard maintains the set of potentially useful hypotheses, the available evaluation functions, and the bookkeeping information to index hypotheses and instances to the relevant nodes of the developing decision-tree.

for the exploration of hypothesis generation. The hypothesis generation component has been implemented only to the extent necessary to perform controlled experiments on the effectiveness of certain pieces of domain knowledge.

The four major components of OXGate are presented in the order in which they appear as elements of the inductive process. This ordering also represents the degrees of interestingness, difficulty and novelty to the machine learning community. The first component discussed, hypothesis generation, is the least understood, most difficult and most interesting of the four. The last component, hypothesis incorporation (decision-tree assembly), is the most understood, easiest, and least interesting. The ordering

also represents an implementation dependency: the more difficult components cannot be adequately implemented and tested without prior development of the lesser ones. While hypothesis generation is certainly the primary thrust of the larger research effort, it cannot be properly addressed without first developing a mechanism for managing the effects of prolific hypothesis generation. Therefore, this thesis concentrates on hypothesis ordering with the intention of providing a foundation for future exploration of hypothesis generation and fully opportunistic constructive induction.

3.1.1 Hypothesis generation

Of the four components of opportunistic constructive induction, the richest area for research is hypothesis generation: the proposition of potentially useful portions of the concept description. The present implementation of OXGate provides a testbed environment for investigating the application of knowledge for hypothesis generation, and some aspects are investigated in Chapter 6. For the most part, knowledge is currently incorporated into special-purpose procedures to generate large numbers of hypotheses for exercising the hypothesis ordering mechanism and the overall OXGate implementation. The following discussion lays the foundation for future development of the hypothesis generation component of OXGate.

The fully developed hypothesis generation component will most likely consist of a collection of procedures, knowledge sources, and special-purpose routines, managed by a central controller which provides the interface to the rest of OXGate through the blackboard. The controller would retrieve the state information from the blackboard, channel it to the relevant subcomponents, and format their outputs into usable hypotheses. Such an approach would provide a flexible hypothesis generation capability, allowing a variety of forms of knowledge to be incorporated into a common framework, and isolating the rest of OXGate from the specific representations of knowledge used.

3.1.1.1 Uses of knowledge

Section 2.2.3 described four approaches to utilizing domain knowledge for hypothesis generation: 1) explicitly proposing hypotheses, 2) proposing operators, 3) retracting or screening hypotheses, and 4) retracting operators. The first three are used to varying degrees in the experiments of Chapters 5 and 6. Three types of knowledge are used in the experiments: syntactic, contextual, and experiential.

Proposing Hypotheses: The first approach to using knowledge for hypothesis generation is for the explicit proposition of hypotheses. This approach was used in the example of Figure 2.5 where hypotheses involving the attribute *volume* were generated once the system discovered the concept involved geometric shapes. In the example, the system discovered the geometry was a block and this change of state was made available to the hypothesis generator. This triggered the application of domain knowledge which proposed *volume* as a potentially useful attribute when dealing with geometric objects. Although various schemes are possible, such knowledge might be represented in the form of productions, or *condition-action* statements, as depicted in the following samples.

$$(geometry = block) \longrightarrow \Big(volume = (height \times width \times depth)\Big)$$

$$(geometry = sphere) \longrightarrow \left(volume = \left(\frac{\pi \times width^3}{6}\right)\right)$$

$$(geometry = pyramid) \longrightarrow \left(volume = \left(\frac{height \times width \times depth}{3}\right)\right)$$

$$(geometry = disk) \longrightarrow \left(volume = \left(height \times \frac{\pi \times width^2}{4}\right)\right)$$

The knowledge shown above is a simplification of that required by the hypothesis generator to form hypotheses. It simply states when and how to create the attribute *volume*. The hypothesis generator must also recognize that *volume* is a real-valued attribute, and it must generate hypotheses based on this new attribute. A functional description of the

use of knowledge in this form is

$$(\text{match-state } \textit{condition}) \Longrightarrow \big(\text{propose}-\text{hypotheses } (\text{create}-\text{attribute } \textit{action})\big)$$

The experiments of Section 5.3 and the one shown in Figure 6.6 use the explicit generation approach for the NetTalk domain, in which a piece of contextual knowledge is used as the basis of the powerful domain-specific hypothesis ($C3 = C4$), i.e., the third character is equal to the fourth, such as a repeated consonant.[1] Although the hypothesis itself is created by hand for the experiments, the approach could easily be automated as described above. For this hypothesis, a declarative representation of the knowledge might look like

$$\big((\textit{domain} = \textit{NetTalk}) \text{ AND } (\textit{concept}-\textit{type} = \textit{pronunciation})\big) \longrightarrow (C3 = C4)$$

Hypotheses are also explicitly generated when the original set of primitive hypotheses is created, i.e., when the original description language is defined. For nominal and Boolean attributes, this process is straightforward. The primitive hypotheses are created from all possible values of the attributes. Appendix B describes this process more fully. Creating primitive hypotheses from integer- and real-valued attributes is more difficult, and requires a mechanism to form ranges of values, thereby converting the attributes to nominal-valued. Statistical partitioning approaches such as those used by STAGGER [Schlimmer, 1987], the PLS series of programs [Rendell, 1985], or the approach described in [Chan and Wong, 1990] provide a starting point for development of this capability in OXGate.

Another form of representing knowledge is frame-based or hierarchical. Manago [1989] uses a frame-based representation in KATE, an extension to ID3 that focuses the examination of attributes on those relevant to the current partial concept description. Kerber [1988] uses explicit hypothesis generation in his system OTIS (OpporTunistic Induction

[1] See Appendix E and Section 5.3 for descriptions of the domain, concept and hypotheses.

System) which maintains a generalization hierarchy of relations, attributes, and values describing the domain of artificial "cancer cells." The hierarchy guides the application of generalization and specialization operations and assists in handling internal disjunction and two other forms of new term construction (transitive closure and counting arguments). OTIS incorporates a scheduler that maintains an agenda of the most promising tasks for generating new terms.

Experiential knowledge (knowledge derived from previously learned concepts) is also useful for explicit hypothesis generation. Chapter 6 describes Koala, a special-purpose routine for examining the results of one decision-tree assembly session and constructing new hypotheses based on this experience for use in future learning. The knowledge is not explicitly represented in the hypothesis generator, but rather is contained in a decision-tree and extracted procedurally to form new hypotheses. These hypotheses embody knowledge of the past successes and are explicitly proposed in future learning tasks, making this approach a legitimate form of hypothesis generation.

Proposing Operators: The second approach to using knowledge for hypothesis generation proposes operators for constructing hypotheses. The operators can be applied to attributes to form new attributes, and from these new attributes hypotheses can be constructed. Operators can also be applied to existing hypotheses to construct more complicated ones. Generalization operators (see Matheus [1989]) may also be applied to the constructed hypotheses at this stage, to create hypotheses less specific to the training instances. The knowledge for proposing these operators could be syntactic, specifying the expected form of the concept description, or it may be contextual, specifying the conditions under which the operators might be applied. Examples of this approach include the proposition of parity operators in a Boolean domain to overcome the *parity problem* [Seshu, 1989], proposition of the multiplication operator in mathematical or physical

systems domains [Langley *et al.*, 1986], and the use of symmetry in spatial domains such as board games [Rendell and Seshu, 1990].

The Boolean operators AND, OR, and NOT are applied in some of the experiments of Chapters 5 and 6 to generate large numbers of constructed hypotheses. Here, the assumed knowledge is syntactic, specifying an expectation of the form of the concept or useful components of its description. This type of application has the potential to overwhelm a system with a vast number of irrelevant hypotheses. The operators must be applied carefully, indicating the need for semantic or contextual knowledge to constrain the generation and examine selected areas of the hypothesis space.

Of the four approaches, operator proposition for hypothesis generation is certainly the richest area for research. It is also the most unconstrained. The current implementation of OXGate provides a convenient development and testing environment for its exploration.

Screening/Retracting Hypotheses: The third approach to using knowledge in the hypothesis generator does not produce hypotheses, but rather constrains their generation by the first two approaches. Applying syntactic knowledge to propose operators can result in a proliferation of inapplicable hypotheses. Semantic knowledge can constrain the production by eliminating logically inconsistent hypotheses and tautologies, for example, $((color = red)$ AND $(color = blue))$ or the hypothesis $((X1 = T)$ OR $(X1 = F)))$. Contextual knowledge can be used to restrict the attributes and hypotheses provided to the operators, avoiding senseless constructions such as $((first\text{-}letter = Q)$ AND $(fifth\text{-}letter = X))$ or hypotheses involving the attribute *taste* when dealing with a robotic manipulator.

Knowledge can also be used to retract previously generated hypotheses when the unfolding concept description indicates that a particular attribute or class of hypotheses is no longer relevant. For instance, if OXGate discovers the concept to involve objects of

sizes larger than automobiles, then any hypothesis constructed with the attribute *taste* could probably be removed. Hypotheses involving small sizes could also be removed from the branches of the decision-tree where the size was established to be large. This latter type of filtering embodies procedural knowledge of how to determine the logical implications and refutations of incorporating a hypothesis into the decision-tree. Whether this function belongs in the hypothesis generator or the hypothesis incorporation component remains to be determined.

Contextual knowledge is used by Koala to restrict the conclusions drawn from experience. The mechanization of this process is described in detail in Section 6.1 and used in the experiments of that section. Essentially, the knowledge is used to specify which attributes are to be considered as operands for hypothesis construction. Of all the potential constructions discovered by Koala, only those allowed by the knowledge are created.

Retracting Operators: The fourth approach to using knowledge for hypothesis generation involves retracting previously proposed operators, causing the retraction of affected hypotheses. An example of this process is the rejection of the multiplication operator, and all hypotheses using it, when OXGate discovers that only nominal-valued attributes are applicable to the domain under consideration. This approach is not utilized in any of the experiments or examples of this thesis.

3.1.1.2 Implementation of the hypothesis generator

Under the current envisionment of OXGate, domain knowledge will be stored in a collection of modules known as knowledge sources (KS in Figure 3.1), as well as special-purpose procedures for generating certain types of hypotheses. State changes in the blackboard will be passed to the knowledge sources by the central controller, which

maintains the interface of the hypothesis generator to the blackboard. The knowledge sources invoked by these changes will propose hypotheses or operators for collection by the central controller. This data-driven approach to knowledge source invocation may be implemented in a control-driven manner similar to the Hearsay approach [Erman *et al.*, 1980]. Instead of polling all the knowledge sources sequentially, or maintaining a society of asynchronous self-polling daemons waiting for their triggers to occur, the controller will simply match the state information to a precompiled matrix of triggers to determine which knowledge sources to invoke.

The central controller is responsible for collecting and collating the hypotheses, rejecting those deemed unacceptable by other knowledge sources and prior information, and removing redundancies. It will pass the hypotheses to the blackboard for posting and retract those already posted if necessary. Whether proposed operators are best stored on the blackboard or internal to the hypothesis generator has yet to be determined. If the hypothesis generator will need to incorporate its own local blackboard for management of the knowledge sources, the operators should also be stored there.

Since the controller provides the interface to the rest of OXGate through the blackboard, the internal representations used by the knowledge sources can be in any form convenient to the developer as long as the hypothesis generation mechanism as a whole posts hypotheses to the blackboard in the proper format (described in Section B.2). A knowledge source must be procedural in some sense, but the knowledge used may be in any form useful to the source to produce a *testable* hypothesis given the current state of the inductive process, e.g., declarative, a set of productions, procedural, or frame-based. Production rules provide a simple mechanism for both explicit hypothesis generation as well as deductive reasoning within the knowledge source. Reasoning about the state changes and responding with carefully chosen hypotheses should improve the speed and quality of the concept learning as the proposed hypotheses are more likely to be directly

applicable. It would be a simple matter to tag the productions to distinguish them from other productions with different uses, providing the hypothesis generator with the means of identifying the roles various pieces of knowledge might play. A scheduling mechanism, similar to the agenda-based approach used in OTIS [Kerber, 1988], could also use these tags as a basis for evaluating the expected quality of the hypotheses produced by the knowledge sources and execute the most promising ones first.

The development of knowledge sources cannot proceed in complete isolation: the particular choice of representation may affect the implementation of the hypothesis incorporation mechanism, since it posts the necessary state information to the blackboard for the hypothesis generator to use. The selection of the specific representations and cueing mechanisms is beyond the scope of this thesis: it must be addressed during development of the hypothesis generation component and its interaction with the hypothesis incorporation mechanism.

3.1.1.3 Extensions to the hypothesis generator

The previous discussions described requirements and behaviors of the hypothesis generation component as originally conceived, with the exception of the experiential learning element, Koala, which evolved during the course of OXGate development and experimentation. In the experiments of Chapter 6, Koala gains experience from learning a particular concept and applies it to improve the concept description when relearning the same concept. This experience may also prove useful for learning other concepts in the same domain or similar domains. Applying experience gained in one situation to benefit learning in a novel situation is a potentially powerful method of hypothesis generation.

Another capability suggested during the development of OXGate involves the management of hypotheses rejected by the hypothesis ordering mechanism. Section 4.2.1.1 describes the expected impact of discarding apparently useless hypotheses: they may be

needed later in the decision-tree assembly process. The experiments presented in Section 5.4.4.2 show that reintroducing the rejected hypotheses can improve the quality of the final decision-tree, but at a substantial cost in processing time. A promising alternative to recovering the rejected hypotheses is to rely on the hypothesis generator to regenerate certain hypotheses when there is sufficient reason to do so. Typically, the knowledge sources of the hypothesis generator must be developed so they respond only to changes in the system state; otherwise, they would repeatedly suggest the same hypotheses and nullify any advantage of hypothesis ordering. However, the hypothesis generator management procedures should be made intelligent enough to recognize when the decision-tree assembly has reached a terminal state with a sufficiently non-homogeneous set of data and an alternative approach to hypothesis generation should be attempted. Such a capability is a topic for future study as proposed in Chapter 7.

Finally, future research should investigate the incorporation of an interface to the hypothesis generator for interaction with the user of the system. Such an interface would allow the user or developer to modify easily the knowledge available to the hypothesis generator. The user could more readily experiment with the effects of particular pieces of knowledge, apply knowledge incrementally to guide OXGate through induction of a particularly difficult concept, or provide other types of guidance to the system such as dynamically modifying the evaluation method and parameters. Another application of a user interface could be to request confirmation of intermediate results during induction. Muggleton applies such a user interface to allow his systems DUCE [Muggleton, 1987] and CIGOL [Muggleton and Buntine, 1988] to ask the user (or *oracle*) whether induced rules are correct, and to request names for intermediate concepts.

3.1.2 Hypothesis ordering

The source of inductive power in OXGate is the freedom of the hypothesis generator to liberally suggest potentially relevant hypotheses, extending the concept description language in the hope of providing the necessary descriptive elements. With this freedom comes the risk of saturating OXGate by prolific hypothesis generation. Hypothesis ordering is one approach for avoiding this saturation. The hypothesis ordering mechanism serves as a filter between the hypothesis generation (generate) and hypothesis evaluation (test) components. It performs an early assessment of all proposed hypotheses, identifies the most promising ones, and rejects the least promising.

Since proper development of the hypothesis generation component is practical only with a means to contain the effects of an explosion of hypotheses, the development of the hypothesis ordering mechanism is the primary implementation and experimental thrust of this thesis. Hypothesis ordering is described in detail in Chapter 4 with experimental support provided in Chapter 5. Simply put, hypothesis ordering is a heuristic approach to assessing and sorting hypotheses based on multiple criteria. The input to hypothesis ordering is the pool of available hypotheses. A competitive evaluation is used to partition the pool into three subsets: the most promising hypotheses, potentially useful hypotheses (retained for future consideration), and the least promising hypotheses. The least promising hypotheses are removed from the pool, and the other two subsets become the output of the hypothesis ordering component for posting on the blackboard.

3.1.3 Hypothesis evaluation

Hypothesis evaluation is the component of OXGate that compares hypotheses against the training examples to determine the next hypothesis to incorporate into the decision-tree. I experimented with several approaches during the earliest stages of the development

of OXGate, and converged on one that operates well. The specific measures used for evaluation, *confirmation* and *simplicity*, are discussed in detail in Sections 4.2.1.2 and 4.2.2. The confirmation is a measure of how well the training data support a hypothesis and is user-selectable to aid experimentation. The simplicity is an estimate of the cost of testing the hypothesis. The hypothesis evaluation component uses a procedural evaluation approach: find the hypothesis with the best confirmation and, in case of ties, choose the simplest. In the case in which multiple hypotheses have identical confirmation and simplicity measurements, OXGate selects the first one it happens upon.

During the operation of OXGate, the most promising hypotheses (provided by the hypothesis ordering mechanism) are evaluated against the full set of training data. The chosen hypothesis is then passed to the hypothesis incorporation component for inclusion in the concept description. If no hypothesis is sufficiently applicable to the training data, the hypothesis evaluation component will request additional hypotheses. The hypothesis ordering component will provide the most promising of the set of potentially useful hypotheses it kept in reserve. The hypothesis evaluator will continue to request additional hypotheses until it either finds an acceptable hypothesis or none are left. This process is described in greater detail in Chapter 4.

3.1.3.1 Some comments on evaluating hypotheses

The principal research issue for hypothesis evaluation is the measure used to determine hypothesis quality or *credibility*. Several researchers have proposed evaluation measures of a form best described as the product of a measure of the evidential support for the hypothesis and an estimate of the "goodness" of the hypothesis apart from the data. This section briefly describes some of these measures.

Hartmann *et al.* [1982] use a measure of hypothesis quality consisting of the amount of information provided by the hypothesis (confirmation) divided by a *generalized efficiency*

measure. The efficiency measure provides an estimate of the quality of the decision-tree, defined by the measure(s) of tree goodness the algorithm is attempting to optimize. Possible measures include the average tree depth, average cost of testing, and storage requirements. One typical measure of testing cost is the inverse of the simplicity: $cost = 1/simplicity$. This approach implies that quality can be defined as the product of the confirmation and the simplicity.

Watanabe [1985] estimates the credibility of a hypothesis, the *inductive probability*, by the product of its confirmation (probability of the data D given the hypothesis H) and its plausibility (the prior probability of the hypothesis), divided by the prior probability of the data. This inductive probability is simply the well-known Bayesian formula: $p(H|D) = p(D|H)p(H)/p(D)$. Since the prior probability of the data $p(D)$ is constant for all hypotheses tested at a decision point, the measure of hypothesis credibility can be simplified to be the product of its confirmation $p(D|H)$ and plausibility $p(H)$. The plausibility, or prior probability of the hypothesis, is often estimated by the simplicity of the hypothesis as discussed in Section 4.2.2. Hence, for comparing hypotheses, the quality measure is again the product of the confirmation and the simplicity.

When using the Bayesian method, the confirmation may be estimated as the percentage of data that is correctly classified (see *average purity*, Section 4.2.1.2). This measure does not map smoothly to the information-gain: for every value of purity there are multiple possible values of information-gain, and vice versa, depending on the mix of correctly and incorrectly classified positive and negative examples. Brieman *et al.* [1984] address the inadequacies of the purity measure and propose a family of *convex* measures to more appropriately reward the hypotheses that result in purer nodes.

Another related measure of hypothesis quality is the *J-measure* [Goodman and Smyth, 1988]. The J-measure is also the product of confirmation and simplicity. The confirmation

measure is similar in structure to the information-gain, and appears to fall within the class of convex functions defined by Brieman *et al.* [1984].

All of the approaches presented above indicate a common theme: hypothesis quality can be measured as the product of a measure of confirmation and simplicity. Despite this theoretical agreement, use of the product for hypothesis evaluation did not produce well-behaved decision-trees in early investigations during the evolution of OXGate (not presented). Small changes in the training data resulted in radically different decision-trees. The decision-trees tended to overspecialize to the training data, resulting in complex structures with replicated subtrees and poor *predictive accuracy* (Section 4.1). One possible explanation is the interaction between the measures of simplicity and confirmation. A hypothesis that is twice as complex as another would require at least twice the value of its confirmation to be selected: a difficult objective to achieve for any of the confirmation measures. The product form appears to unfairly reward the simplest hypotheses, effectively obviating the benefits provided by the constructed hypotheses. For these reasons, the product of confirmation and simplicy for hypothesis evaluation was rejected in favor of the procedural approach described on page 55.

3.1.4 Hypothesis incorporation

The final stage of the processing cycle is the incorporation of the hypothesis chosen by the evaluator. The hypothesis incorporation component determines the form of the resultant concept description and assembles it accordingly. The chosen hypothesis, the list of hypotheses it belonged to, and the set of training instances used to select it are acquired from the blackboard. The effects of the decision to accept the selected hypothesis are then propagated to the blackboard for use by the rest of the system.

In OXGate the resultant concept description is a decision-tree. Therefore, hypothesis incorporation involves splitting the set of relevant instances into two subsets, those

classified as positive by the hypothesis and those classified as negative, and allocating them to the left and right branches of the developing tree. After removing the hypothesis from the pool of hypotheses, the remainder of the pool is assigned to each branch and a processing cycle is initiated for each branch.

The hypothesis incorporation component is also responsible for posting the state information regarding the developing concept onto the blackboard. The state information is intended to be used by the hypothesis generator to propose new hypotheses. Generally, development of this capability would proceed in parallel with implementation of the hypothesis generator as the information requirements become known. However, a decision-tree is a simple structure, so the basic state information consists only of the selected hypothesis and the current tree. The baseline hypothesis incorporation capability is readily implemented.

Another possible function of hypothesis incorporation would be to filter out hypotheses that are logically implied or contradicted by the chosen hypothesis. For example, if the chosen hypothesis is (*color* = *red*), then for the positive branch, all hypotheses of the form (*color* = *xxx*) could be removed. Other hypotheses built using *color* may also be removed, as long as they are implied or contradicted by selecting (*color* = *red*). The hypotheses ((*color* = *red*) OR (*shape* = *oval*)), ((*color* = *blue*) OR (*color* = *green*)), and (NOT (*color* = *red*)) are candidates for removal. The hypothesis ((*color* = *red*) AND (*shape* = *oval*)) is not a valid candidate for removal because the component (*shape* = *oval*) may be required later. It should be removed only if the hypothesis (*shape* = *oval*) is present.

In the negative branch, all consequences of the negation of the selected hypothesis can be removed as well as all specializations of the hypothesis. In the example, (NOT (*color* = *red*)), and ((*color* = *red*) AND (*shape* = *oval*)) would be candidates for

removal. The hypothesis ((*color* = *red*) OR (*shape* = *oval*)) should not be removed unless the hypothesis (*shape* = *oval*) is present.

Because of the flexibility of the hypothesis representation language, locating all logical implications and contradictions of a hypothesis is a computationally expensive task. Preliminary experiments during the evolution of OXGate (not presented herein) used a simple hypothesis filter to remove some of the more obvious implications (simple conjunction, simple disjunction, and negation). Filtering proved to be slightly beneficial only when a large percentage of hypotheses were removed (when the number of possible values the pertinent attribute could take on was large) and when there was a high evaluation cost per hypothesis (when the set of instances being considered was large). When used in conjunction with hypothesis ordering, this type of filtering was always detrimental to the processing speed and provided no evident improvement in accuracy. Performance was best when the task of removing the useless hypotheses was left to the hypothesis ordering mechanism: the useless hypotheses would fail to be supported by the instances associated with the branch and would be quickly rejected. Based on these early results, hypothesis filtering was disabled for the experiments of Chapters 5 and 6.

3.2 Integration of the Pieces

This section presents an extended example of opportunistic constructive induction to highlight the interactions among OXGate components. The example provides a demonstration of several uses of domain knowledge for hypothesis generation and ordering during the construction of new terms in the domain of mineralogical classification. In its strictest sense, classification assumes complete knowledge so that, given the values for the relevant attributes, the system can deductively identify and name the correct concept. Pure induction is the inverse of this process, attempting to identify the relevant

attributes given examples of the concept. The two processes converge when complete knowledge is unavailable and induction is required to bridge the gaps in the classification process, and the results of partial classifications can be used to further the inductive process.

Mineralogy is a complex and inherently noisy domain which could benefit significantly from the data-handling capabilities of computers. There exist several thousand catalogued minerals, yet the classification of many of them is subject to debate among the experts. One reference catalogues 2600 minerals, but admits the data are flawed, being an attempt to compile varied and sometimes disparate sources [Roberts *et al.*, 1974]. Not only do the experts disagree about the classification of certain substances, but in some instances the values listed for certain attributes of a single mineral are mutually exclusive, physically impossible combinations. Thus, this domain has built-in class and attribute errors with no means of clearly determining the precise values. A mineral classification system must, therefore, have a probabilistic or statistical mechanism for assessing the best fit to these uncertain values, i.e., a mechanism well suited for instantiation through induction.

The list of attributes potentially useful for mineralogical classification is large and varied, and the attributes available in the laboratory differ from those available in the field. Some attributes are measurable physical characteristics such as chemical composition, specific gravity, and behavior under certain laboratory tests. Some attributes are inferred, such as the inference of internal crystallographic structure from the crystal shape and cleavage planes. Others are observational/subjective characteristics such as luster, appearance of outgrowths, color, and behavior under stress. Still others are relational, such as hardness, the ability of one mineral to scratch another, which is measurable only with respect to other minerals. To complicate matters further, some of the attributes display an intimate dependency upon others. For example, the "behavior under stress"

mentioned above, termed *tenacity*, is frequently dependent upon the geometry of the crystal structure and the axes of measurement. along one dimension the mineral may be flexible, but along another it may be brittle. In short, the number of potentially useful attributes is large, and fo: any particular sample the measured values will likely be incomplete.

Deterministic classification of individual minerals appears to be best accomplished by a two-stage approach [Dennan, 1959]. The first stage is to classify the mineral into a family group through the recognition of familial attributes composing that group. The second stage is to discriminate the mineral from others within the family using distinguishing attributes. This same approach can be used during induction by allowing knowledge gained through the first stage of processing to be used to focus the second stage. The resultant concept description is then constructed from the partial descriptions created in each stage.

One example of this iterative approach proceeds in the following fashion. Knowing that the domain is mineralogy (a state posted on the blackboard), the system can immediately focus on considering the complex attribute *chemical composition* first, as suggested by domain knowledge. There exist eight mineral *classes*, distinguished by major chemical components.[2] This knowledge allows the chemical formulae of the examples to be procedurally partitioned with respect to these components, so that each example can be temporarily represented by a handful of chemical features. The hypothesis

[2]The eight classes of minerals and their distinctive formulae features are:

I.	Native Elements (gold, silver, copper, lead, etc.)	$(Au \vee Ag \vee Cu \vee Pb \vee ...)$
II.	Sulfides, Sulfosalts	(S)
III.	Oxides, Hydroxides	$(O \vee (O \wedge H)) \Rightarrow (O)$
IV.	Halides	$(Cl \vee F \vee Br \vee I \vee At)$
V.	Carbonates, Nitrates, Borates	$((C \vee N \vee B) \wedge O_3)$
VI.	Sulfates, Chromates, Molybdates, Tungstates	$((S \vee Cr \vee Mo \vee W) \wedge O_4)$
VII.	Phospates, Arsenates, Vanadates	$((P \vee As \vee V) \wedge O_4)$
VIII.	Silicates	$(Si \vee O_4)$

Of these eight classes, the last four would be the primary features to test first since the constituents of classes I-IV can appear as components of minerals in classes V-VIII.

generator would first propose hypotheses formed from the last four classes, for example, (*contains* ((C∨N∨B)∧O_3)). If these are insufficient (the hypothesis incorporation component indicates a failure to find a suitable hypothesis), then the hypothesis generator will create hypotheses from the features for classes I through IV. Induction over this focused feature space can proceed quickly and should result in the recognition of the statistical class of minerals to which the positive examples belong.

Once the selected hypothesis is incorporated into the decision-tree and posted to the blackboard, the knowledge sources in the hypothesis generator would recognize the mineral class. Now other knowledge can be used to refined the description, such as the tendency of one element to be substituted for another within that class. This substitution of elements causes a condition known as *isomorphism*, where a mineral series will show a continuous change in chemical composition without a change in form. Isomorphisms occur due to the substitution of foreign ions into the crystal lattice whose properties (radius, valence, ionic potential) are similar to the expected ions. The challenge occurs in the identification of minerals at certain points along this composition continuum, especially when considering that some degree of contamination occurs in every mineral and must be allowed for in the classification. The knowledge of which elements to look for and which substitutions to expect is an application of domain knowledge readily invoked by the determination of the mineral class.

As an example, suppose the initial processing has determined the mineral belongs to the silicate ($SiVO_4$) class. One of the families of minerals within this class is the olivine series, displaying a complex isomorphism caused by a three-way substitution between magnesium (Mg), iron (Fe) and manganese (Mn) as illustrated in Figure 3.2. The system would attempt to isolate the concept to this family by proposing the hypothesis (*contains* (Mg∨Fe∨Mn)) along with other hypotheses potentially useful within the silicate class.

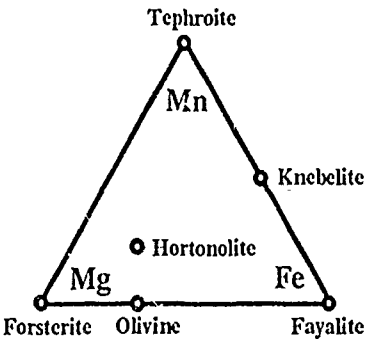


Figure 3.2 Mineralogical Isomorphism. The olivine series of the silicate class of minerals. The triangle depicts the isomorphism caused by the three-way mutual substitution of magnesium (Mg), iron (Fe), and manganese (Mn) in the crystal lattice. The locations of the minerals on the triangle indicate the typical ratios of the three elements. Although the triangle depicts precise locations for the minerals (for example, the composition of olivine has a ratio of Mg to Fe atoms and, therefore, the ratio of ideal forsterite to ideal fayalite, of 63:37), any ratio is possible in nature; hence, the boundaries between minerals in the series are imprecise.

Since chemical analysis of any sample in this series would almost certainly yield some amounts of Mg, Mn, and Fe, and since the boundaries delineating the particular minerals are unclear, chemical analysis would serve to identify only the mineral family or series. In this example, the evaluation component would select the hypothesis (*contains* (MgVFeVMn)). The hypothesis would be incorporated into the decision-tree and posted to the blackboard, and this change of state would let the hypothesis generator know that all the examples of the mineral belong to the olivine series. Armed with this newly established hypothesis, the system can begin the third stage of processing: finding the features to distinguish the positive examples of the mineral from the rest of the family. At this point, two things happen to ease the inductive process. First, the number of examples to consider is reduced since only those belonging to the olivine series need be considered. Second, the establishment of the mineral series should invoke additional domain knowledge to suggest discriminating attributes (such as hardness) within the family as well as discard attributes known to be useless (such as crystallographic structure in

an isomorphism). This has the net effect of reducing the size of the attribute/hypothesis space while focusing attention on the attributes most likely to be useful. Meanwhile, the hypothesis ordering mechanism also assists by focusing attention on the most promising hypotheses and filtering out the least promising ones. The inductive process should quickly proceed in a fairly conventional fashion to the description distinguishing the mineral from other members of its family. This description, in conjunction with the characteristic description of the family, forms the basis for future classification of the mineral.

3.3 Summary

This chapter described OXGate, the prototype opportunistic constructive induction system. OXGate provides a domain-independent development and testbed environment for the application of knowledge to guiding decision-tree construction. The four major components of OXGate have been implemented to varying degrees, reflecting their roles in the establishment of a baseline capability.

The hypothesis generator component provides the richest area for future development. Knowledge can be applied for hypothesis generation in several fashions: explicitly proposing hypotheses and constructive operators, retracting hypotheses and operators, and reasoning deductively about changes in the state of induction to determine which procedures or knowledge to invoke. A central controller mechanism would provide an interface to the blackboard to interpret the state changes posted by the hypothesis incorporation component, resolve conflicts between knowledge sources, and ensure that all proposed hypotheses are posted to the blackboard in the proper format. Development of these capabilities is the next logical thrust for future research.

The hypothesis ordering and hypothesis evaluation components are essentially complete, providing the fundamental capability for exploration of the hypothesis generation mechanism and its interaction with hypothesis incorporation. Hypothesis ordering is implemented with a heuristic filtering mechanism using multiple evaluation criteria for hypothesis selection. It identifies the most promising hypotheses for consideration by the hypothesis evaluation component and rejects those appearing useless. The hypothesis evaluation component assesses the hypotheses provided by the ordering mechanism against the full set of training instances to select the best hypothesis. The hypothesis incorporation component then adds the hypothesis to the decision-tree, partitions the training data into the appropriate subsets, and posts these changes to the blackboard for use by the hypothesis generator.

Of the four major components of OXGate, the hypothesis generation and ordering mechanisms are the most interesting and, before this thesis, the least developed. Tractable investigation of hypothesis generation requires an operational hypothesis ordering capability, consequently, the latter is developed first. Hypothesis ordering is discussed in detail in the next chapter, followed by extensive experimental results in Chapter 5. Preliminary investigations into some uses of domain knowledge for hypothesis generation are presented in Chapter 6.

CHAPTER 4

HYPOTHESIS ORDERING

Hypothesis ordering was described in Section 3.1.2 as the second component of abduction. It is the mechanism for assessing the potential utility of generated hypotheses and selecting the most promising ones before testing the hypotheses against the full body of evidence. Hypothesis ordering attempts to locate the most promising of the generated hypotheses and discard the least promising, using a limited amount of *evidential* information (data) augmented by available *extra-evidential* sources (knowledge and biases).

An example of this behavior can be found in the scientific discovery process. The scientist briefly entertains a number of possible explanations of the observed data, using knowledge of the domain to create hypotheses and screening out many as being implausible based on additional knowledge. The surviving hypotheses, those generated in this first step, then compete against each other to determine which are worthy of thoughtful consideration or experimental assessment. During this competition, readily accessible evidential information is used to refute obviously inappropriate hypotheses and support the more promising ones, and extra-evidential considerations (such as simplicity or elegance, ease of testing, analogues in other domains, and the scientist's intuitions) are applied to establish a preferential ordering. Only the subset of hypotheses deemed most promising is retained for the complete evaluation procedure of induction proper.

Since this thesis specifically addresses the assembly of decision-trees, the operational use of hypothesis ordering can be described more concretely. At any point during the assembly process, a pool of hypotheses is available for consideration, potentially restocked regularly by the hypothesis generator. The hypothesis ordering mechanism examines each

hypothesis in the pool at the current node of interest and produces a global estimate of the potential utility of the hypothesis for the current node and future subordinate nodes.

In this thesis, hypothesis ordering uses a *multiple-objective evaluation* method to estimate the predicted hypothesis utility [Seshu *et al.*, 1989]. The method is heuristic; therefore, to avoid relying too much on the outcome, the method is used to highlight the hypotheses appearing to be very good or very poor relative to the rest of the hypotheses. These two classifications imply three general categories of predicted hypothesis utility, which map into three subsets of hypotheses: the most promising, the potentially useful, and the apparently poor. The subset of most promising hypotheses, herein known as the *primary subset*, contains the hypotheses with the highest predicted utility, warranting precedence during hypothesis evaluation. The subset of potentially useful hypotheses, the *secondary subset*, contains those hypotheses that are not outstanding in either extreme, and so are held in reserve for later use: they do not warrant immediate rejection, but they also do not appear to be immediately useful at the current node. The subset of apparently poor hypotheses, the *rejected subset*, contains those hypotheses with such a low predicted utility relative to the other hypotheses that they are removed from further consideration.

During the operation of OXGate, hypothesis evaluation is applied first to the hypotheses in the primary subset, searching for the best hypothesis given the training data. If none of these hypotheses is sufficiently applicable, the secondary subset is considered. However, rather than evaluate all of the hypotheses in the secondary subset at once, the secondary subset is first submitted to the hypothesis ordering mechanism for additional partitioning into primary, secondary and rejected subsets,[1] and only the hypotheses in the

[1]Since the hypotheses compete against each other to determine the partitioning (Section 4.2.5), hypotheses assigned to the secondary subset during initial competition can be reassigned to the primary or rejected subsets when competing among only themselves. This process has the effect of dynamically relaxing the requirements for membership in the primary and rejected subsets.

new primary subset are evaluated. If none of these hypotheses is adequate, the process will repeat using the new secondary subset. The recursive partitioning of the secondary subset will continue, evaluating the best of the available hypotheses and rejecting the worst, until a suitable stopping point is reached. Partitioning will stop upon finding an acceptable hypothesis, reaching a minimum number of hypotheses (below which the overhead of ordering offsets any gains), or exhausting the set of hypotheses. The recursive nature of this order-evaluate cycle allows OXGate to utilize more than the original three categories while creating the refinements only when necessary. This approach avoids an undesirable reliance on preset partitioning thresholds and an oversensitivity to the results of the multiple-objective evaluation, while only incurring the overhead of highly refined partitioning upon demand.

When the best available hypothesis is found, it is incorporated into the decision-tree as the test at the node of interest. The union of the original (for the current node) primary and secondary subsets becomes the pool of hypotheses for the next stage of decision-tree assembly, augmented as appropriate by new suggestions from the hypothesis generator. In addition, some hypotheses logically nullified by the chosen hypothesis may be filtered out as part of the hypothesis incorporation process (see Section 3.1.4).

4.1 The Expected Tradeoff

The goal of hypothesis ordering is to reduce the number of hypotheses that the more expensive evaluation mechanism must examine. Since hypothesis ordering is a heuristic approach, one would expect a tradeoff, gaining processing speed at the expense of another measure of performance. The most obvious measure potentially affected is the accuracy of the resultant classifier. Early in the decision-tree assembly process hypothesis ordering may reject or shadow hypotheses necessary further down in the tree for

maximal resubstitution accuracy (maximum classification accuracy on the training data). As will be discussed in Section 4.2.3, forcing the hypothesis ordering mechanism to retain all *primitive* hypotheses can generally remove this effect. However, even in the case in which an imperfect decision-tree is created (one with non-maximal resubstitution accuracy), the performance of the decision-tree on independent test data is not necessarily adversely affected.

The performance measure typically used in machine learning literature to compare inductive systems is the *predictive accuracy.* Predictive accuracy (herein known as *accuracy*) is a measure of the power of the generalization: the ability to predict the proper classification of unseen data. Several factors affect the generalization performance, including the degree to which the training data represent the test data, and the classifier's sensitivity to noisy or rare data. Accepted methods of obtaining robust generalization often sacrifice a portion of the resubstitution accuracy to gain in predictive accuracy, accomplished in decision-tree classifiers by removing (*post-pruning*) or preventing the creation of (*pre-pruning*) certain branches of the tree [Breiman *et al.*, 1984].

The hypothesis ordering mechanism, in removing globally unpromising hypotheses, can be considered to be related to pre pruning since it may force early termination of the development of the decision-tree as the pool of promising hypotheses is depleted. The resultant decision-tree is similar to that of a post-pruning approach: a decision-tree developed from the most well-represented and consistent training examples, with the tacit assumption that it is better to ignore rare examples (which might represent small disjuncts or which might simply be noise) in exchange for the potential of better generalizations [Breiman *et al.*, 1984]. Given the nature of generalization and the tenuous relationship between resubstitution accuracy and predictive accuracy, the use of hypothesis ordering does not imply a reduction in predictive accuracy. As will be seen in Section 5.3, when compared to the decision trees developed using all of the available hypotheses (primitive

and constructed), hypothesis ordering can provide substantial speed improvement, with little or no degradation of predictive accuracy.

4.2 Mechanisms for Ordering Hypotheses

Hypothesis ordering involves two distinct processes: assessment of the potential utility of the hypotheses and division of the pool of hypotheses into the three disjoint subsets described earlier. In this thesis, the assessment is performed through the application of *multiple-objective evaluation*, assessing each hypothesis along several dimensions and combining the results to provide an indication of potential utility relative to the remaining hypotheses. The methods of combining the evaluation dimensions and determining the disposition of the hypotheses are varied and belong to the field of multiple-criteria decision making [Yu, 1985]. A significant contribution of this research effort is the analysis and empirical assessment of several approaches for hypothesis ordering.

In this thesis, three dimensions are used for multiple-objective evaluation. The first dimension, the *Quick-Look*, is an estimate of the confirmation of the hypothesis: the performance of the hypothesis on a subset of the training data. The second dimension is the *simplicity* of the hypothesis. The third dimension is the *primitiveness*, indicating whether the hypothesis is a primitive derivation from the original description language (Appendix B). Abstractly, the Quick-Look is an evidential measure, the simplicity is a context-free extra-evidential measure, and the primitiveness is a context-dependent extra-evidential measure. Other measures are also possible, such as the degree of confidence ascribed to a hypothesis by its domain knowledge-based generator or the amount of combined support disjoint knowledge sources provide a hypothesis. The investigation of additional/alternative measures is suggested as future work (Chapter 7).

4.2.1 The Quick-Look

The Quick-Look is a heuristic method I devised for estimating the evidential support for the hypotheses. It is a measure of the confirmation of the hypothesis over a randomly drawn subset of the training data. The primary motivation for using the Quick-Look is to identify the hypotheses strongly supported or strongly discounted by the evidence. Partial justification for using the Quick-Look is in [Etzioni, 1988], where a hypothesis filter is established, based on a subset of the training instances and the desired reliability of the filter. In Etzioni's approach, some good hypotheses may be filtered and some poor hypotheses retained, depending on how well the sample represents the training data. The probability of these events occurring can be made arbitrarily small by increasing the sample size. The Quick-Look faces the same dependency on the sampling, but since it is only one of several dimensions in the evaluation, and since the hypotheses are competing against each other rather than an established threshold, the adverse effects of unfavorable sampling are diminished.

The use of the Quick-Look is also similar to the use of windowing in ID3 (Section 2.4.1), but the resemblance is superficial. Windowing uses a subset of the training data to develop a decision-tree. If the remaining training data are not classified correctly, some of the misclassified examples are added to the subset and the tree is redeveloped. Windowing reduces the amount of computer memory required for decision-tree development by limiting the number of examples to consider. The description language remains untouched. The Quick-Look also operates on a subset of the training data, not to develop the decision-tree, but to focus the description language. The decision-tree is developed by the hypothesis evaluation component of OXGate, which uses the full set of training data and relies on the constrained description language to reduce the size of the search space, thereby saving processing time.

4.2.1.1 A potential problem and some solutions

The greatest difficulty Quick-Look faces is the potentially detrimental effect that highly disjunctive concepts may have on its utility as a predictive measure of hypothesis quality. Some of the islands of a highly disjunctive concept may be only marginally represented by the training data, and some islands may not be represented at all. While this is a common problem for all inductive systems, Quick-Look exacerbates the sparseness of the representation by taking only a subset of the training data, reducing the likelihood of selecting any particular example even further. For example, if an island is represented by 5 of a possible 1000 examples in instance space, a training set of 500 randomly drawn examples (with replacement) has a 92% probability of selecting at least one example from the island. A Quick-Look subset of 10% of the training data (50 examples) has only a 22% probability of containing an example from the island. Therefore, highly specialized hypotheses, applicable only to the smaller islands, will incur a larger probability of being rejected relative to their more robust counterparts. Some of the possible approaches to alleviating this adverse condition are:

1. Assume that the rare examples are indistinguishable from noisy data, ignore them if necessary, and hope the generalization provides good predictive accuracy on the test data. As discussed in Section 4.1, this is the philosophy behind decision-tree pruning and is a common method of dealing with exceptional examples.

2. Increase the size of the Quick-Look sample, thereby improving the likelihood of selecting representative samples from each island. The drawback of this approach is the increased computational overhead, effectively negating the processing speedup available through hypothesis ordering. One possible approach to avoiding an unnecessarily large Quick-Look sample size is to adjust the sample size according to the *concept dispersion* or *variance* [Rendell and Seshu, 1990]. For concepts that are

highly dispersed in instance space, more samples from the sparser areas (areas with small, scattered islands) would improve the likelihood of selecting representative samples from the islands. The development of mechanisms for assessing the complexity of the concept and dynamically scoping the sample size is a topic suggested for future research (Chapter 7).

3. Rely on the hypothesis generator to produce hypotheses that are at the right level of generality or abstraction to predict the poorly represented islands and are supported by enough examples to be rated well by the Quick-Look. The generator must detect regularities in the data and propose hypotheses incorporating those regularities. This is not an additional requirement imposed by the use of Quick-Look, for predicting the missing islands is the fundamental challenge faced by any constructive induction system [Drastal and Raatz, 1989, Rendell and Seshu, 1990].

4. Artificially increase the proportion of positive to negative examples in the training set to provide denser islands, thereby increasing the probability of selecting an example from a smaller island. In the previous example, assume that of the 1000 examples in the instance space, 200 are positive. Doubling the number of positive examples in the training set (by representing each one twice) would raise the proportion of the examples on the small island from 5 out of 1000 to 10 of 1200. The Quick-Look sample of 50 examples would then have a 34% chance of containing an example from the island. Two potential effects of this approach would be an increase in the false positive rate and a tendency to overgeneralize. This approach is studied in Section 5.4.4.3.

5. Retain the set of primitive hypotheses at all times, disallowing their rejection regardless of their performance in the Quick-Look. This nearly guarantees maximal

resubstitution accuracy, since the primitive hypotheses provide the original description language for use in the lower branches of the tree if needed. At the same time, overspecialization to the training data may be avoided, with a corresponding increase in predictive accuracy, since the more potentially useful constructed hypotheses are also in the description language and may be incorporated higher in the tree. This approach is investigated in Section 5.4.4.1.

6. Retain the set of rejected hypotheses for possible use when the primary and secondary subsets prove insufficient. One likely result would be a substantial increase in the processing time, since the hypothesis ordering mechanism would have to reexamine many useless hypotheses. The experiments presented in Section 5.4.4.2 investigate this approach. A potentially effective alternative to retaining all rejected hypotheses is to rely on the hypothesis generator to regenerate certain hypotheses when there is sufficient reason to do so.

4.2.1.2 Measuring confirmation

Several measures are available to estimate the confirmation of a hypothesis, and are applicable to both the Quick-Look as well as the full hypothesis evaluation. Each of these measures involves the same basic process: test the hypothesis on the sample data, count the number of correctly and incorrectly identified positive and negative examples, and combine these four values into a single term. Various measurements have been explored in the works of Breiman *et al.* [1984], Hartmann *et al.* [1982], Goodman and Smyth [1988], Mingers [1989], and many others. The three measures examined in this thesis are the information-gain (IG), average purity (AvePur), and positive purity (PosPur).

In the following definitions, the references to *data* denote the training data testing at the decision-tree node of interest. These definitions and their formulae are applicable to

the Quick-Look when considering a randomly drawn subset of training data, as well as the hypothesis evaluation mechanism when using all the data available at a node.

Let the four measured values be
PG = the number of correctly classified positive examples
NG = the number of correctly classified negative examples
PB = the number of incorrectly classified positive examples
NB = the number of incorrectly classified negative examples

Then let
LT = the number of examples classified as positive $(PG + NB)$
RT = the number of examples classified as negative $(NG + PB)$
P = the total number of positive examples $(PG + PB)$
N = the total number of negative examples $(NG + NB)$
D = the total number of examples $(P + N)$

Information-Gain: The IG is a rough estimate of the information gained by choosing the hypothesis as the criterion for splitting at a node of the decision-tree. It is the difference between the entropy of the data at the node and the weighted sum of the entropies of the data in the two branches that would be created by the split. Contrary to the common use of information-gain, in OXGate the IG is signed to distinguish between a hypothesis and its antithesis.

The maximum entropy $S_{max} = S(P/D) + S(N/D)$ where $S(x) = -x \ln x$

The entropy of the positive (left) branch $S_+ = S(PG/LT) + S(NB/LT)$

The entropy of the negative (right) branch $S_- = S(NG/RT) + S(PB/RT)$

The total entropy of the split $S_{tot} = ((S_+ \times LT) + (S_- \times RT))/D$

The information-gain $IG = S_{max} - S_{tot}$

And if $PB/RT > PG/LT$, the sign of IG is made negative.

Average Purity: The AvePur is the percentage of all examples classified correctly by the hypothesis. $AvePur = (PG + NG)/D$

Positive Purity: The PosPur is the percentage of positive examples classified correctly by the hypothesis. $PosPur = PG/P$

Section 5.4.2 presents evidence showing that the information-gain is the preferred measure of the three for hypothesis confirmation. Several other inductive systems use the information-gain as the metric for evaluation; one of the most famous is ID3 by Quinlan [1986]. Common sense suggests that the measure used for the Quick-Look should be the same as for hypothesis evaluation since the Quick-Look serves to estimate the utility the hypotheses will be awarded during full evaluation.

4.2.2 Simplicity

The second measure used in the multiple-objective evaluation is the simplicity: the inverse of the cost of processing the hypothesis. Simplicity is often used as an estimate of the *plausibility* of the hypothesis or its *prior probability* [Watanabe, 1985]. Simplicity is a context-free, extra-evidential measure based solely upon the syntactic structure of the hypothesis, independent of the data.

The preference for the simplest hypotheses consistent with the data is a long-standing heuristic commonly known as Occam's Razor. More than a heuristic, this bias has a mathematical basis: a simple hypothesis consistent with the data is provably likely to be an approximately correct description of the true concept [Blumer *et al.*, 1987]. According to Dietterich [1990], regardless of the nature of the simplicity measurement there are relatively few simple hypotheses; therefore, a simple hypothesis is unlikely to be consistent with the data by chance and deserves preference over more complex hypotheses.

The exact nature of the simplicity measurement should reflect the cost that the decision-tree is intended to minimize [Hartmann *et al.*, 1982, Breiman *et al.*, 1984]. The

costs of the individual syntactic elements must be incorporated into the measurement, yet there is great liberty in determining what those costs should be. Counting the number of tests on the attributes is an obvious potential element of the measurement. However, the manner of treating the constructive operators is unclear: semantic differences between operators suggest a continuum of operator costs.

One approach is to simply use the inverse of the number of tests on attributes and to ignore the effect of constructive operators. One justification for this approach is the recognition that in real-world applications, the computational cost of performing the constructive operations can be negligible with respect to the cost of testing the attributes. For example, the primitive hypothesis (*color* = *red*) would require a single test: its simplicity is 1. The constructed hypothesis ((*color* = *red*) AND (*size* = *large*) AND (*shape* = *round*)) entails a simplicity of 1/3. The difficulty with this approach is that it often creates a decision-tree with poor readability and predictive accuracy: the measure is unable to distinguish between hypotheses of the forms (*test*), $f(test)$, and $f(g(h(j(test))))$. For example, it will not indicate a preference for $(X1 = T)$ over (NOT $(X1 = F)$).

Assigning a uniform cost to the application of each constructive operator is also not a completely satisfactory approach. The semantic differences between operators suggest a graded cost structure. Clearly $((\log_B A) > 3)$ is a computationally more costly operation than $((A + B) > 3)$, yet each consists of two attribute tests, the application of one operator, and a comparison to a constant.

The development of a practical and syntactically attractive approach to operator cost assignment should be the subject of future research. In this thesis, the Boolean operators AND, OR, and NOT have costs of zero: preliminary testing during the evolution of OXGate exhibited no noticeable, consistent benefit from assigning non-zero costs to

the Boolean operators. The operators EQP, MEMBER, and WHATIS have costs of one. Their use is described in Chapter 6.

4.2.3 Primitiveness

The third dimension of multiple-objective evaluation used in this thesis, *primitiveness*, is a binary indication of whether the hypothesis is primitive. Since the hypothesis ordering mechanism performs a global assessment of the initial pool of hypotheses at the beginning of decision-tree assembly, those hypotheses representing the smaller islands of the true concept description may be inadvertently rejected. They are simply overshadowed by the larger values of confirmation of the hypotheses applicable higher in the decision-trees. If the hypothesis generator does not regenerate the necessary hypotheses before they are required deeper in the decision-tree, the remaining hypotheses may be inadequate to completely classify the training data. Retention of all primitive hypotheses, or *primitives*, should provide the capability of achieving at least the maximal resubstitution accuracy available with a purely selective induction system. The experiments of Section 5.4.4.1 investigate the utility of retaining the primitive hypotheses.

In some cases, it is conceivable that the primitives need not be retained. For example, in an application in which a function of *volume* is found to be a useful constructed feature (constructed from the primitive features derived from the attributes *height, width,* and *length*), it is quite possible that the primitive features and the corresponding primitive hypotheses will never be used again. Generally, predicting whether a primitive used in one construction will be useful further down in the decision-tree is difficult at best without full knowledge of the target concept, therefore, caution must be used when discarding primitive hypotheses. This is a subject for future research.

4.2.4 Combining the measures

The selection of promising hypotheses for future complete evaluation is a process analogous to the way decisions are often made in business or other disciplines. the decision maker has a subordinate examine the space of alternatives and provide him with the set of most promising decisions for his consideration. Although the subordinate assembles a set of candidate decisions and the decision maker converges on a single one, both assess the possible decisions by means of *multiple-criteria decision making.* Similarly, in this thesis, multiple-criteria decision making is used in both the hypothesis ordering and the hypothesis-evaluation processes.

The central theme of multiple-criteria decision making is the evaluation of the potential costs and benefits of candidate decisions in order to identify the most desirable of the decisions. The process is known as *multiple-objective evaluation* (MOE): the evaluation of several measures (objectives) of costs or benefits, and the combination of those objectives into a measure of quality useful for comparing the decisions. A myriad of methods exist for MOE, many involving not only an attempt to optimize the choice of decision, but also to optimize the selection process itself [Yu, 1985]. This thesis examines four of the more tractable approaches. the use of *non-dominance*, weighted combination of the objectives, product of the objectives, and procedural use of the objectives.

4.2.4.1 Non-Domination

The most intuitively appealing of the MOE combination methods are those that do not attempt to produce a single comparative term for each hypothesis, but instead compare the hypotheses along each of the objectives and select the ones with the best overall performance. One method of this type is the identification of the *non-dominated*

hypotheses (NDII), a method I developed[2] based on the determination of the non-dominated regions of a decision space [Yu, 1985].

The non-dominated regions of a decision space are those regions containing decision points that are better than every other point in at least one evaluation dimension. The regions are the extrema of the convex hulls formed over the set of known decision points. As illustrated in Figure 4.1, the non-dominated points are never surpassed in every dimension by another point. The appeal of this approach is that the relative importance of the evaluation measures need not be established to identify which decisions to prefer: all of the decisions known to be inferior to at least one other decision are discarded. The use of the non-dominance approach for hypothesis ordering in OXGate involves broadening the definition of the non-dominated region, and using this new definition as the basis for dividing the pool of hypotheses into the three subsets described earlier. This extension is discussed in Section 4.2.5.1.

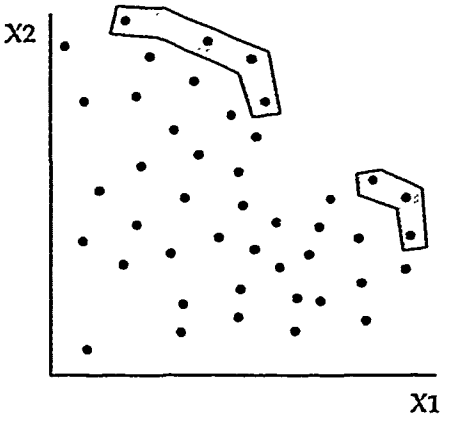


Figure 4.1 The Regions of Non-Domination. In this projection of decision space onto the two evaluation dimensions X1 and X2, the non-dominated decision points are highlighted.

[2]The use of a non-dominated region for hypothesis selection was suggested in [Seshu *et al.*, 1989]. Details of that implementation have not been published.

4.2.4.2 Weighted combination

One of the simplest combination methods in decision making is the linear weighting and combination of the evaluation measures, the *weighted multiple-objective evaluation* (WMOE) approach. If the relative importance of the evaluation measures is known with enough certainty and is quantifiable, then for decision D, a weight α_i can be assigned to each of the n measures X_{D_i}. The quantity Q_D represents the quality of decision D and is given by

$$Q_D = \sum_{i=1}^{n} \alpha_i X_{D_i}$$

The main drawback of the weighted combination approach is the need to quantify the relative importance of each evaluation measure. Often this commitment to fixed weights is unattainable or undesirable. The effective setting of the weights varies with the situation and the distribution of the known decision points, and locating the proper mix of weights is itself a MOE process. [Yu, 1985]

4.2.4.3 Product combination

Several sources use the product of a measure of confirmation and a measure of plausibility as the basis for estimating the overall quality of a premise (e.g., the J-measure [Goodman and Smyth, 1988], or the inductive probability or *credibility* [Watanabe, 1985]). These were discussed in Section 3.1.3. The product approach may tend to unduly favor the simplest hypotheses during hypothesis ordering. The Quick-Look confirmation measure must be over twice as large for the more complex hypotheses if they are to compete with the simplest hypotheses.

4.2.4.4 Procedural combination

When the relative importance of the evaluation measures cannot be quantified, but the precedence of the measures is known with certainty, the selection of the best decisions can be accomplished procedurally. The decisions are first ordered according to the most important objective. When decisions are equally valued over the most important objective, the second objective is applied to break the ties. The less important objectives are applied in the same fashion as needed. This approach forms the basis of the *lexicographical evaluation function* (LEF) used in the series of INDUCE and CLUSTER programs [Stepp and Michalski, 1986].

The procedural combination approach is not a useful approach for hypothesis ordering in DXGate since the goal of hypothesis ordering is to identify the *set* of most promising hypotheses, not the single best hypothesis. It also requires that the priorities of the objectives be fixed in advance; consequently, this method cannot respond dynamically to the composition of the set of hypotheses. Although empirical support of the unsuitability of the procedural approach for hypothesis ordering is not presented in this thesis, preliminary experimentation demonstrated this approach to produce generally poor decision-trees and to be unstable when used for hypothesis ordering, i.e., very sensitive to the composition of the randomly drawn Quick-Look data sets. While not particularly useful for hypothesis ordering, the procedural combination is the method of choice for the hypothesis evaluation mechanism: it produces decision-trees with the simplest of the most informative decisions at each node.

4.2.5 Separating the hypotheses

The second stage of hypothesis ordering is the separation of the pool of hypotheses into three subsets: primary, secondary, and rejected. Two approaches can be used to

determine the partitioning: setting of thresholds and competition among hypotheses. The competitive approach is the more desirable of the two because it allows the behavior of the partitioning to respond flexibly to the characteristics of the hypotheses and the training data. The establishment of fixed thresholds can result in a large number of hypotheses being retained when the data represent a simple concept, and the premature rejection of many hypotheses when the data represent a highly disjunctive concept, simply due to low absolute values of the measured confirmation. The competitive approach is dynamic since only the relative, and not the absolute, values of the measures are considered.

The non-domination approach to hypothesis ordering naturally places the hypotheses in competition against one another. For the other combination methods, a single value of predicted utility is computed for each hypothesis: this value can form the basis of a clustering approach that causes the hypotheses to gravitate into the three distinct subsets. Unfortunately, true clustering operations are computationally expensive and would curtail much of the processing speed benefits available with hypothesis ordering. Instead, an approach to approximating the clustering behavior is used: the hypotheses are ranked according to their predicted utilities, and a *histogrammatic* method partitions them into the three subsets.

Of the three dimensions used for the multiple-objective evaluation, primitiveness requires special treatment. Section 4.2.3 pointed out the predicted benefits of always retaining the primitive hypotheses. One way of implementing this requirement is to treat primitiveness as an objective of such high importance that the primitive hypotheses are always in the primary subset. The goal, however, is not to always have the primitive hypotheses in the primary subset, but rather to keep them out of the rejected subset. This is accomplished by performing hypothesis ordering using the confirmation and simplicity measures, and then transferring any primitive hypotheses relegated to the rejected subset into the secondary subset.

4.2.5.1 Non-Domination

The non-domination approach to hypothesis ordering (NDH) is an extension of the fundamental description presented in Section 4.2.4.1. By definition, non-dominated hypotheses are never surpassed in every evaluation dimension by any other hypothesis. If applied in OXGate, this constraint would force too much reliance on the confirmation measurement obtained by the Quick-Look. Since the Quick-Look provides only a rough estimate of the confirmation of the hypotheses, failure to tolerate a margin for error can result in an unwarranted restriction of the primary subset. By including tolerance bands for both the primary and secondary subsets, the pool of hypotheses can be partitioned into the "generally undominated" hypotheses (primary subset), the "heavily dominated" hypotheses (rejected subset), and the remaining hypotheses (secondary subset). Figure 4.2 illustrates the application of tolerance bands in two dimensions.

The tolerance bands are necessary only for dimensions susceptible to error. Simplicity is a known quantity with no error; therefore, it requires no tolerance bands. Tolerance bands may be useful, though, for low values of simplicity to avoid unwarranted

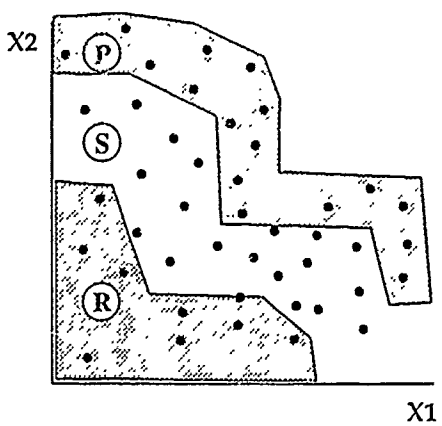


Figure 4.2 Using Tolerance Bands with Non-Domination. The use of tolerance bands avoids overreliance on the quality of estimations X1 and X2. The three bands represent the generally undominated hypotheses (primary subset, P), the heavily dominated hypotheses (rejected subset, R), and the remaining ones (secondary subset, S).

discrimination between marginally different hypotheses. For example, preferring a hypothesis with fifteen components over one with sixteen is too refined a discrimination (if based on simplicity alone) for use in hypothesis ordering. Since the constructed hypotheses used in the experiments with OXGate are relatively simple, tolerance bands are not used with the simplicity measure in this thesis.

The application of the primary tolerance band for the Quick-Look confirmation measure is illustrated in Figure 4.3. Conceptually, the approach for partitioning the hypotheses into the subsets proceeds as follows. First, the truly non-dominated hypotheses are determined (points A, B, and C in the figure). Second, the primary tolerance band is extended from these hypotheses. Any hypotheses falling within the banded areas are included in the primary subset, except for those that are dominated by another hypothesis by more than a tolerance width (for example, point D). Third, the hypotheses in the primary subset are removed from the pool and the process is reapplied to the remaining hypotheses using the secondary tolerance band to determine the secondary subset. The hypotheses remaining after this application are rejected.

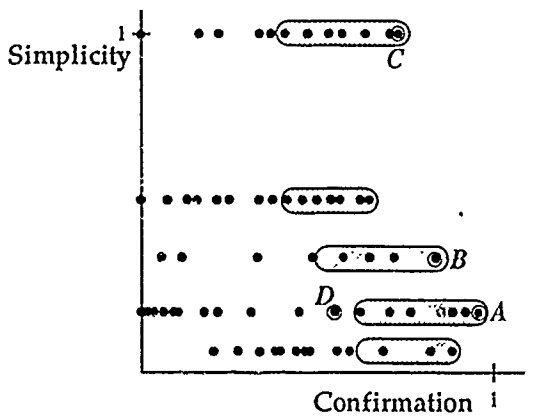


Figure 4.3 Non-Dominated Hypotheses in OXGate. The diagram shows the use of the tolerance band for the primary subset of hypotheses. The tolerance band for the secondary subset is not shown. The tolerance band is applied for the Quick-Look measure of confirmation, but is not necessary for the precisely determinable dimension of simplicity. (Note the discrete nature of the simplicity measure used by OXGate.)

4.2.5.2 Weighted and product combinations

Both the weighted and the product combination approaches to multiple-objective evaluation combine the evaluation dimensions into a single estimate of hypothesis utility. Using this estimate, the hypotheses can be ordered in a list. Partitioning this list into three subsets is a matter of finding suitable breakpoints, either by clustering, establishing fixed thresholds, or using a histogrammatic method, described below. Figure 4.4 illustrates the partitioning of the weighted combination in a two dimensional evaluation space. Figure 4.5 shows the partitioning for the primary subset using the weighted sum of confirmation and simplicity measures in OXGate.

The product combination is illustrated in Figure 4.6. It is easy to see why this form might not be useful for hypothesis ordering. The primary subset tends to favor hypotheses with $X1 \approx X2$ (confirmation $\approx$ simplicity) while avoiding the extremes. Moderately simple hypotheses with medium values of confirmation are preferred over complex hypotheses with high confirmations. If the hypothesis generator happens to propose the correct hypothesis to describe the target concept, but the hypothesis is fairly complex, the product form of the hypothesis ordering mechanism could inadvertently

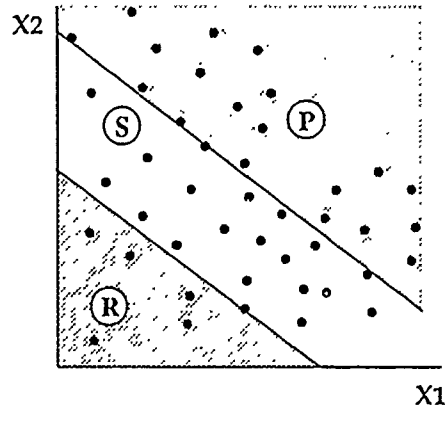


Figure 4.4 Partitioning Hypotheses with Weighted MOE. Key: (P)rimary, (S)econdary, and (R)ejected subsets.

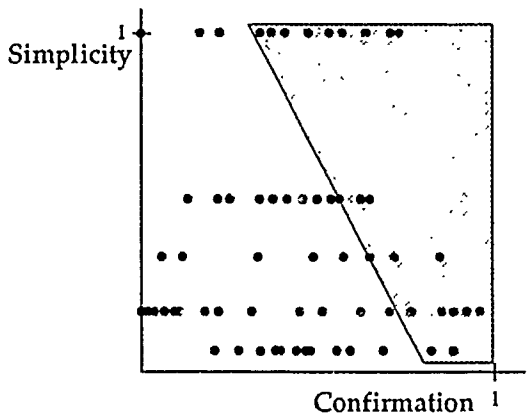


Figure 4.5 Weighted MOE in OXGate. This diagram illustrates the primary partition of hypotheses as applied to the weighted sum of the Quick-Look measure of confirmation and simplicity. (cf. Figure 4.3)

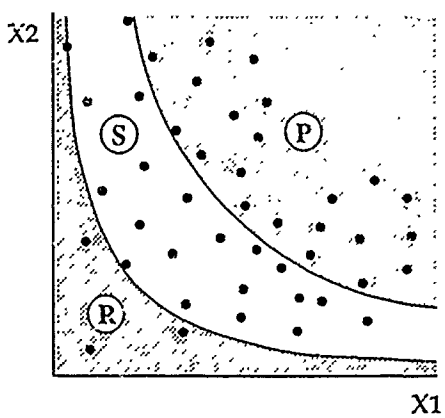


Figure 4.6 Partitioning Hypotheses with Product MOE. Key: (P)rimary, (S)econdary, and (R)ejected subsets.

discard it. Another difficulty with using the product form occurs with the information-gain estimate of confirmation. The information-gain is not a linear measurement: a hypothesis that correctly classifies twice as many examples does not necessarily yield twice as much information-gain. The product of information-gain and simplicity tends to favor a simple hypothesis even when a more complex hypothesis is correspondingly more accurate.

One method of partitioning an ordered list of quality estimates is by an approach I have loosely termed the *histogrammatic* method. Essentially, the process is analogous to that of an instructor who grades "on a curve." The instructor takes the distribution of student course scores and looks for suitable breakpoints within sensible windows of opportunity in order to define the boundaries between the grades of A, B, C, etc. The histogrammatic method examines the ordered list of quality estimates, searching for sufficiently large drops in quality between adjacent items in the list, as depicted in Figure 4.7. Acceptable drops occurring within defined windows are used as the breakpoints to partition the list into the primary, secondary, and rejected subsets.

The windows are currently defined in OXGate by a collection of parameters specifying: the minimum number of hypotheses in the primary subset, the minimum quality of

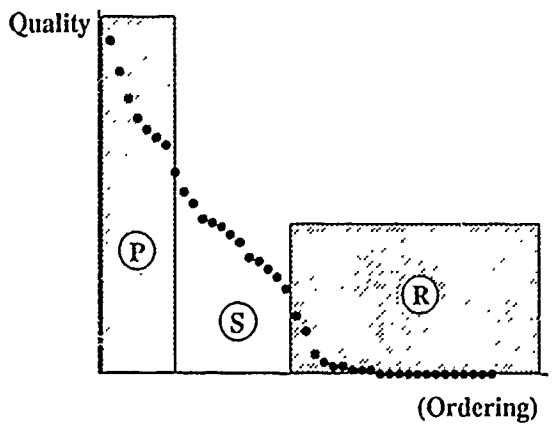


Figure 4.7 The Histogrammatic Approach to Partitioning.

hypotheses in the primary subset as a percentage of the highest quality in the list, the magnitude of the drop as a percentage difference between adjacent terms, and other related parameters to distinguish the secondary subset from the rejected subset. The major deficiency of this approach is its complexity: with so many parameters to adjust, it may be difficult or impossible to establish a set of parameters robust enough to apply properly to every possible pool of hypotheses. Alternative methods of partitioning the ordered list, such as self-clustering algorithms or the use of clustering attractors, may provide more robust and psychologically satisfying solutions. Clustering algorithms tend to be computationally expensive, however, and would add substantially to the processing overhead that hypothesis ordering is intended to reduce.

4.2.5.3 Procedural combination

The procedural combination and partitioning approach to multiple-objective evaluation can be accomplished in at least two ways. In the first approach, the hypotheses are placed on an ordered list. The most important objective is used first to order the hypotheses. As in the non-dominated approach, tolerance levels can be established to avoid placing too much reliance on the accuracy of the measurements. Equally valued hypotheses, or when using tolerance bands, similarly valued hypotheses, are then ordered by the second important objective, and so on. The ordered list is then partitioned in a manner similar to that of the weighted and product combination methods; however, now there is no single measure with which to compare adjacent hypotheses. The list must be partitioned according to some other criteria such as defining the subsets to be fixed percentages of the total number of hypotheses on the list. The effect would be to have the primary subset consist mainly of the hypotheses with the highest values of the most important objective regardless of the values of the other objectives. Similarly, the rejected subset would tend to consist of the hypotheses with the lowest values of

the most important objectives. The other objectives would affect only the placement of a hypotheses in a particular subset when the partitioning splits a group of hypotheses having equal or similar values of the most important objective. There appears to be no redeeming value to this approach: it generally ignores all but the most important objective, and the partitioning method is blind to the makeup of the pool of hypotheses.

An alternative procedural combination approach to multiple-objective evaluation is illustrated in Figure 4.8 with $X1$ as the most important objective and $X2$ as the next. The hypotheses with the highest value of $X1$ are considered first: the ones with the highest values of $X2$ are placed in the primary subset and the ones with the lower values of $X2$ are assigned to the secondary subset. The algorithm proceeds from the right towards the left in the figure, assigning the hypotheses with the best values of $X2$ to the primary subset for a particular value of $X1$, and the rest to the secondary subset. As the value of $X1$ approaches the middle of its range, the partitioning is adjusted so that only the hypotheses with the very best values of $X2$ are selected for the primary subset, the ones with the worst values of $X2$ are rejected, and the rest are assigned to the secondary subset. As the value of $X1$ approaches the low end of its range, the partitioning is readjusted so that the hypotheses with the best values of $X2$ are assigned

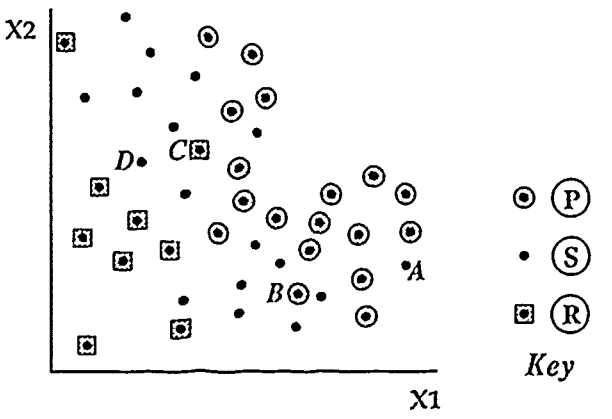


Figure 4.8 Partitioning Hypotheses with Procedural MOE.

to the secondary subset, and the remainder are rejected. Tolerance levels can also be incorporated into this algorithm to avoid being overly sensitive to the accuracy of the measurements.

This procedural combination algorithm suffers some of the deficiencies of the histogrammatic approach. Several parameters have to be used to specify the breakpoints to allocate hypotheses to the three subsets, as well as define the conditions for adjusting the partitioning behavior as described above. More importantly, this approach can exhibit an unacceptable anomaly as illustrated in Figure 4.8. Hypothesis A is assigned to the secondary subset since its value of $X2$ is less than that of the two other hypotheses with nearly the same value of $X1$. Hypothesis B is assigned to the primary subset since it has the greatest value of $X2$ for its value of $X1$. Yet, hypothesis A is clearly superior to B for each measure $X1$ and $X2$. Similarly, hypothesis C should not be rejected when D is retained in the secondary subset. This procedural approach for multiple-objective evaluation is simply too ill-behaved for use in hypothesis ordering.

4.3 Summary and Comments

Hypothesis ordering is the second step of the inductive process. Its function is to make initial estimates of the utilities of hypotheses, present the most promising of the hypotheses for rigorous evaluation, and reject the seemingly useless hypotheses. The hypothesis ordering component acts as a buffer between the hypothesis generator, which proposes hypotheses, and the hypothesis evaluation component which must evaluate the hypotheses against the full set of available training data. Since the evaluation against the full set of data is an expensive operation, hypothesis ordering is necessary to reduce the number of hypotheses the evaluator must consider.

Hypothesis ordering involves two processes: assessing the hypotheses, and dividing the pool of hypotheses into three disjoint subsets. Hypotheses are assessed through the use of multiple-objective evaluation, a method of using several measures to estimate overall hypothesis goodness. Three objectives are used in the current implementation of OXGate: the Quick-Look, simplicity, and primitiveness. The Quick-Look measures the confirmation of a hypothesis against a random subset of the training data. Simplicity is a measure of the structural complexity of the hypothesis and provides an estimate of its plausibility. Primitiveness indicates whether the hypothesis is derived from an original primitive feature, or whether it is constructed from other hypotheses.

Several approaches to combining the measures for multiple-objective evaluation were considered in this chapter: non-domination, weighted combination, product combination, and the procedural approach. In each approach, the pool of hypotheses is evaluated over the measures, and the hypotheses compete in some manner against each other for membership in one of three subsets: the primary subset (most promising), the secondary subset (potentially useful), and the rejected subset (apparently useless). The approach used to partition the hypotheses into the subsets depends on the particular combination method in use. In each approach, primitiveness is given special treatment: a primitive hypothesis is never rejected. If designated for the rejected subset, it will be reassigned to the secondary subset.

The non-domination approach for hypothesis ordering is an extension of the use of non-dominated regions. By definition, a non-dominated hypothesis is one that is not surpassed in every evaluation dimension by any other hypothesis. To avoid overreliance on the Quick-Look confirmation measurement, which is only a rough estimate, tolerance bands are included for both the primary and secondary subsets. In this approach, the pool of hypotheses is partitioned into the "generally undominated" hypotheses (primary

subset), the "heavily dominated" hypotheses (rejected subset), and the remaining hypotheses (secondary subset).

The weighted and product combination approaches both involve producing a single measure of hypothesis quality, ordering the hypotheses by their quality, and partitioning the ordered list into the three subsets. The greatest difficulty incurred with these approaches is the development of an inexpensive partitioning mechanism robust enough to adapt to the particular mix of hypotheses.

The procedural approach orders the hypotheses according to the most important objective first, resolves ties by using the next most important objective, and continues in a similar fashion until the hypotheses are completely ordered or all objectives have been used. If the list is partitioned as in the weighted or product approaches, the application of the lesser objectives is generally wasted overhead. An alternative approach is to partition groups of hypotheses at a time: for each band of values of the most important objective, the hypotheses included in the band are partitioned into the three subsets according to the ordering provided by the lesser objectives. Proper assignment of hypotheses to the subsets involves a dynamically changing criteria for membership that is adjusted according to the value of the most important objective.

Of the four approaches to multiple-objective evaluation, the non-dominance and weighted combination methods provide the best empirical performance as will be shown in Chapter 5. The non-domination approach is the most intuitively appealing of the methods discussed, since it does not attempt to produce a single comparative term for each hypothesis based on limited information. Instead, it compares the hypotheses along each of the objectives and selects the ones with the best overall performance. The non-domination approach to hypothesis ordering is psychologically satisfying, functionally robust, and computationally economic, making it the method of choice for implementation in OXGate.

CHAPTER 5

EXPERIMENTS AND ANALYSIS: HYPOTHESIS ORDERING

This chapter presents several experiments demonstrating the use of hypothesis ordering in OXGate, and the analysis of several important design considerations. The chapter is divided into five major sections. In the first section, the experimental method and the domains used during the investigation are outlined. The second section baselines OXGate against the well-known inductive system ID3 to establish the basic representational and processing speed differences. Section 5.3 presents a preview of hypothesis ordering at its best, providing the context for the succeeding discussions. The system design is presented in Section 5.4, providing an overview of, and rationale for, several key design choices made during the evolution of OXGate. Following the system design, Section 5.5 presents the system analysis: the investigation of the robustness of the hypothesis ordering mechanism as it is applied across several domains. The main results and conclusions are summarized in Section 5.6.

5.1 Experimental Method

The experiments of this chapter are all aimed at evaluating the effectiveness of the hypothesis ordering mechanism. The hypothesis ordering component was tested by flooding OXGate with constructed hypotheses and measuring the ability of OXGate to manage the extra load. The constructed hypotheses were either created by hand, produced by special-purpose routines, or both. The specific method used for generating constructed hypotheses varied with the experiment and test domain.

The experiments presented here are intended to illustrate the points being made, rather than provide an exhaustive account of all possible permutations. Many experiments were run early in the evolution of OXGate that steered its development toward its current state. Several of the adjustable parameters were tuned during this phase to provide reasonably robust performance for concepts from the Boolean and Nominal domains (described below), and later were checked using concepts from all four test domains to ensure their continued acceptability. These parameters include: the minimum number of instances in the Quick-Look dataset (10), the maximum size of the Quick-Look dataset as a percentage of the available training data (10%), a set of five parameters describing the breakpoints in the histogrammatic partitioning approach, the relative weights used in weighted multiple-objective evaluation, the width of the tolerance bands for the non domination approach (readdressed in Section 5.5.1), and the minimum numbers of examples (40) and hypotheses (15) required before hypothesis ordering is enabled. This chapter highlights the most important and illustrative of the experiments accomplished after the initial phase of OXGate's development.

5.1.1 Measurements and displayed data

The figures of this chapter show various measurements made on the decision-tree created by the system under consideration: the abscissa is the number of examples used in training (developing the decision-tree) and the ordinate is the dependent variable. The three measured variables are the error-rate, the number of primitive hypotheses incorporated in the decision tree, and the amount of CPU time necessary to develop the decision-tree. Two computed variables, the statistical significance and the slow-down/speedup factor, are also used.

The error-rate is simply the complement of the predictive accuracy (100% - accuracy) as measured on a relevant set of test data. The number of primitives is an indication of

the cost-complexity of the decision-tree, providing a measure of the number of directly testable components assembled in the tree. In ID3, the number of primitives is calculated to reflect the number of primitives that would be needed in an equivalent binary decision-tree. This measurement was chosen over others (e.g., number of nodes or leaves) to directly relate the OXGate decision-trees to those of ID3. The amount of CPU time is a useful measure for comparisons within a figure, but not from one figure to another; different Sun® workstation configurations were used for batch experimentation depending upon machine availability. The CPU time is included as a comparative measure, not for strict theoretical purposes, but to provide a means of gauging the practicality of the OXGate approach in real-world applications.

All graphs show mean values measured over 10 runs, except where noted. Error bars, where included, represent the 95% confidence interval (two-sided) using the Student's t distribution. The statistical significance, where shown, is also derived assuming the Student's t distribution, and represents the likelihood of obtaining the observed results if the two error-rate curves being compared are assumed to represent the same distribution. The careful reader should note that, although tempting, it is incorrect to state that two curves appearing to be significantly different, i.e., having low values of significance, belong to different distributions. All that can be said is that the experiment supports the conjecture of one curve being better than the other, but does not prove it.

In Section 5.5.1 another statistical analysis tool, the *Friedman test*, is used to distinguish from among data distributions that appear similar, i.e., where pairwise comparisons of the distributions have high values of statistical significance. The Friedman test is a method of rank analysis, allowing a distribution to be selected as "best" based on its overall performance [Friedman, 1937].

Two additional points should be noted regarding the graphs. First, the confidence intervals are sometimes omitted for clarity of presentation. In those cases, the intervals

provide misleading information: the processing times for the larger training set sizes were so large that fewer than ten runs were made, resulting in relatively large confidence intervals. Second, some graphs exhibit a peculiar "bump" for hypothesis ordering on small training sample sizes. Since the Quick-Look examines only a fraction of the available training data, hypothesis ordering is disabled for training sets of forty or fewer examples to avoid oversensitivity to the particular examples drawn.

5.1.2 Domains and concepts

Table 5.1 provides a general overview of the concepts used in the experiments, taken from four domains. Two domains are artificial (Boolean and Nominal), and two (NetTalk

Table 5.1 Summary of Concepts Used

Concept		Description	Instance Space	Disjuncts[1]
Boolean (Appendix C)	3-term 3DNF	3 trinary conjuncts	12 binary attributes ⇒ 4096 instances	3, 13, 13 33%
	4-term 3*DNF	1 binary, 3 trinary conjuncts		4, 27, 27 50%
Nominal (Appendix D)	A	structured, medium complexity	4 nominal attributes 3 to 5 values/attribute ⇒ 240 instances	4, 7, 7 37%
	B	unstructured, higher complexity (than Nominal A)	5 nominal attributes 4 to 9 values per attribute ⇒ 1350 instances	8, 25, 24 28%
	C	unstructured, low complexity	4 nominal attributes 3 to 5 values per attribute ⇒ 240 instances	3, 3, 9 60%
NetTalk (Appendix E)	Silent	silence in center window position	7 nominal attributes 27 values per attribute ⇒ 10^{10} instances possible, 143,000 available in database	unknown ≈ 113 ≈ 109 14%
Breast Cancer (Appendix F)		nonrecurrence of breast cancer	9 nominal attributes 2 to 13 values per attribute ⇒ 91,000 instances possible, 286 available in database	unknown ≈ 74 ≈ 83 70%

[1]Disjuncts. the number of positive leaves in the decision-tree, an estimate of the concept complexity. Three values for the number of disjuncts are given for each concept. the first is from the target concept

and Breast Cancer) are complex real-world domains obtained from the Repository of Machine Learning Domains maintained by the University of California at Irvine. Complete descriptions of the domains, concepts, and data are located in Appendices C through F. Table 5.2 provides a mapping indicating which concepts are used in each experimental category of this chapter.

Table 5.2 Matrix of Experiments

Section	Experimental Dimension	Application Domain			
		Boolean	Nominal	NetTalk	Cancer
Baseline	Representation	3-term 3DNF	A, B, & C	√	√
	Processing speed	3-term 3DNF	B	√	
Preview	OXGate preview			√	
Design	MOE vs. SOE	3-term 3DNF	B	√	
	Confirmation measures		A		
	Combination methods	3-term 3DNF	B	√	
	Concept dispersion			√	
Analysis	Tolerance bands	4-term 3*DNF	A & B	√	
	Pruning	3-term 3DNF	B	√	√
	OXGate in action	4-term 3*DNF	B	√	√

5.2 Baselining OXGate

In this section, the basic inductive operation of OXGate is contrasted with the well-established system ID3. The simple version of ID3[2] used for comparison performs basic selective induction with no decision-tree pruning, uses the information-gain as its evaluation measure, and operates on single-concept learning tasks. The experiments are intended to determine whether OXGate has a representational advantage or disadvantage

description, the second is from OXGate (using primitives only), and the third is from ID3 (n-way splitting). The fourth term is the coverage. the percentage of instances in the database which are positive examples of the concept. The OXGate and ID3 values were obtained from decision-trees created using the full set of available examples, except for the decision-trees in the NetTalk domain. These were created using a training set of 2000 examples.

[2]This implementation of ID3 was written in CommonLisp by Raymond Mooney (©1988).

relative to ID3 as discussed in Appendix B, along with an assessment of the computational overhead incurred by the OXGate testbed environment. Performance is examined in two dimensions: (1) predictive accuracy as a function of choice of representation, and (2) processing speed. The corresponding conclusions are: (1) neither representation is uniformly preferable for simple selective induction, and (2) the overhead of OXGate's hypothesis maintenance and blackboard administration causes a substantial, yet acceptable, speed degradation.

5.2.1 Predictive accuracy baseline

The goal of these experiments is to determine on the basis of predictive accuracy whether the binary decision-tree representation of OXGate is superior or inferior to the n-ary decision-trees produced by ID3. First, the equivalence of the two approaches under controlled conditions is established, followed by several comparisons of normal operation. In these experiments, only the primitive hypotheses were used by OXGate and the hypothesis ordering mechanism was disabled.

In the first experiment, the artificial 3-term 3DNF Boolean concept was used to force ID3 to generate binary decision-trees. The hypotheses were organized on OXGate's blackboard so that they would be considered in the same order in which ID3 examined the attributes. Since both systems use the estimate of information-gain to choose the best hypothesis or attribute and select the first one found in case of a tie, the resultant decision-trees should be identical. They were identical as indicated in Figure 5.1.

The next experiment examines the effect of randomly ordering the placement of hypotheses on OXGate's blackboard. This should have the effect of producing different decision-trees than ID3. At those decision points where more than one hypothesis has the maximum information-gain, the chance ordering of hypotheses can result in a different decision being made by OXGate than by ID3. Since the ordering of the attributes

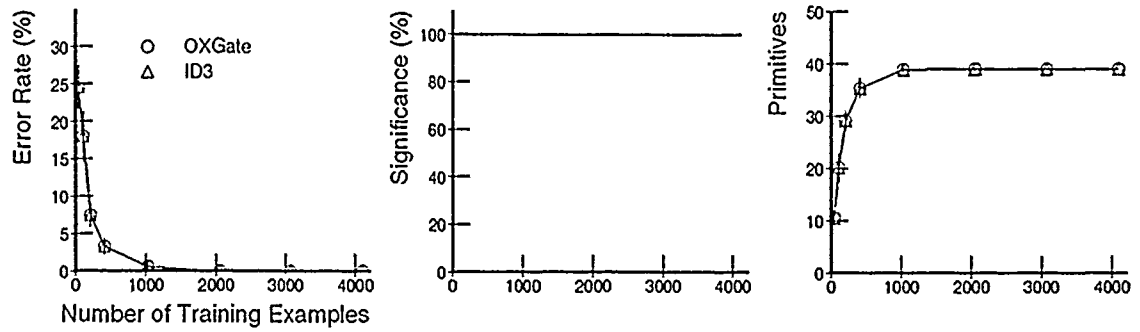


Figure 5.1 Representation Baseline: Boolean 3-Term 3DNF (Ordered). The hypotheses were organized on OXGate's blackboard in the same order in which their equivalents appear internally in ID3. For each set of training data, the decision-trees produced by ID3 and OXGate were identical.

in ID3 is essentially random with respect to the concepts, i.e., no attempt is made to optimize the ordering of the attributes, the additional randomness imposed in OXGate should have no significant impact. Figure 5.2 shows that for this particular set of ten experimental runs, OXGate performs slightly better than ID3, but not significantly so. The two approaches are essentially equal with respect to predictive accuracy and decision-tree complexity.

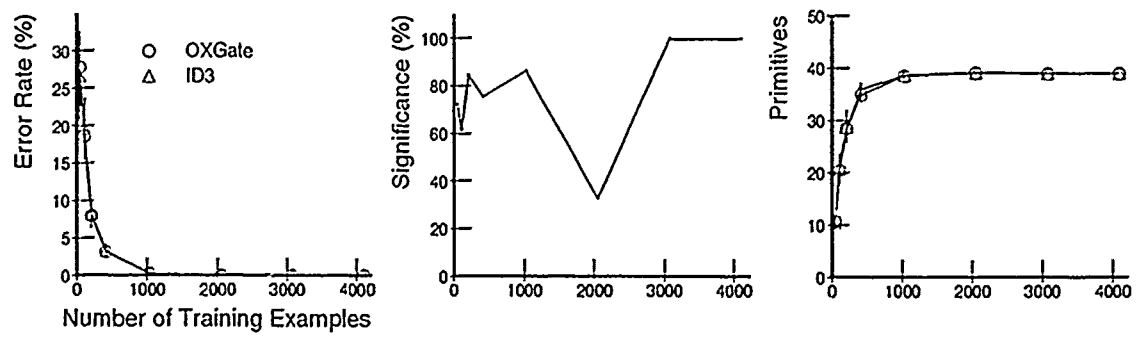


Figure 5.2 Representation Baseline: Boolean 3-Term 3DNF (Unordered). Unordered hypotheses in OXGate cause different decision-trees to be formed than with ID3. The significance curve suggests no preferred representation.

For certain classes of concepts, ID3 can be expected to perform better than OXGate. Nominal Concept A (Appendix D) is structured to match the n-ary splitting of ID3 to a moderate degree. With this concept, OXGate can be expected to fragment the decision-tree (OXGate uses attribute-value pairs for decision points rather than attributes), over-fitting the training data and providing poorer predictive accuracy. Figure 5.3 shows ID3 to perform better than OXGate on this concept.[3]

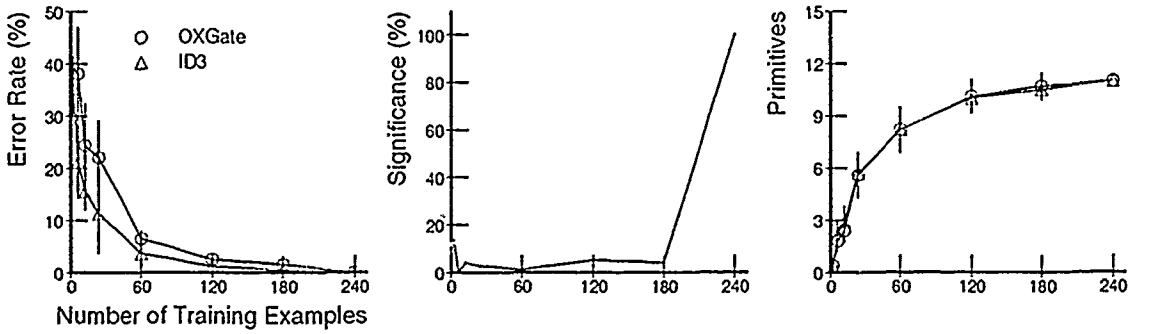


Figure 5.3 Representation Baseline: Nominal Concept A. ID3 is substantially better than OXGate on this concept because it is structured in an n-way decision form.

Nominal Concept B is more complex than Nominal Concept A and is intentionally structured to match the n-ary splitting of ID3 to a lesser degree. Figure 5.4 indicates no appreciable preference between the decision-trees produced by OXGate and ID3 in terms of the error-rate; however, OXGate produced more compact decision-trees.

Nominal Concept C is a very simple concept involving only one value for each of three attributes. The primitive hypothesis representation used by OXGate allows the three terms of concept C to be isolated and discovered with very small training set sizes. As shown in Figure 5.5, the correct decision-trees were assembled by OXGate much earlier than by ID3. OXGate's decision-trees were also more compact than those of ID3 in all

[3]The sudden increase in the significance at the right extreme of the graphs is simply the result of both systems converging to maximal performance when the set of test data is the same as the set of training data.

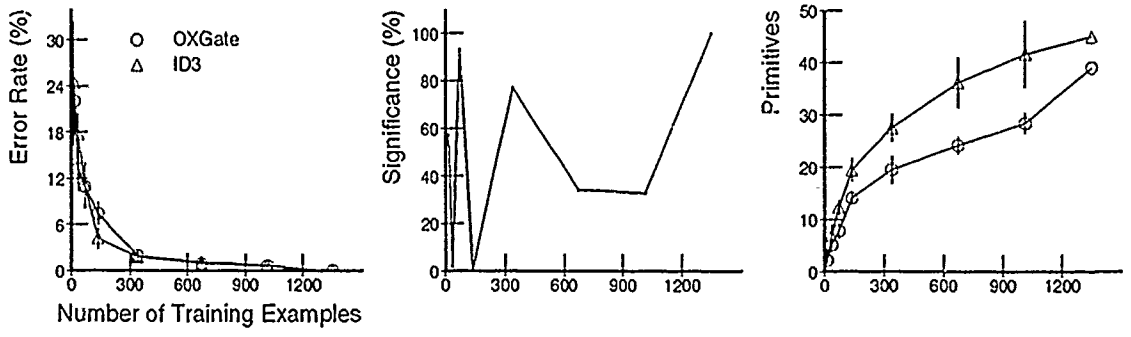


Figure 5.4 Representation Baseline: Nominal Concept B. ID3 is not significantly better than OXGate, because the concept is only partially structured in an n-way decision form.

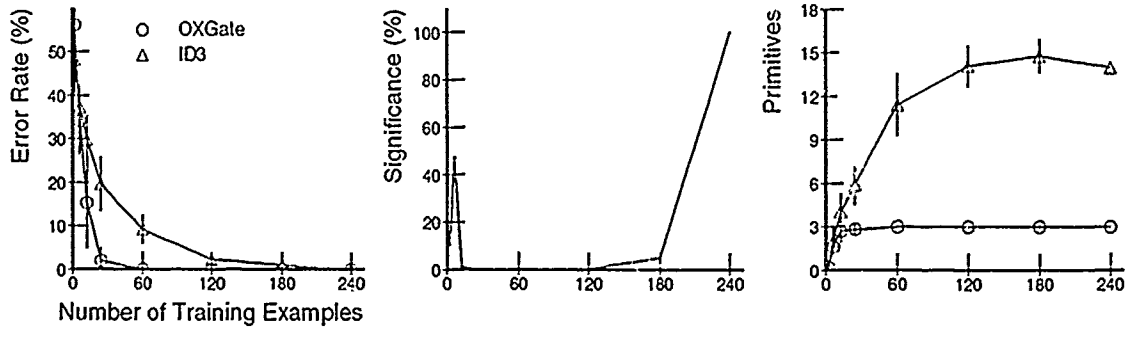


Figure 5.5 Representation Baseline: Nominal Concept C. The performance of OXGate is substantially better than that of ID3 for this concept.

instances, and optimally compact after sixty training instances. The poor performance of ID3 is a manifestation of the *replication problem* (Section 2.4.2)

Having established that artificial concepts can be constructed to prefer either the decision-tree representation used by ID3 or by OXGate, two real-world concepts are considered next. The first is the NetTalk Silent concept. In Figure 5.6, ID3 is shown to have the better decision-tree representation for the larger training set sizes. The performance of the two systems on the second real-world concept, the nonrecurrence of

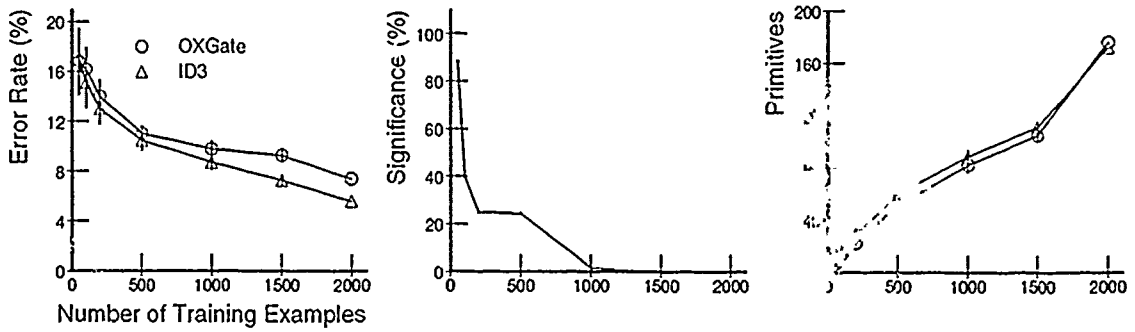


Figure 5.6 Representation Baseline: NetTalk Silent ...cept ID3 appears substantially better than OXGate for this NetTalk concept wh... ...ing the larger training set sizes.

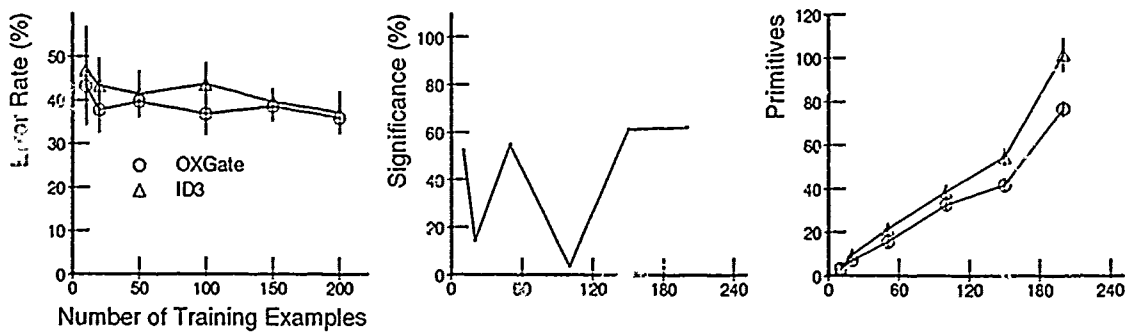


Figure 5.7 Representation Baseline: Breast Cancer Concept. OXGate appears generally better than ID3 for the breast cancer concept.

breast cancer, is shown in Figure 5.7. In this experiment, the binary decision-trees of OXGate appear to perform better than those of ID3.

The baseline predictive accuracy experiments do not indicate a general preference for either the OXGate or ID3 decision-tree repres...ntation. Rough overall parity is important: it supports the use of the binary decision-tree ...epresentation necessary in OXGate for the incorporation of constructed hypotheses. It also sets the stage for the succeeding experiments. Since no overall intrinsic representational advantage exists for either

approach, I can proceed to examine the effects of the addition of domain knowledge in the form of constructed hypotheses.

5.2.2 Benchmark speed comparisons

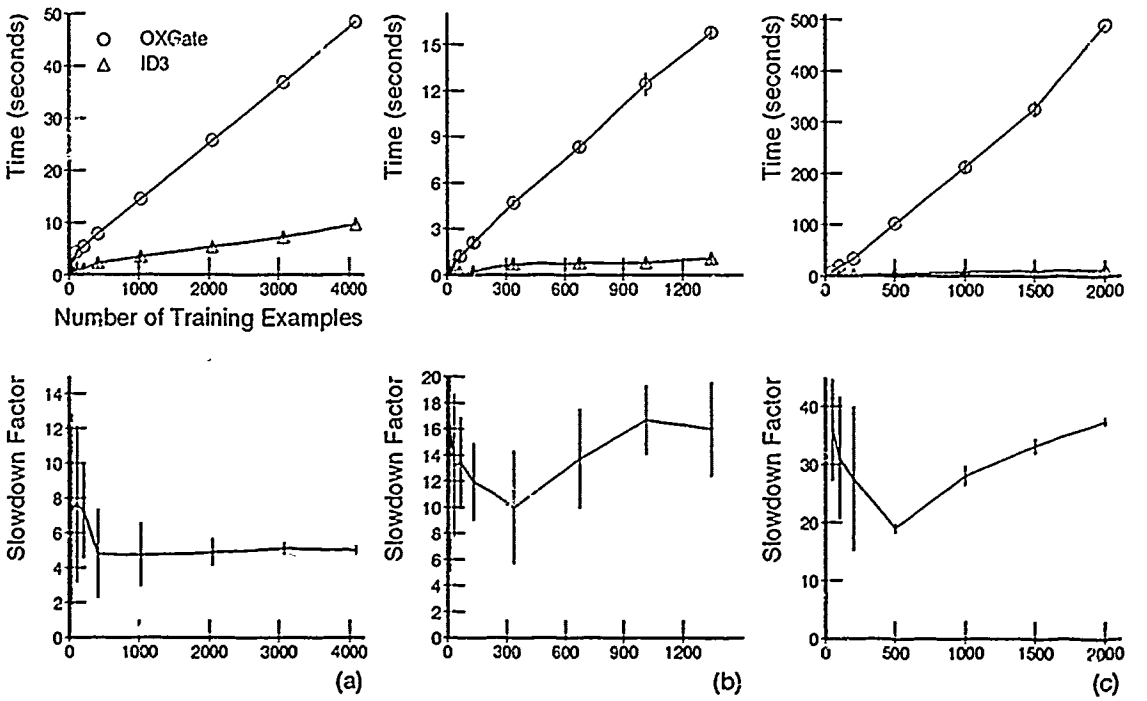


Figure 5.8 Benchmark Speed Comparisons. The overhead of blackboard maintenance and individual hypothesis consideration creates a performance degradation. Key: (a) Boolean 3-term 3D*F Concept, (b) Nominal Concept B, and (c) NetTalk Silent Concept.

The speed comparisons shown in Figure 5.8 contrast the processing times used by ID3 and OXGate for three of the experimental domains. As in the previous section, OXGate was operated in its fundamental mode, using only the primitive hypotheses available in the original instance description language. The comparisons indicate a substantial, but not excessive, processing speed differential between the baseline OXGate system and ID3.

For the domains tested, ranging from the simple to the complex, ID3 is between 5 and 40 times faster than OXGate. Several factors contribute to this difference. The most significant is that OXGate provides a development environment, while ID3 does not. OXGate is constructed on a blackboard substrate to provide high modularity and flexibility for continued experimentation and expansion [Jagannathan *et al.*, 1989]. It also contains a wealth of expansion "hooks" and test switches, unnecessary for the basic operation but desirable for growth and analysis. On the other hand, ID3 is a simple program that does exactly one thing: it finds the best overall primitive attributes and assembles a decision-tree from them. Once the development of OXGate has been completed and the critical components identified and optimized, the remaining portions (including the blackboard) can be recoded for direct communications and data-handling, effectively crystalizing the system into an efficient form.

Another significant factor affecting the processing speed of OXGate is the means of representing and testing hypotheses and instance data. Hypotheses are represented explicitly for ease of understanding, modification and construction. The instances are also coded explicitly, using property lists to relate the attributes to their values. In ID3 attributes are represented as positions in an ordered list, and the values are stored in the respective positions. The ID3 approach is more efficient than that of OXGate, but lacks the accessibility needed for the development of constructed hypotheses and the extensibility necessary for their incorporation.

Overall, the baseline overhead in OXGate is not exorbitant. It is a necessary cost of the flexibility required at this point in the development of OXGate. During normal operation, the addition of a large number of constructed hypotheses will impose a large amount of additional processing overhead, but as will be shown in this chapter, the incorporation of the hypothesis ordering mechanism removes a substantial portion of that cost.

5.3 A Preview of OXGate in Action

The purpose of this section is to provide a preview of the benefits of hypothesis ordering: to show its effectiveness and to provide a focal point for the discussion of the experiments in subsequent sections. This set of experiments uses the NetTalk Silent concept (see Appendix E). The results for the other domains are presented in Section 5.5. The effectiveness of the application of domain knowledge through constructed hypotheses is presented first, followed by a demonstration of the ability of hypothesis ordering to eliminate much of the computational overhead incurred with the addition of constructed hypotheses.

5.3.1 Experiments and results: NetTalk

As discussed in Section 5.1, an abundance of constructed hypotheses was generated in an attempt to saturate OXGate and gauge the utility of hypothesis ordering. Since the NetTalk Silent concept describes the concept "silence in the center window position" (character position 4, C4), I conceived of two forms of potentially useful domain knowledge, both involving the characters on either side of C4. The first form of knowledge was the conjecture that the characters adjacent to the center position might be important. This conjecture is strongly supported by the data of Lucassen and Mercer [1984], where the mutual information between the center window position and neighboring letters is shown to be greatest with the adjacent positions, and to decrease with the distance from the center. This knowledge was procedurally applied to exhaustively generate conjuncts of pairs of adjacent characters focused on position 4 (i.e., $((C3 = a)$ AND $(C4 = a))$, ..., $((C4 = z)$ AND $(C5 = z))$, where Cn indicates the *nth* window position), including the character "blank" to signal the beginning or end of words. This application yielded 1458 constructed hypotheses.

The second form of knowledge was more general, subsuming some of the first constructed terms and anticipating that when the adjacent characters were the same, one of them would be silent. This was used to create two additional constructed hypotheses, $(C3 = C4)$ and $(C4 = C5)$, for a total of 1460 constructed and 189 primitive hypotheses.[4]

This experiment consisted of testing the performance of OXGate under three conditions: using only the primitive hypotheses (*OXGate-Prim* in the figures), using the union of the primitive and constructed hypotheses, but with no hypothesis ordering *(OXGate-DK)*, and using hypothesis ordering on the union of primitive and constructed hypotheses *(OXGate-DK/NDH)*. The data of the next three figures reflect the averages of ten experimental runs,[5] except for the values of *OXGate-DK* on the larger training set sizes. Due to the large processing time requirements of *OXGate-DK*, only three samples were taken using 1000 training instances, and two samples for each of 1500 and 2000 training instances. The smaller sample sizes resulted in large, uninformative confidence intervals for *OXGate-DK*; therefore, the intervals have been omitted from the graphs for clarity. The smaller sample sizes also yielded abnormally large significance values where displayed for the 1000, 1500, and 2000 training instance results.

Figure 5.9 presents the improvement in predictive accuracy obtained through the inclusion of the hypotheses constructed from the domain knowledge. The use of the domain knowledge results in a marked and consistent improvement over the use of only primitive hypotheses. Examination of the *OXGate-DK* decision-trees shows that several of the constructed hypotheses were consistently incorporated early in the tree assembly, the most useful being $(C3 = C4)$. In addition, when compared against Figure 5.6, the

[4]Other obvious forms of knowledge were not used, such as omitting occurrences of the "blank" in C4 and other impossible structures (e.g., C5 is blank when both C4 and C6 are not), and using knowledge about legal letter combinations to filter out hypotheses such as $((C3 = q)$ AND $(C4 = x))$.

[5]Each datum was obtained by training OXGate on the indicated number of examples and then testing on an independent set of 6000 instances.

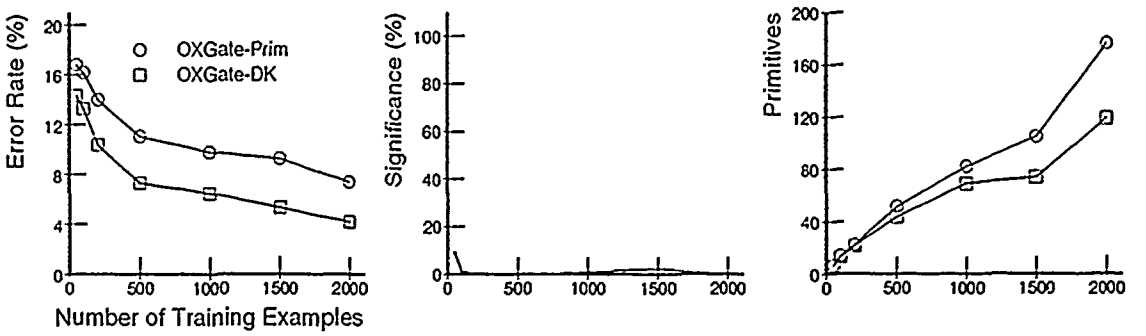


Figure 5.9 Application of Constructed Hypotheses. The inclusion of constructed hypotheses based on domain knowledge (DK) provides a pronounced improvement over the original (Prim) description language.

use of the domain knowledge more than compensates for the representational advantage ID3 has over OXGate for the NetTalk Silent concept.

The impact of hypothesis ordering is presented in Figure 5.10. The hypothesis ordering mechanism used for this experiment was the non-dominated hypothesis approach NDH (Section 4.2.4.1), with the retention of primitive hypotheses (Section 5.4.4.1) and tolerance bands untuned for this domain.[6] Although the NDH approach *(OXGate-DK/NDH)* produces decision-trees with roughly the same total number of internal tests as the non-ordered use of the pool of constructed hypotheses *(OXGate-DK)*, the decision-trees are less predictive than those of *OXGate-DK*. Comparison of the decision-trees produced by *OXGate-DK* and *OXGate-DK/NDH* reveals that although the decision-trees are very similar near the root nodes, the *OXGate-DK/NDH* uses fewer constructed hypotheses towards the leaves, presumably causing an overfitting of the training data and lack of generalization. This effect was expected as discussed in Section 4.2.1.1 and a possible solution is presented below.

[6]See Section 5.5.1 for a discussion of tolerance band tuning and its apparent insensitivity to domain.

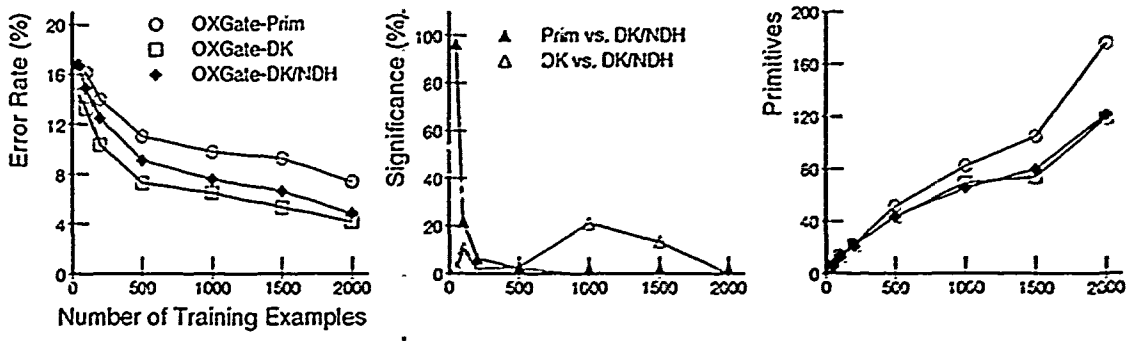


Figure 5.10 Effects of Hypothesis Ordering. With the NetTalk Silent concept, hypothesis ordering (DK/NDH) causes some loss of prediction over the non-ordered approach (DK), yet is better than using no knowledge. The significance curves contrast the use of primitive hypotheses alone (Prim) with the ordered use of the constructed hypotheses (DK/NDH), and the ordered (DK/NDH) with the non-ordered (DK) uses of constructed hypotheses.

Although the use of hypothesis ordering fails to capture all of the generalizations available from the pool of constructed hypotheses, it fulfills its promise of being a useful heuristic mechanism by the improvement it provides in processing speed. Figure 5.11 shows the amount of CPU time used by OXGate under the three experimental conditions. Not only did hypothesis ordering substantially speed up the processing as compared to the *OXGate-DK*, but by producing more compact decision-trees than baseline OXGate, it was able to complete the decision-tree assembly in the shortest time of the three approaches.

The results of this experiment suggest a way of recovering some of the potential generalizations the use of hypothesis ordering could not capture. According to Figures 5.10 and 5.11, disabling hypothesis ordering for training set sizes below 100 instances would result in better generalizations when using small sample sizes, at an acceptable computational cost. For the larger training sets, the hypothesis ordering mechanism should be enabled, resulting in a smooth error-rate curve consistently better than using no

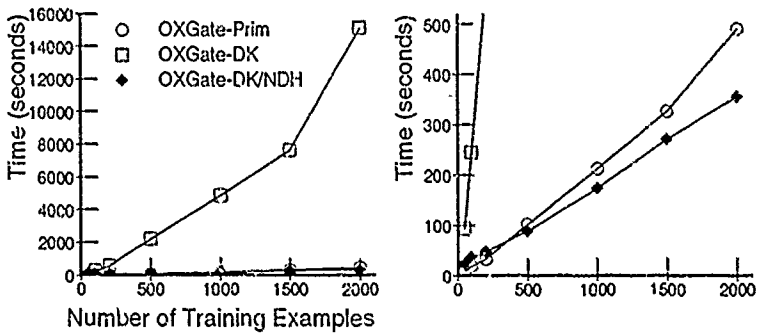


Figure 5.11 Effect of Hypothesis Ordering on Speed. The left-hand graph shows the average processing times of the three approaches. Hypothesis ordering (OXGate-DK/NDH) provides an impressive speed improvement over OXGate-DK, the non-ordered approach (up to 43 times faster). The right-hand graph is a closeup of the data. With the NetTalk Silent concept, hypothesis ordering was ever faster than using only the primitive hypotheses (OXGate-Prim), significant at $\alpha < 0.0035$.

constructed hypotheses. The creation of a mechanism to monitor and control the operation of hypothesis ordering is a possible subject for future research (see Chapter 7).

5.3.2 Preview summary

The use of hypothesis ordering is a practical heuristic for containing the effects of massive hypothesis generation. This section demonstrated that: (a) hypothesis ordering allows the inductive mechanism to discover and incorporate the more potent generalizations proposed by the hypothesis generator, (b) the use of hypothesis ordering can substantially reduce the processing overhead of having a large number of constructed hypotheses available in OXGate, and (c) the proposed approach to hypothesis ordering is viable in a complex, real-world domain.

5.4 System Design

This section presents several of the design choices made during the development of the hypothesis ordering mechanism in OXGate. These choices were made rationally as discussed in Chapter 4, and are supported by the empirical evidence in this section. The areas investigated are: comparison of the multiple-objective and single-objective evaluation approaches, selection of a well behaved confirmation measure, selection of a viable combination method for multiple-objective evaluation, and methods of compensating for concept dispersion. In this section, the data were collected over five experimental runs, except where noted. With only five runs, the error bars appear misleadingly large.

5.4.1 Multiple- versus Single-Objective Evaluation

The following six figures demonstrate the advantage of multiple-objective evaluation (MOE) over single-objective evaluation (SOE) when used for hypothesis ordering. The first three figures present the comparisons of MOE to SOE when using the weighted combination approach for hypothesis ordering, and the other three figures present the comparisons for the non-domination approach. Three concepts, random Boolean 3-term 3DNF concepts, Nominal Concept B, and the NetTalk Silent concept are used for each combination approach.

For each comparison, a large number of constructed hypotheses were generated procedurally to burden the hypothesis ordering mechanism. In the experiments with the Boolean 3-term 3DNF concepts, there were 24 primitive hypotheses ($(X1 = T)$, $(X1 = F)$, $(X2 = T)$,...) and 2280 constructed hypotheses. Twenty-four of the constructed hypotheses were the negations of the primitives. The identification of logical redundancies (e.g., noting that $(X1 = T)$ is equivalent to (NOT $(X1 = F)$)) was not a concern: the aim was simply to exercise the hypothesis ordering mechanism. Using the

union of the primitive hypotheses and their negations as the set of operands, 1128 binary conjuncts and 1128 binary disjuncts were formed out of every possible pair. Again, logical consistency was not a concern: the hypothesis ordering mechanism rejects hypotheses of the form $((X1 = T)$ AND $(X1 = F))$ since their confirmations are always zero. Likewise, the experiments with Nominal Concept B used 25 primitive and 2475 constructed hypotheses, generated in the same manner. The experiments with the NetTalk Silent concept used the 189 primitives and 1460 constructed hypotheses described in Section 5.3. In each figure, the behavior of OXGate without hypothesis ordering *(OXGate-Prim)* indicates the performance when using the primitive hypotheses only.

The experiments using only the primitives hypotheses are indicative of the behavior of SOE with the simplicity measure, S. Since the partitioning mechanism of hypothesis ordering is competitive, only the hypotheses with $S = 1$ would be placed into the primary subset (unless no hypothesis had $S = 1$, in which case the hypotheses with the next highest simplicity would be selected). Even using the non-domination approach with tolerance bands would not place more complex hypotheses in the primary subset: the next simplest hypothesis would have $S \leq 0.5$, implying that the primary tolerance band would have to extend over more than 50% of the evaluation space to include the hypothesis. For the description languages and concepts used in this section, the primary subset would consist only of hypotheses with $S = 1$, i.e., the complete set of primitive hypotheses.

The hypotheses in the secondary subset are tested by the hypothesis evaluation mechanism only if none of the hypotheses in the primary subset are acceptable (Chapter 4). When the original description language is adequate to completely describe the set of training examples, at least one of the primitive hypotheses will always be acceptable at each stage of decision-tree assembly. Under this condition, none of the hypotheses in the secondary subset are ever examined, implying that the use of primitives hypotheses only *(OXGate-Prim)* is an accurate emulation of SOE with the simplicity measurement.

In each of the domains tested in this section, Boolean, Nominal and NetTalk, the decision-trees created using only the primitive hypotheses always had perfect resubstitution accuracies. The original description languages were adequate to completely represent the training examples; consequently, no other hypotheses were necessary for decision-tree induction. Since no hypotheses from the secondary subset were needed, the use of primitive hypotheses only is an accurate emulation of SOE with simplicity. In the following figures, *OXGate-Prim* is understood to represent single-objective evaluation using the measure of simplicity.

Figure 5.12 contrasts the use of MOE and SOE for the weighted combination approach on the Boolean 3-term 3DNF concepts. The plots labeled *WMOE* indicate the use of the weighted combination approach using the Quick-Look confirmation measure and the simplicity.[7] To isolate the effect of using a single objective from the contribution of the partitioning approach associated with the weighted combination, the same algorithm was used for SOE as MOE. For SOE with the confirmation measure, the weights were modified to exclude the simplicity *(QL Only)*. The SOE with the simplicity measure was tested as described above *(OXGate-Prim)*. The use of MOE for hypothesis ordering shows a marked improvement over both SOE approaches in predictive accuracy as well as the conciseness of the resultant decision-trees.

Figures 5.13 and 5.14 contrast the use of MOE and SOE for the weighted combination approach for Nominal Concept B and the NetTalk Silent concept. The performance differentials are not as pronounced as in Figure 5.12, yet the use of weighted MOE is still superior to hypothesis ordering with the Quick-Look only, as well as hypothesis ordering with simplicity only.

[7] A satisfactory set of weights was determined in other experiments: $((5 \times Conf.) + (1 \times Simp.))$. Primitive hypotheses had special treatment as discussed in Section 4.2.5.

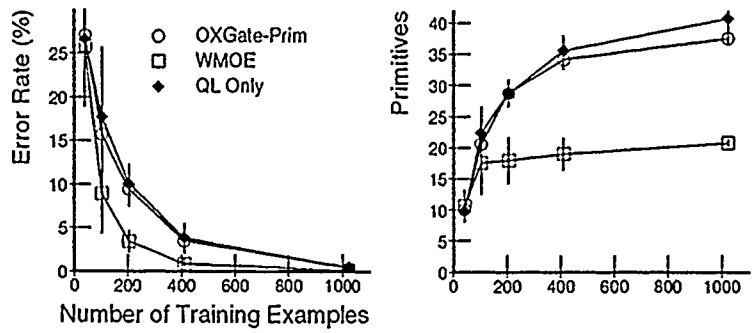


Figure 5.12 MOE vs. SOE: Weighted Combination, Boolean 3-term 3DNF. Comparison of the weighted combination approach to multiple-objective evaluation using the Quick-Look confirmation measure and simplicity (WMOE) versus single-objective evaluation with the Quick-Look confirmation measure alone (QL Only), and single-objective evaluation with simplicity alone (OXGate-Prim). This experiment shows a dramatic improvement in predictive accuracy and tree conciseness when using multiple-objective evaluation for hypothesis ordering.

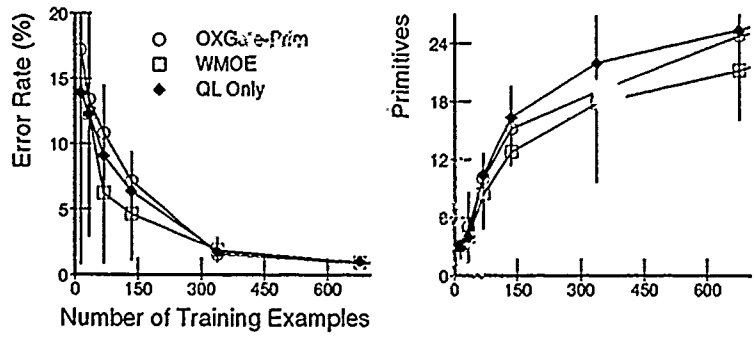


Figure 5.13 MOE vs. SOE: Weighted Combination, Nominal Concept B. Although not as dramatic as in Figure 5.12, the multiple objective approach (WMOE) is still noticeably superior to both of the single-objective evaluation approaches.

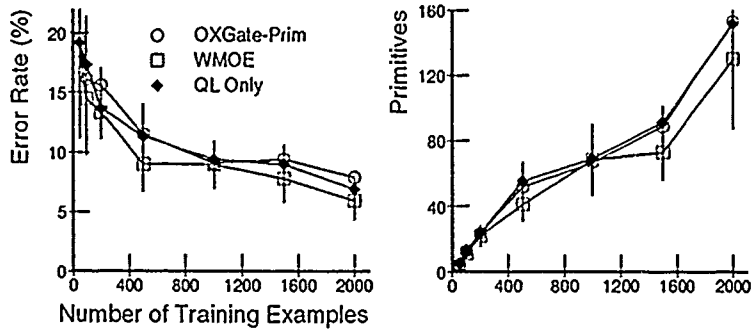


Figure 5.14 MOE vs. SOE: Weighted Combination, NetTalk Silent Concept. Although not as dramatic as in Figure 5.12, the multiple-objective approach (WMOE) is still noticeably superior to both of the single-objective evaluation approaches.

The experiments of Figures 5.15, 5.16, and 5.17 are identical to those just presented except that the non-domination approach to hypothesis ordering was used. Obtaining the *QL Only* data was accomplished by modifying the function that determines simplicity to always return the value "1." The values of the tolerance bands for partitioning with the non-domination approach were determined in other experiments (see Section 5.5.1). Except for poor performance with the Boolean concepts for low values of training set sizes[8] (Figure 5.15), the use the non-domination approach with multiple-objectives provided a small, but noticeable, improvement over the single-objective evaluation.

The combined results of the six experiments promote a general conclusion: the use of the multiple objectives, Quick-Look confirmation and simplicity, provides nearly consistently better performance than either the confirmation measure or simplicity alone. For the remainder of this thesis, hypothesis ordering will always incorporate multiple-objective evaluation.

[8]For 41 training examples, *QL Only* appears better than *NDII*, significant at $\alpha < 0.11$. For 102 training examples, *QL Only* appears better than *NDII*, significant at $\alpha < 0.20$.

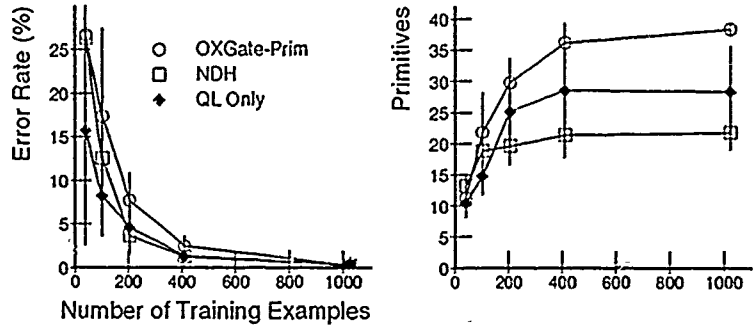


Figure 5.15 MOE vs. SOE: Non-Domination, Boolean 3-term 3DNF. Comparison of the non-domination approach to multiple-objective evaluation using the Quick-Look confirmation measure and simplicity (NDH) versus single-objective evaluation with the Quick-Look confirmation measure alone (QL Only), and single-objective evaluation with simplicity only (OXGate-Prim). This experiment shows marginal improvement in predictive accuracy for NDH over QL Only, but a substantial improvement in tree conciseness when using multiple-objective evaluation for hypothesis ordering. Note also the relative unpredictability (high variance) of the complexity of decision-trees for QL Only as indicated by the error bars.

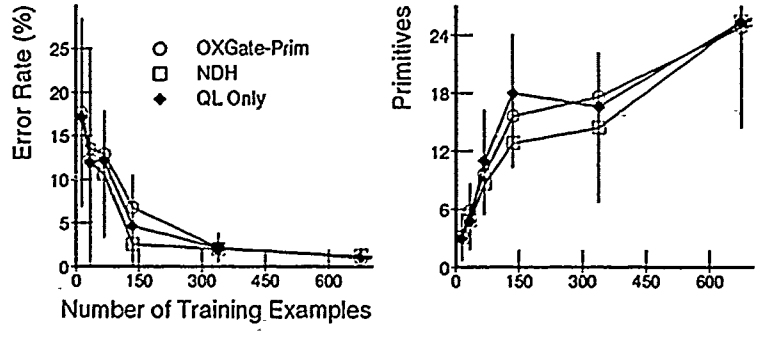


Figure 5.16 MOE vs. SOE: Non-Domination, Nominal Concept B. In this set of experiments, the multiple objective approach (NDH) appears slightly better than both of the single-objective evaluation approaches.

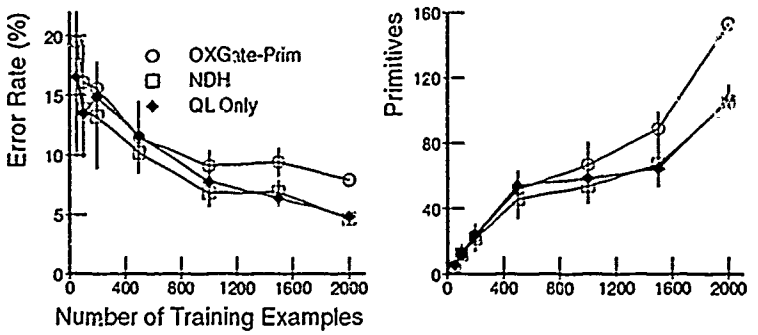


Figure 5.17 MOE vs. SOE: Non-Domination, NetTalk Silent Concept. In this set of experiments, the multiple objective approach (NDH) appears slightly better than both of the single-objective evaluation approaches.

5.4.2 Confirmation measures

Section 4.2.1.2 described the three measurements of confirmation considered in this thesis: information-gain (IG), average purity (AvePur), and positive purity (PosPur). Experiments accomplished early during the development of OXGate, but not presented in this thesis, demonstrated marked inconsistencies in decision-tree construction (unnecessarily complex decision-trees, poor predictive accuracies, and poor repeatability among experiments) when the confirmation measure used for hypothesis evaluation was not the same measure used for the Quick-Look evaluation of hypothesis ordering, regardless of the combination method used. This suggests that the confirmation method used in the Quick-Look should be the same as that of hypothesis evaluation. This section empirically addresses the question of which confirmation measure to use.

Since both hypothesis ordering and evaluation use the same confirmation measure, selection of the most useful measure can be addressed by examining their use for hypothesis evaluation, the more extensive of the assessments. In the experiments of Figures 5.18 and 5.19, hypothesis ordering was disabled to isolate the effect of the measurement choice for hypothesis evaluation. Nominal Concept A was used, providing 16

primitive hypotheses for the experiments of Figure 5.18 and 1008 additional constructed hypotheses (constructed in the manner described in 5.4.1) for the experiments of Figure 5.19. All data were acquired over ten experimental runs, except for those using the larger number of training examples and the set of constructed hypotheses *(OXGate-DK)* due to the large processing times without hypothesis ordering: five runs were used for the runs with 120 training examples, three runs for 180 examples, and one run for 240 examples (the full instance space).

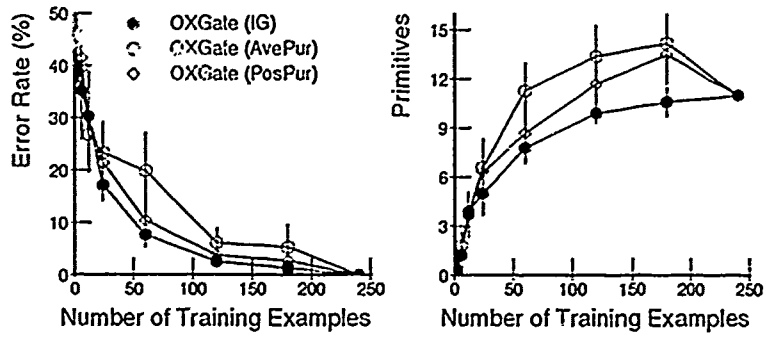


Figure 5.18 Information-Gain versus Purity Measures. Comparison of three confirmation measurement methods on the induction of Nominal Concept A. In each run, the sixteen primitive hypotheses were used as the pool of available hypotheses. Hypothesis ordering was disabled; the comparison is between the measurements used for hypothesis evaluation. Key: Information-Gain (IG), Average Purity (AvePur), Positive Purity (PosPur).

Mingers [1989] and Breiman *et al.* [1984] demonstrate that the predictive accuracy of induced decision-trees is relatively insensitive to the quality of the evaluation measure (including random choice). Instead, the complexity and understandability of the decision-trees are affected. Except for the single anomalous "bump" (significant only at $\alpha < 0.20$) for AvePur at 60 training examples in Figure 5.18, the results correspond with those of Mingers and Breiman *et al.* From the figures it is clear that the average purity (AvePur) confirmation measure is unstable: it produces decision-trees of widely varying complexities. The use of positive purity (PosPur) is more stable, but still exhibits

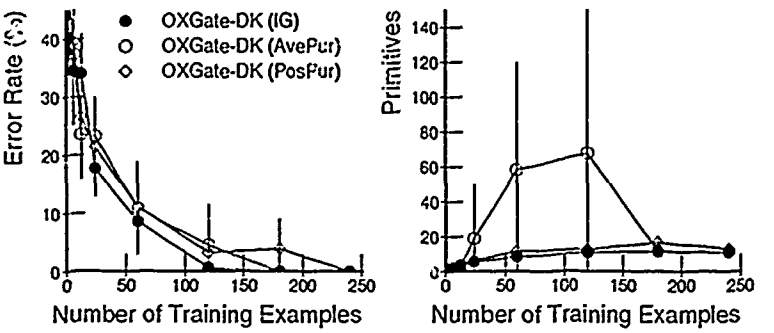


Figure 5.19 Information-Gain versus Purity Measures: With Knowledge. Comparison of three confirmation measures under the same conditions as in Figure 5.18, except the pool of hypotheses includes 1008 constructed hypotheses.

inferior behavior compared to the use of the information-gain (IG). Since Hartmann *et al.* [1982] asserted the efficacy of the information-gain measure for the development of decision-trees, it has been the measurement of choice in many inductive systems, including Quinlan's ID3. Similar experiments with OXGate in the other domains (not presented here) also support the use of information-gain for hypothesis evaluation and, therefore, hypothesis ordering. Except where noted, the information-gain was used for both hypothesis evaluation and ordering for the remaining experiments.

5.4.3 Combination methods

In this section, three of the four multiple-objective evaluation combination methods presented in Section 4.2 are compared, along with their associated partitioning schemes. The three appoaches are: weighted combination, non-domination, and product combination. The fourth method, procedural combination, was so ill-behaved in preliminary experiments that it was excluded from further consideration as an appropriate method of hypothesis ordering.

Figure 5.20 contrasts the performance of the three combination methods and the baseline on random Boolean 3-term 3DNF concepts. The experiments on the combination methods used the pool of 2304 hypotheses described in Section 5.4.1. The baseline *OXGate-Prim* operated on the 24 primitive hypotheses without hypothesis ordering. Data for the graphs were accumulated over five experimental runs, except for the *Product* data which were taken from ten runs. These additional runs make the confidence intervals of the *Product* data appear smaller by comparison than the corresponding intervals of the other approaches.

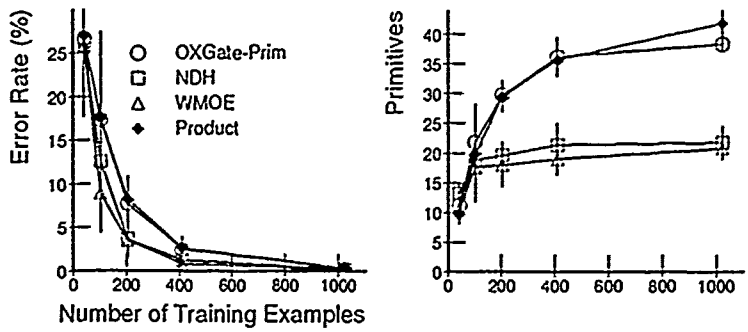


Figure 5.20 Combination Methods: Boolean 3-term 3DNF.

Although not radically worse than the performance of *NDH* (non-domination) or *WMOE* (weighted combination) in predictive accuracy, the product combination produced significantly more complex decision-trees. In fact, the performance of *Product* was very similar to that of *OXGate-Prim*. Analysis of the experiments revealed that the product combination approach had a very strong tendency to reject constructed hypotheses early in the decision-tree assembly, leaving only the primitive hypotheses for further processing. This behavior was expected for the information-gain confirmation measure as discussed in Chapter 4. The product combination approach was also tried with the average purity and positive purity confirmation measures, with similar results. In all cases, the decision-trees bore a strong resemblance to those created with only

primitive hypotheses. Clearly, this behavior is inappropriate for hypothesis ordering: the essence of opportunistic constructive induction is to develop and identify useful constructed hypotheses to improve the decision-trees. This glaring inadequacy eliminates the product combination approach from further consideration.

Figure 5.20 also illustrates the similarity between the weighted combination *(WMOE)* and non-domination *(NDH)* approaches to hypothesis ordering. Figures 5.21 and 5.22 provide additional evidence of the rough equivalency of the two approaches when used for Nominal Concept B and the NetTalk Silent concept. These results indicate there is no empirical preference for either approach based on predictive accuracy or decision-tree complexity. Figure 5.23 shows the relative processing speeds for the *NDH* and *WMOE* approaches on the three concepts. The results for the artificial domains show the non-domination approach incurred larger, less predictable processing times than the weighted approach. With the NetTalk Silent concept, the two approaches were nearly identical.

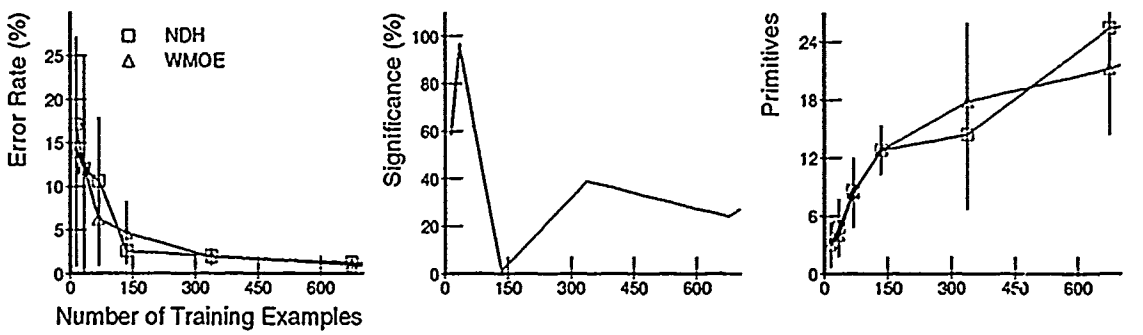


Figure 5.21 Combination Methods: Nominal Concept B.

Empirically, these experiments suggest that the weighted combination approach is the best of the four methods presented in Chapter 4. However, as argued by Yu [1985], the weighted approach has several drawbacks, including its lack of psychological palatability: when people make decisions, they seldom can impose relative weightings among objectives. The current implementation of the histogrammatic partitioning method

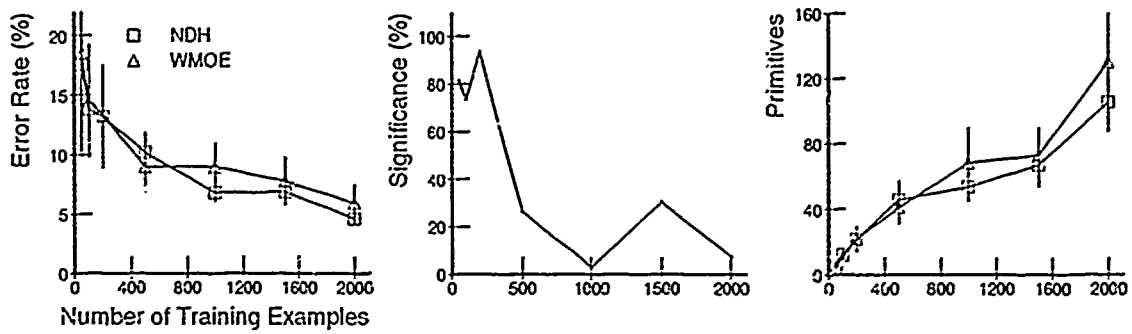


Figure 5.22 Combination Methods: NetTalk Silent Concept.

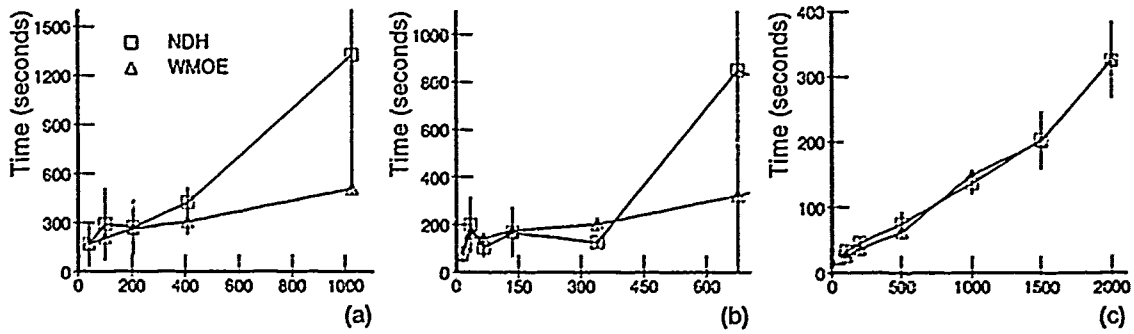


Figure 5.23 Combination Methods: Processing Time Comparison.
Key: (a) Boolean 3-term 3DNF, (b) Nominal Concept B, (c) NetTalk Silent Concept.

associated with the weighted combination approach also provides an extra degree of disquiet, with five additional parameters to adjust to obtain acceptable performance. The non-domination approach only requires that two parameters be adjusted, and as demonstrated in Section 5.5.1, this approach is fairly robust. Since the non-domination approach is more psychologically satisfying (Chapter 4), provides substantial processing time reductions (Section 5.5.3), and is the more uncommon of the approaches, the remainder of this chapter will explore the effectiveness of the non-domination approach to hypothesis ordering.

5.4.4 Compensating for concept dispersion

One of the potential problems with using the Quick-Look for hypothesis ordering was the premature rejection of hypotheses when learning highly disjunctive concepts. Several potential solutions were presented in Section 4.2.1.* and three of those are addressed here: retaining the primitive hypotheses at all times, recycling the pool of previously rejected hypotheses, and adjusting the density of the disjoint islands. Experiments with the NetTalk Silent concept show that the retention of the primitive hypotheses is a valuable addition to the capabilities of OXGate.

5.4.4.1 Retaining primitive hypotheses

The set of 1400 constructed hypotheses described in Section 5.3 was used in conjunction with the original set of 189 primitive hypotheses to provide the hypothesis ordering mechanism with an initially rich description language. The NetTalk Silent concept is a highly dispersed concept; therefore, the sample drawn for the Quick-Look has a high likelihood of failing to represent all the islands available in the training data. As a result, some hypotheses needed to assemble a good decision tree are rejected prematurely,

because they score poorly during the hypothesis ordering stages occurring early in the assembly process. The net effect is to remove so much of the vocabulary from the description language that OXGate is not able to completely describe the set of training examples. This manifests itself as shown in Figure 5.24 *(NDH wo/Primitives)* in unnaturally small decision-trees and very poor predictive accuracies, as well as poor resubstitution accuracies (not shown). In addition, the large confidence intervals indicate a large variation in the quality of the decision-trees, reflecting an oversensitivity to the makeup of the Quick-Look samples.

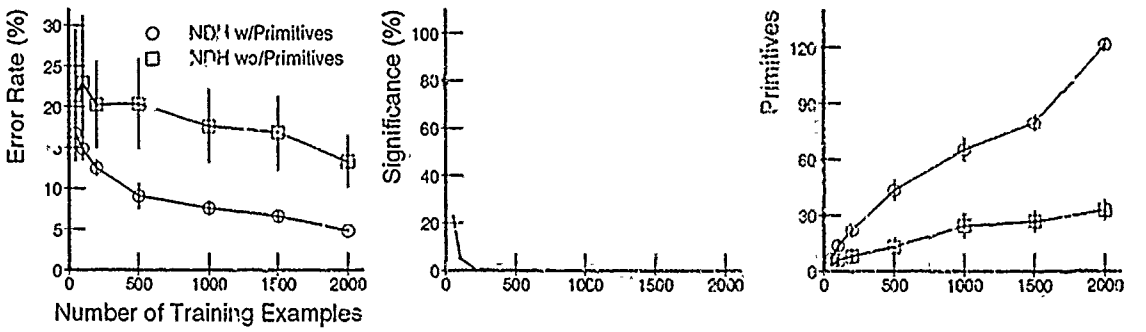


Figure 5.24 The Effect of Retaining the Primitive Hypotheses. Retention of the primitive hypotheses greatly improves both the accuracy and the stability of decision-trees created while using hypothesis ordering: NetTalk Silent Concept, 189 primitive and 1460 constructed hypotheses.

Forcing the retention of primitives does not necessarily compensate for unfavorable Quick-Look sampling and prevent the premature rejection of hypotheses, but it does maintain at least a minimal vocabulary. With this vocabulary, the training examples can be fully classified for maximal resubstitution accuracy, while at the same time, useful constructed hypotheses may still be identified and incorporated in the decision-tree, increasing the potential for improving the predictive accuracy. Figure 5.24 shows the dramatic improvement provided by the retention of the primitive hypotheses *(NDH w/Primitives)*: the predictive accuracy is greatly improved, and the small confidence intervals indicate

a relative insensitivity to the constitution of the Quick-Look sample sets. The increased complexity of the decision-trees does not indicate overfitting of the data, but rather the ability of the vocabulary to adequately represent the training instances.

The processing burden incurred by retaining the primitive hypotheses is difficult to measure. Since the number of primitive hypotheses (189) is much smaller than the number of constructed hypotheses initially available (1460), and the primitives are less expensive to test, the expected overhead of retaining the primitives should be minor. Figure 5.26 in Section 5.4.4.2 shows the relative processing speed performance of hypothesis ordering with the retention of primitives *(NDH w/Primitives)* and without (shown as *NDH wo/Prim/Rec*). While retaining the primitives appears to require about three times the processing time as not retaining them, the difference can be accounted for in the relative complexity of the decision-trees. Retaining the primitive hypotheses appears to be an inexpensive and effective method of augmenting the capabilities of hypothesis ordering.

5.4.4.2 Recycling rejected hypotheses

Another approach to compensating for the interaction of concept dispersion and the Quick-Look sampling is to reuse hypotheses previously rejected by hypothesis ordering. The pool of rejected hypotheses is maintained, and when the vocabulary available in the primary and secondary subsets is inadequate to classify the remaining training instances, the rejected hypotheses are added back into the vocabulary and the process of hypothesis ordering is reinvoked to continue to develop the decision-tree. In effect, recycling rejected hypotheses amounts to reexamining areas of hypothesis space that the hypothesis ordering mechanism had previously designated as "dead ends."

The resultant decision-trees are potentially better than when using the retention of primitives approach, because recycled constructed hypotheses may be incorporated in

the lower branches of the trees with a corresponding increase in predictive accuracy. This gain in capability is complicated by two side effects. The first is the damage done to the decision-tree before the need for recycling the rejected hypotheses is identified: marginally acceptable hypotheses in the primary and secondary subsets are incorporated into the tree as the description vocabulary is exhausted, producing trees with relatively poor internal decision points. The second side-effect is the large increase in processing time incurred by the reconsideration of a potentially large pool of previously rejected hypotheses.

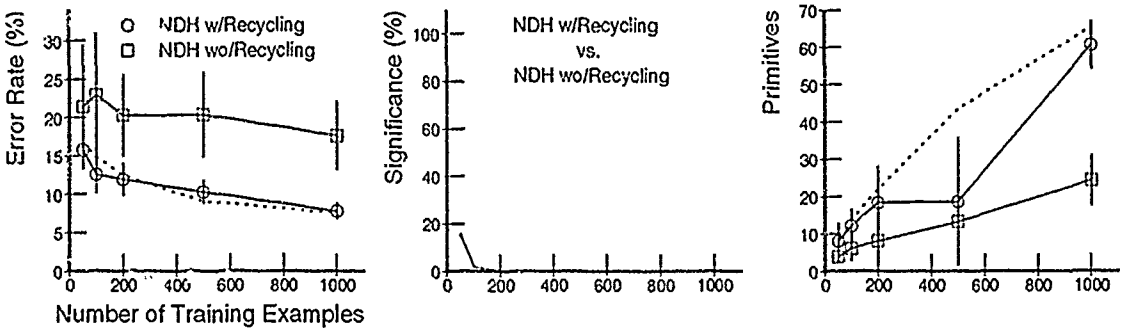


Figure 5.25 The Effect of Recycling Rejected Hypotheses. Reusing previously rejected hypotheses substantially improves the accuracy of hypothesis ordering. The dashed lines represent the behavior of NDH w/Primitives from Figure 5.24. (Note: The number of training examples used is terminated at 1000 due to the large processing time requirements.)

Figure 5.25 presents the results of recycling rejected hypotheses while learning the NetTalk Silent concept. The experiments were conducted as in the retention of primitives approach, with the exceptions that the number of training instances was limited to 1000 to reduce the processing burden, and only six runs were used. Primitives were not retained. The figure shows that recycling the rejected hypotheses *(NDH w/Recycling)* provides a large improvement in predictive accuracy over the non-recycled approach *(NDH wo/Recycling)*. The large difference in accuracy indicates that hypothesis ordering

rejected hypotheses necessary in the lower levels of the decision-tree: an expected behavior since hypothesis ordering performs a global evaluation of the hypotheses (also discussed in Section 5.4.4.1). This result indicates that hypothesis ordering alone is overly restrictive in this domain, most likely a consequence of high concept dispersion (Section 4.2.1.1).

Recycling the rejected hypotheses when the description language is exhausted can reintroduce terms necessary for induction of the lower-level nodes of the decision-tree, as it did in this experiment. However, it does not completely compensate for the damage done to the tree before identifying the need to recycle hypotheses. A comparison of *NDH w/Recycled* with the use of domain knowledge and no hypothesis ordering (*OXGate-DK* in Figure 5.9) shows a moderate degradation in predictive accuracy when using hypothesis ordering, even with recycled hypotheses.

The speed of processing was also affected as expected. Figure 5.26 shows that the recycling approach *(NDH w/Recycling)* substantially reduces much of the speed improvement available with hypothesis ordering *(NDH wo/Prim/Rec)* due to the reintroduction of a large number of rejected hypotheses. The gain in processing speed provided by hypothesis ordering over the non-ordered approach *(No NDH)* is only a factor of roughly five times when recycling the rejected hypotheses *(NDH w/Recycling)*. The tradeoff between the loss in predictive accuracy and this small gain in speed (less than one order of magnitude) makes this approach to hypothesis ordering only marginally desirable as a heuristic.

As an alternative to retaining the primitive hypotheses, recycling the rejected hypotheses provides both benefits and drawbacks. A comparison of the complexities of the resultant decision-trees shows that the recycling approach produced substantially more compact trees than did the retention of primitives approach: examination of the decision-trees revealed the incorporation of constructed hypotheses in the lower branches

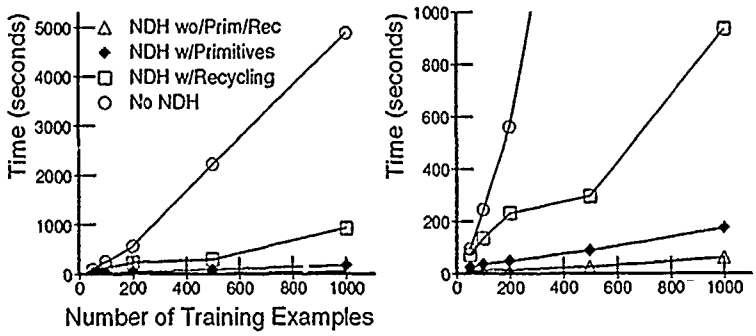


Figure 5.26 Processing Time Comparisons. The left graph shows the relative processing times of hypothesis ordering without the retention of primitive hypotheses or recycling of rejected hypotheses (NDH wo/Prim/Rec), with the retention of primitive hypotheses (NDH w/Primitives), with the recycling of rejected hypotheses (NDH w/Recycling), and using the full set of constructed and primitive hypotheses without ordering (No NDH). The right graph is a closeup of the faster processes. (Note: The number of training examples used is terminated at 1000 due to the large processing time requirements of No NDH.)

as expected. At the same time, recycling hypotheses provided nearly the same accuracy as that obtained by retaining the primitive hypotheses (*NDH w/Primitives* in Figure 5.24 and repeated in Figure 5.25 with a dashed line), an indication that the decision-trees were not smaller due to an exhausted description language. However, hypothesis ordering when recycling the rejected hypotheses takes considerably longer than when retaining the primitives. With 1000 training examples, the gain in processing speed provided by hypothesis ordering over the non-ordered approach *(No NDH)* is nearly a factor of 30 times when retaining the primitives *(NDH w/Primitives)*, but only slightly over 5 times when recycling the rejected hypotheses *(NDH w/Recycling)*. Since the objective of hypothesis ordering is to speed up processing while maintaining most of the predictive accuracy, recycling rejected hypotheses appears to be a much less useful addition to hypothesis ordering than retaining the primitive hypotheses. The fact that the recycling rejected hypotheses provided simpler decision-trees is a secondary consideration.

Another possible approach to hypothesis ordering combines recycling rejected hypotheses and retaining primitives. This approach was not tested, but the effects can be projected for the NetTalk Silent concept based on the experiments of this section and Section 5.4.4.1. An inspection of the decision-trees developed while retaining the primitive hypotheses showed that they all had maximal resubstitution accuracies, indicating that the primitive hypotheses provided a sufficiently rich description language. Therefore, no rejected hypotheses would have been recycled, resulting in a performance equal to that of retaining the primitive hypotheses only. This analysis also holds for the other concepts used in this thesis, including the Breast Cancer concept.

In other domains the combined approach may be useful, especially for complex concepts incorporating parity. If a concept requires parity constructions deep in the decision-tree (e.g., the hypothesis (A XOR B)), hypotheses of this type may be rejected early and would have to be recycled or recreated by the hypothesis generator. Primitive hypotheses alone would be insufficient to induce the parity construction.[9] Future research should investigate the benefits of a combination of recycling rejected hypotheses and retaining the primitives.

5.4.4.3 Adjusting island densities

A third approach to avoiding the effects of unfavorable Quick-Look sampling is the artificial adjustment of the proportion of positive to negative examples in the set of training data. When a target concept is highly disjunctive and the number of negative examples is much larger than the number of positive examples, there exists a strong likelihood that some of the smaller islands will be represented by only a few training examples. These small islands have a good chance of being completely overlooked by the Quick-Look

[9]See [Seshu, 1989] for one approach to overcoming the parity problem.

sampling. By increasing the number of examples per island, the probability of sampling from any particular island goes up; consequently, the small islands have a greater chance of being represented in the Qui-k-Look sample as discussed in Section 4.2.1.1.

The idea of adjusting the ratio of positive to negative examples to improve the accuracy of the resultant decision-tree was proposed in an unpublished manuscript by Spackman [1990]. In his approach, both the training and test data sets were adjusted equally: the training data remained representative of the test data. My approach in OXGate was to manipulate only the training data, incurring a change in the prior probabilities of the training data and making the training data less representative of the test data. Although this approach risks a degradation in the predictive accuracy, this adjustment had very little effect on the fundamental behavior of OXGate.

The two curves in each Error-Rate graph of this section represent the use of the original set of training examples (unbalanced) and the modified training set (balanced). The balanced set of data was created by replicating the positive examples in the original training set five times, increasing the ratio of positive to negative examples from 271:1729 (14% positive of 2000 examples) to 1626:1729 (48% positive of 3355 examples). The test data remained at 15% positive examples. I selected the nearly 50/50 ratio of positive to negative examples since this ratio was considered by Spackman to be the best. To contrast the behavior of the use of balanced and unbalanced data sets, the figures in this section all use the statistically expected number of unique training examples as the abscissae in the graphs. For example, a sample of 1678 instances from the balanced data set would be expected to contain 813 positive and 865 negative examples. Of these, the expected number of unique positive examples would be 135, yielding a total of 1000 unique examples. Therefore, half of the balanced data (1678 instances) corresponds to half of the original, unbalanced data (1000 instances) on the graphs.

Figure 5.27 shows a general insensitivity to balancing. Only ID3 appears to be consistently affected when using roughly 1000 unique training instances. The other two graphs, OXGate using only the 189 primitive hypotheses *(OXGate-Prim)*, and OXGate using the additional 1460 constructed hypotheses *(OXGate-DK)*, indicate that little benefit or cost is associated with using the balanced data when hypothesis ordering is disabled.

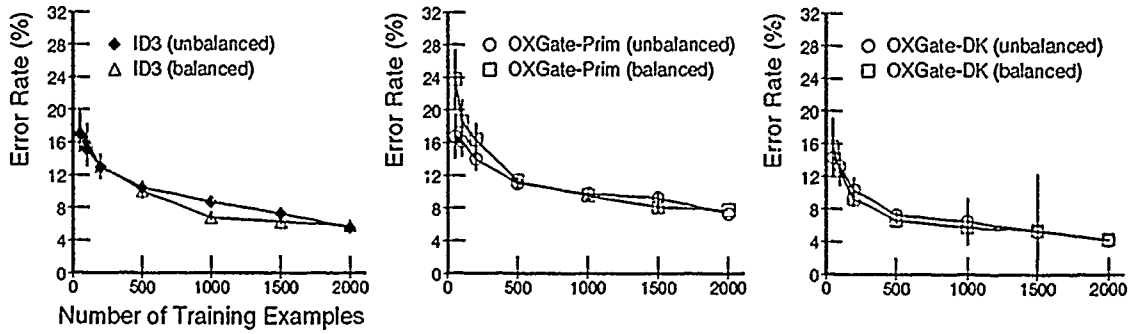


Figure 5.27 Balancing Positive and Negative Examples. The three graphs contrast the use of the original 2000 training examples (unbalanced = 14% positive examples) with the use of the modified training set of 3355 examples (balanced = roughly 50% positive examples). For the "balanced" curves, the Number of Training Examples shown on the abscissa represents the expected number of unique instances used for the decision tree construction; e.g., the set of 3355 actual examples corresponds to the set of 2000 original training instances.[10]

The next step in this experiment was to investigate the effect of balancing the data on hypothesis ordering. Figures 5.28 and 5.29 show the effect for hypothesis ordering without *(NDH-NP)* and with *(NDH-P)* the retention of primitives. The use of balanced data without the retention of primitive hypotheses *(NDH-NP)* provided some improvement, but the results were still poor and sensitive to the makeup of the Quick-Look samples. When using the balanced data in conjunction with the retention of primitives *(NDH-P)*, a small, but definite, improvement over the unbalanced case was evident. Figure 5.30

[10] As explained earlier, the number of experimental runs made with the larger training sets for *OXGate-DK* is fewer than 10 due to the large processing times, resulting in abnormally large confidence intervals.

shows the comparison of hypothesis ordering using the balanced data and retention of primitives *(NDH-P)* against the non-ordered use of the full set of constructed hypotheses on the original training data *(OXGate-DK)*. Not only is hypothesis ordering able to provide nearly identical predictive accuracies to those of *OXGate-DK* (and at one point better, significant at $\alpha < 0.25$), but does so up to 25 times as quickly for the NetTalk concept.

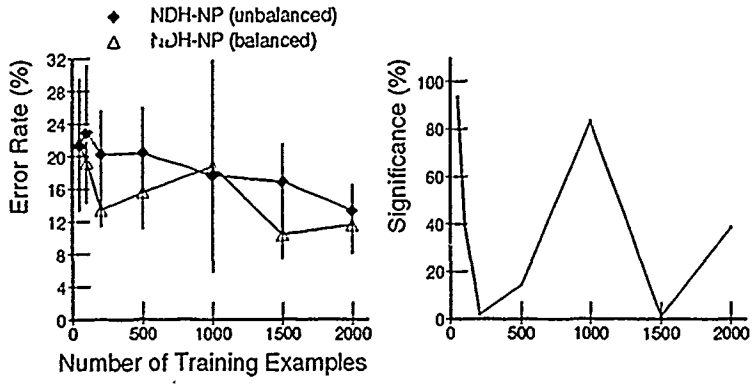


Figure 5.28 Using Balanced Data Without Retaining Primitives. Simply balancing the training data is not enough to stably improve hypothesis ordering as shown by the somewhat better, but still erratic, performance of hypothesis ordering without the retention of primitives.

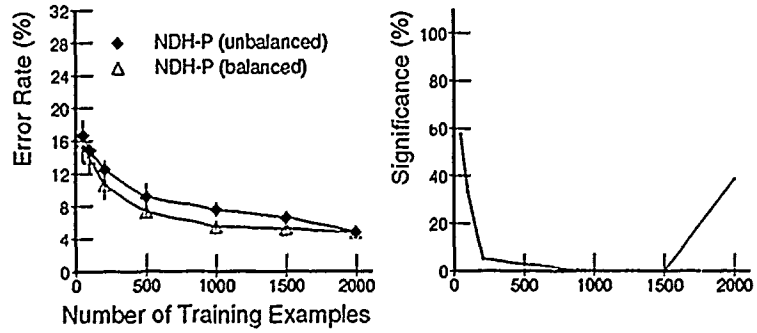


Figure 5.29 Using Balanced Data While Retaining Primitives. When hypothesis ordering retains the primitive hypotheses (NDH-P), balancing the training data provides a clear improvement.

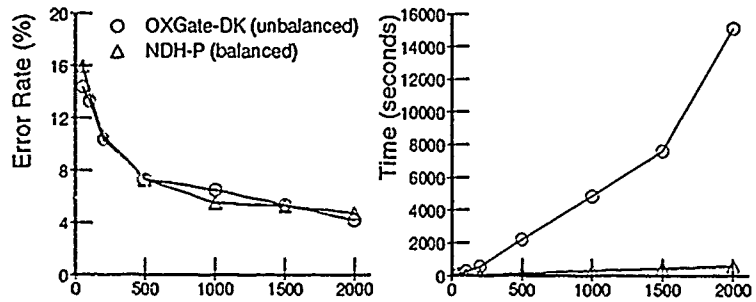


Figure 5.30 Overall Performance Improvement with Balanced Data. When using both the retention of primitive hypotheses and balanced data (NDH-P), hypothesis ordering has a nearly identical predictive accuracy as the non-ordered use of the full set of constructed hypotheses (OXGate-DK) on the original training data. NDH-P (balanced) also nets up to a 25-fold increase in processing speed over OXGate-DK (unbalanced). (Confidence intervals omitted for clarity.)

While the use of balanced training data appears to provide the last bit of capability needed by hypothesis ordering to match the predictive power of the non-ordered approach, it has not been incorporated as a feature of OXGate. The effects of skewing the prior probabilities of training data need further study. Without extensive analysis and empirical support, forcing the training data to be unrepresentative of the test data seems to be a risky approach.

5.4.5 System design summary

This section presented several of the system design choices made during the development of the hypothesis ordering mechanism in OXGate. The areas investigated were: comparison of the multiple-objective and single-objective evaluation approaches, selection of a well-behaved confirmation measure, selection of a viable combination method for multiple-objective evaluation, and methods of compensating for concept dispersion. The investigations of the first three areas yielded the following observations:

1. Multiple-objective evaluation, using the Quick-Look confirmation and simplicity measures, provided better overall performance than the Quick-Look confirmation measure alone (single-objective evaluation).

2. Of the three confirmation measures tested, the information-gain measurement proved superior for hypothesis evaluation, and by association, for ordering.

3. The weighted combination and non-domination approaches to hypothesis ordering proved roughly equivalent to each other and superior to the procedural and product combination forms. The weighted combination approach was faster than the use of non-domination in the artificial domains, and equal in the NetTalk domain.

Based on these observations, the considerations presented in Chapter 4, and the desire to explore the more novel approach, I incorporated the non-domination approach to multiple-objective evaluation as the standard method of hypothesis ordering in OXGate. Included with this decision is the use of the information-gain measure for both hypothesis evaluation and the Quick-Look confirmation measure.

After establishing the standard method of hypothesis ordering, several approaches to compensate for concept dispersion were addressed: retaining the primitive hypotheses, recycling the rejected hypotheses, and artificially balancing the training data. The retention of primitive hypotheses proved to be an inexpensive and effective method of compensating for the unfavorable interactions of Quick-Look sampling and concept dispersion. It has been incorporated as a permanent feature of OXGate. Recycling the rejected hypotheses was a computationally expensive approach with little advantage over the retention of primitives for the concepts used in this thesis, and was dropped from further consideration. The use of tolerance bands with the non-domination approach to hypothesis ordering provided reliable enough selectivity that the rejected hypotheses were not needed. The artificial balancing of the training data, while providing a clear

improvement for hypothesis ordering for the NetTalk Silent concept, is an area requiring further research.

5.5 System Analysis

This section investigates the robustness of the hypothesis ordering mechanism with the non-domination (NDH) approach of multiple-objective evaluation. The first investigation addresses the adjustment of the tolerance bands associated with NDH. This is followed by an assessment of post-pruning, a popular method of decision-tree refinement, to determine if it improves the performance of OXGate. Finally, the ability of NDH to provide substantial processing speed reduction while maintaining good decision-tree performance is demonstrated in all four experimental domains.

5.5.1 Tolerance bands

Section 4.2.5.1 described the rationale for the use of tolerance bands for the non-domination (NDH) approach to hypothesis ordering. Two parameters must be adjusted to attain satisfactory performance: the width of the primary band (Figure 4.3) and the width of the secondary band. For NDH to be a robust approach, a pair of values must be found to give consistently satisfactory performance across multiple domains. This section describes how a pair of acceptable values for the tolerance bands was determined, and investigates their sensitivity to domain.

The implementation of OXGate evolved while using the Nominal Concepts A and B along with two others not described. Various values of the primary and secondary tolerance bands were tested to select a set of values that would yield a good balance between predictive performance and processing time. Based on these results, I selected values of 0.10 and 0.15 for the primary and secondary tolerance bands of the Quick-Look

confirmation measures. The values are in terms of the information-gain measurement and represent absolute quantities.

Figure 5.31 shows the results of applying the same strategy to the random Boolean 4-term 3*DNF concepts (see Appendix C). The pool of hypotheses contained the same 24 primitive and 2280 constructed hypotheses described in Section 5.4.1. The results indicate the best choice is *NDH(10-25)*, 0.10 primary and 0.25 secondary tolerances, with *NDH(10-15)* as the second choice. The confidence intervals (error bars) are not shown, but for 205 training instances the comparison of the error-rates for *NDH(10-15)* and *NDH(10-25)* is significant at $\alpha < 0.23$, indicating that the two distributions are not necessarily as different as they appear on the graph. Even though *NDH(10-15)* does not appear optimal for this concept, it does appear to be adequate. It is an acceptable compromise between the Boolean and Nominal domain tolerance band tunings.

Another test that supports the acceptability of *NDH(10-15)* is the statistical rank analysis with two-way layout, or the *Friedman test* [Friedman, 1937]. Using the approach outlined in [Hettmansperger, 1984], each test cluster (group of experiments run at a particular value of training examples) is treated as a "judge" to rate the behavior of the seven "contestants." The mean values of the experimental runs (ten runs each for the

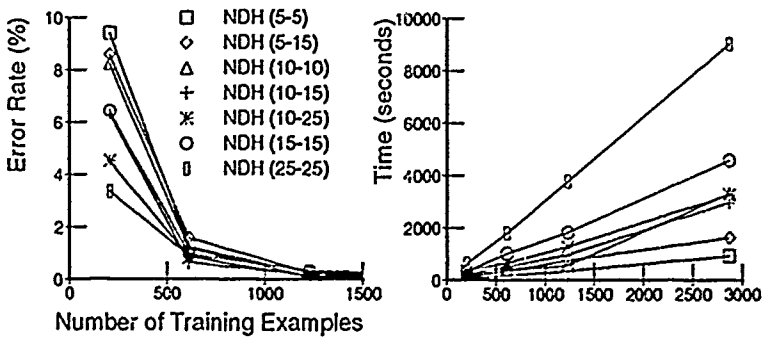


Figure 5.31 Tolerance Band Assessment: Boolean 4-term 3*DNF.

two smaller numbers of training examples, and five runs each for the larger ones) are used to rank the contestants. The judge's scores are summed and statistically analyzed to determine if there is a difference among the contestants.

Based on the error-rate data, with the rank of one associated with the lowest error-rate, *NDH(10-15)* is ranked a strong third place overall among the seven. The seven contestants are not significantly different from each other (different only at $\alpha > 0.25$). Based on the processing time data, with the lowest time assigned the rank of one, *NDH(10-15)* is ranked fourth overall. In this case, some contestants are significantly different than others: *NDH(5-5)* is significantly better than both *NDH(15-15)* and *NDH(25-25)* at $\alpha < 0.05$. When both the error-rate and time data sets are used (eight judges total), *NDH(10-15)* is ranked at a very strong second place, very close behind *NDH(5-5)*, which finished a weak fifth in error-rate, indicating that it does not provide a satisfactory time-versus-accuracy tradeoff. The seven contestants are significantly different from each other only at $\alpha > 0.70$. The conclusion to be drawn from this test is that, while no contestant is significantly better overall than any other, *NDH(10-15)* is the most reasonable choice for this concept since *NDH(5-5)* must be rejected for its poor error-rate behavior.

Figure 5.32 shows the results of the same set of experiments when applied to the NetTalk Silent concept. Apparently, the best sets of tolerance bands are *NDH(15-15)* and *NDH(25-25)*, but these proved to be poor performers with the Boolean concepts. The next best is *NDH(10-15)*, the set of tolerance values selected above. Comparing *NDH(10-15)* to *NDH(15-15)* and *NDH(25-25)* at 600 training instances yields similarities significant at $\alpha < 0.29$ and $\alpha < 0.32$, respectively. If these samples are assumed to belong to the same distribution, the odds of seeing the data as presented are approximately 3:7. This implies that it is quite possible for the apparent differences to be coincidental.

The Friedman test also demonstrates the seven sets of tolerance bands (contestants) to be statistically indistinguishable. For the test using only the error-rate data, only

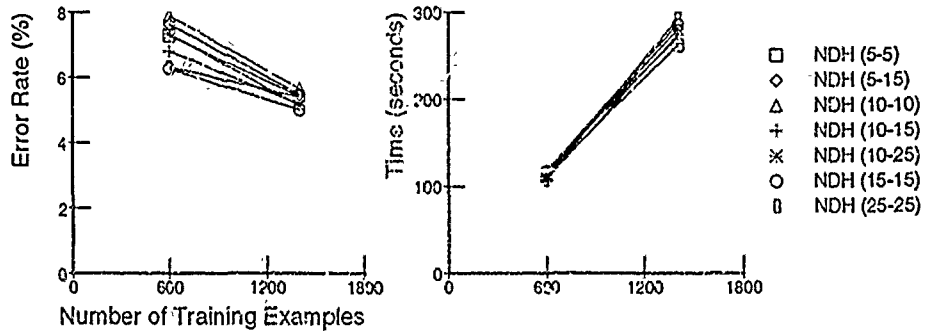


Figure 5.32 Tolerance Band Assessment: NetTalk Silent Concept.

NDH(10-10) could be considered different (worst), but at $\alpha > 0.10$. The test using the processing time data shows the contestants are different only at $\alpha > 0.45$. The test using the data from both error rate and processing time (four judges) shows the contestants are different only at $\alpha > 0.30$. Even though the contestants are not statistically different with any reasonable significance, *NDH(10-15)* is the contestant of choice based on the rankings. The tolerance band set *NDH(10-15)* is tied for second place with *NDH(25-25)* in the error-rate test, tied for first place with *NDH(5-5)* in the processing time test, and placed a strong first overall with *NDH(5-5)* ranked second. Therefore, *NDH(10-15)* appears to be the best choice for this NetTalk concept as well as the concepts tested in the Boolean and Nominal domains.

Tolerance band testing in the Breast Cancer domain is not presented because the domain was so ill-behaved that no tuning was effective (see Appendix F). No amount of tuning produced improvements; thus, no determination of the best bands was possible.

For the three domains investigated, the tolerance bands of 0.10 and 0.15 provide the most consistently acceptable performance in both predictive accuracy and processing time reduction. The three domains provided concepts with complexities ranging from simple (4 disjuncts) to very complex (>100 disjuncts), and coverages ranging from sparse (14%) to rich (50%), as described in Table 5.1 on page 97. Since the chosen tolerance

bands are appropriate for this breadth of concept diversity, they should be acceptable across all of the domains and concepts used for the remaining experiments. This tuning may also be sufficiently robust to be applied in other domains not considered in this thesis.

5.5.2 Post-Pruning

One of the most important criteria to judge the quality of a decision-tree is its predictive accuracy: the ability to correctly classify instances not in the training data. The predictive accuracy of a decision-tree is closely related to how well the decision-tree assembly algorithm is able to generalize from the training data; in other words, how well it can avoid overfitting to the training data. Proper generalizations yield high predictive accuracies. Good generalizations can be achieved through a sufficiently rich description language: the assumption is that, if terms at the correct level of generalization are in the existing vocabulary, they will be found and incorporated. This assumption is the bedrock on which opportunistic constructive induction is built. Other accepted methods of obtaining generalization often sacrifice a portion of the resubstitution accuracy to gain in predictive accuracy, accomplished in decision-tree classifiers by removing (*post-pruning*) or preventing the creation of (*pre-pruning*) certain branches of the tree [Breiman *et al.*, 1984]. This section examines the use of post-pruning as a potential addition to the capabilities of OXGate, as well as an alternative to hypothesis ordering.

The method of post-pruning used with OXGate is the same approach found in Assistant86. Each node in the decision-tree is tested to obtain an estimate of the *static* error at the node and the *dynamic* error, a weighted sum of the subtree error estimates given the decision split. If the static error is less than or equal to the dynamic error, then the subtrees are removed. [Cestrik *et al.*, 1987]

Figure 5.33 shows the effect of post-pruning on decision-trees created using primitive hypotheses only *(OXGate-Prim)* and those created using the union of primitive and constructed hypotheses *(OXGate-DK)*. The constructed hypotheses for the Boolean and Nominal concepts are those described in Section 5.4.1. The set of constructed hypotheses for the NetTalk Silent concept consists of the 1458 binary conjunctions described in Section 5.3, without the two generalizations $(C3 = C4)$ and $(C4 = C5)$.

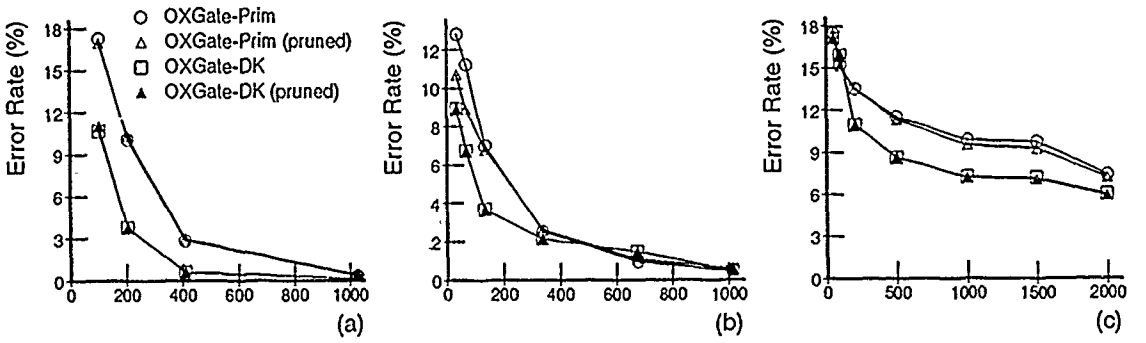


Figure 5.33 Effects of Pruning. The graphs show the effect of post-pruning decision-trees created with the original primitive hypotheses (OXGate-Prim) and constructed hypotheses (OXGate-DK). Hypothesis ordering was disabled. Key: (a) Boolean 3-Term 3DNF, (b) Nominal Concept B, (c) NetTalk Silent Concept.

As indicated in Figure 5.33, post-pruning provided a small improvement in the decision-trees for the Nominal and NetTalk concepts when using the primitive hypotheses only. Since the description language was highly limited, the correct level of generalization may have been unavailable, resulting in the overfit of the decision-trees to the training data. Post-pruning removes some of the overfit, generally providing better predictive accuracy. When using the constructed hypotheses to extend the description language, post-pruning had no effect on the decision-trees; presumably, sufficiently general terms were present in the vocabulary to avoid overfitting the training data in the first place.

Figure 5.34 presents the results of applying post-pruning to decision-trees created with the extended description language and hypothesis ordering enabled. Post-pruning had a

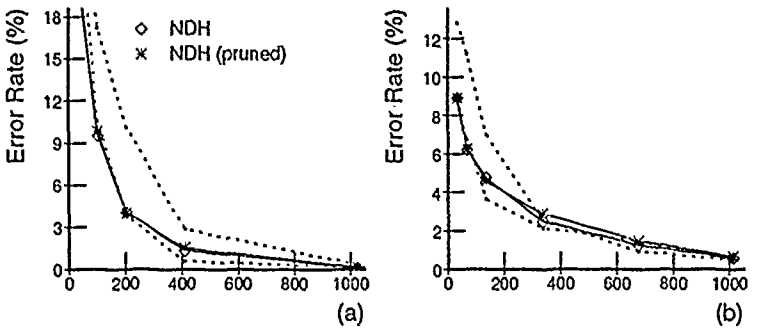


Figure 5.34 Pruning with NDH. Post-pruning had a negligible effect on the decision-trees developed using hypothesis ordering on the sets of constructed hypotheses. Key: (a) Boolean 3-Term 3DNF, (b) Nominal Concept B. The dashed curves represent the envelopes of the corresponding graphs of Figure 5.33, shown for comparison.

negligible effect. Where it does appear to have slightly altered the predictive accuracy, the effect was detrimental.

The experiments on the Breast Cancer concept (Appendix F) were performed differently than in the other domains. The set of constructed hypotheses for the Breast Cancer domain was obtained by developing a decision-tree to classify all 286 examples in the database. (The maximal resubstitution accuracy is 98% due to inconsistent training data.) The experiential learning program described in Section 6.1 accumulated all of the possible binary conjuncts appearing in the tree as well as the generalizations of those conjuncts. These 242 learned hypotheses were provided as the set of constructed hypotheses. In addition, a set of ranges for three of the attributes (age, tumor-size, and inv-nodes) was also provided to simulate the grouping behavior of Assistant.

When building a decision-tree, OXGate continues to refine it until the leaf nodes are as pure as a specified value. Normally, this value is set quite high, implying an expectation of relatively noise-free training data. The Breast Cancer domain appears

to be very noisy.[11] One approach to dealing with the noise is to relax the leaf purity requirements to avoid overfitting to the noisy training data.

Figure 5.35 shows the behavior of decision-trees at a variety of leaf purity requirements. Each experiment consisted of using a random set of 200 training examples to develop the decision-tree, and testing the tree on the remaining 86 examples. Each data point in the figure is the result of ten experimental runs.

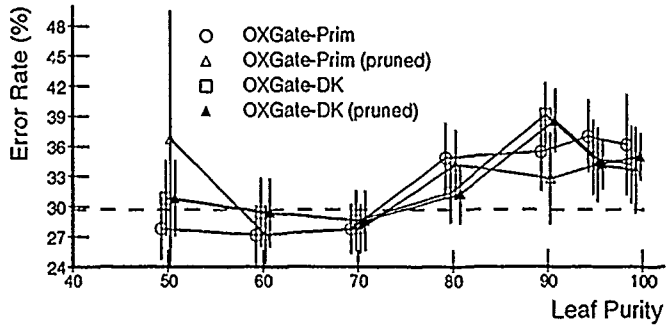


Figure 5.35 Pruning with the Breast Cancer Concept. In these experiments, decision-trees were developed using 200 examples of the Breast Cancer database, and tested on the remaining 86 examples. The graph shows the results for several leaf purities (50%, 60%, 70%, 80%, 90%, 95%, and 99%) used to determine when to stop splitting. The dashed line indicates the error-rate of chance. The four error bars clustered at each chosen leaf purity value are spread apart for clarity of presentation. The 95% confidence intervals are shown to demonstrate the erratic behavior with this domain.

Pruning of the *OXGate-Prim* decision-trees (developed with just the 51 primitive hypotheses) helped somewhat for the larger leaf purities, and severely degraded the performance with 50% leaf purity. The effect of pruning on the *OXGate-DK* decision-trees (developed with the union of the 51 primitive and 242 learned hypotheses, and hypothesis ordering disabled) was negligible. At 70% leaf purity, which is approximately the error-rate obtained by guessing, the four approaches yielded nearly identical results.

[11] See Appendix F for alternative explanations of its misbehavior.

The experiments across the four domains reveal that post-pruning does not contribute to the quality of the decision-trees produced by OXGate when a sufficiently enriched description language is used. In the low-noise domains (Boolean, Nominal, and NetTalk), no benefit was demonstrated by pruning beyond that provided by knowledge, regardless if hypothesis ordering was enabled. Pruning also did not consistently contribute to decision-tree accuracy in the Breast Cancer domain (a possibly noisy domain), especially when knowledge was used. As a supplement to hypothesis ordering, post-pruning is not needed. Without hypothesis ordering, the decision-tree assembly could become computationally very expensive and post-pruning would only add to the burden; therefore, post-pruning is not a suitable replacement for hypothesis ordering which serves to reduce the computational expense. Although post-pruning provides some benefit when using a sparse description language without knowledge, I have rejected it as an addition to OXGate: the goal of this research is to develop a mechanism for enriching the language, not to settle for a sparse original language.

5.5.3 OXGate in action

Section 5.3 presented a preview of the effectiveness of hypothesis ordering when applied to learning the NetTalk Silent concept. In this complex, real-world domain, hypothesis ordering proved to be a practical approach for reducing the computational burden imposed by a large number of constructed hypotheses. In this section, the use of hypothesis ordering is examined for learning concepts from the other three domains: Boolean, Nominal, and Breast Cancer. The results for the NetTalk domain are repeated here for completeness.

The first three figures of this section present the effects hypothesis ordering has on the predictive accuracy and tree conciseness available with constructed hypotheses.[12] The fourth figure shows the relative processing times in the four domains, and the fifth figure displays the processing speed improvements provided by hypothesis ordering over the unordered use of the constructed hypotheses.

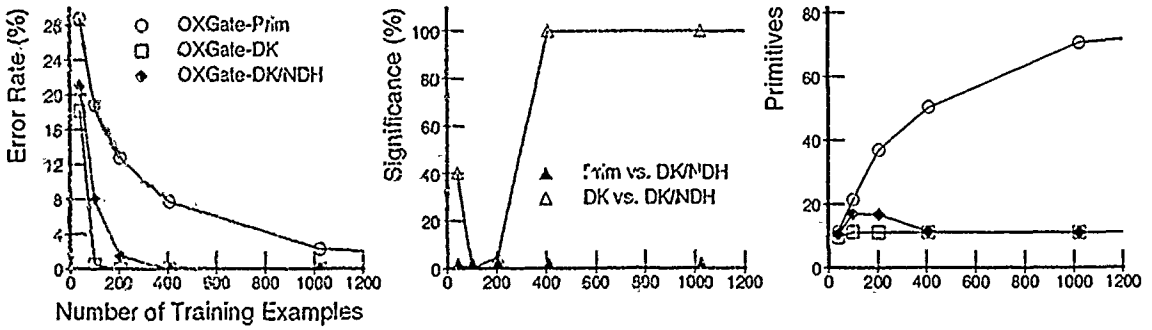


Figure 5.36 Effects of Hypothesis Ordering: Boolean 4-term 3*DNF. The use of constructed hypotheses (DK) exhibits a vast improvement in predictive accuracy and tree conciseness over the use of primitives alone (Prim), indicating that the knowledge applied was proper for this concept. Hypothesis ordering (DK/NDH) rescinded some of the accuracy improvements for the smaller sizes of training data.

Figure 5.36 illustrates the benefits of the application of proper knowledge to hypothesis generation. Trinary conjunctions (e.g., $((X1 = T)$ AND $(X10 = F)$ AND $(X3 = T))$) were added to the pool of constructed hypotheses, enabling very quick convergence to the actual concept. With the smaller sets of training examples, hypothesis ordering proved fairly detrimental to the predictive accuracy. This behavior was predicted in Section 4.2.1.1: the training examples represent a small portion of the instance space, and the Quick-Look is a fraction of the training data. One possible solution could be to

[12]The Boolean 4-term 3*DNF experiments used 264 binary and 1760 trinary conjunctions as the set of constructed hypotheses (Appendix C). The Nominal Concept B experiments used the 2475 constructed hypotheses described in Section 5.4. The Breast Cancer concept experiments used the 242 constructed hypotheses described in Section 5.5.2. The NetTalk Silent concept experiments used the 1460 constructed hypotheses described in Section 5.3.

increase the size of the Quick-Look sample, with a corresponding increase in processing time. A variant of this approach is to increase the sample size only for small or highly dispersed training sets (proposed in Chapter 7). Another approach is to disable hypothesis ordering for relatively small training set sizes. Figure 5.41(a) shows that for training set sizes less than 200, hypothesis ordering provides speedups of less than 6. Since predictive accuracy is being sacrificed for only nominal increases in processing efficiency, the disabling of hypothesis ordering at these lower training set sizes may be appropriate. Investigation of the means to assess dynamically the performance of the hypothesis ordering and disable the mechanism is also proposed in Chapter 7.

In the Nominal and Breast Cancer domains (Figures 5.37 and 5.38), the knowledge used to generate the constructed hypotheses does not appear to be appropriate: the gains in predictive accuracy and conciseness are marginal or nonexistent. The use of hypothesis ordering neither improved nor worsened the performance relative to the unordered approach, yet as shown in Figure 5.40 it did remove a substantial amount of the processing overhead. The conclusion drawn from these experiments is that hypotheses generated from inappropriate knowledge are quickly removed from OXGate. Hence, hypothesis ordering is effective as a filter between the hypothesis generator and evaluator, preventing poorly applied knowledge from inundating the evaluator with useless hypotheses.

Figures 5.39 and 5.41 demonstrate the balance in performance tradeoffs hypothesis ordering was intended to achieve. With NetTalk, a complex, real-world domain, the decision-trees developed using hypothesis ordering were not as accurate as those produced with the full set of constructed hypotheses: this was expected (Section 4.2.1.1). Yet, this small loss in accuracy was incurred while producing a substantial increase in processing speed, an acceptable tradeoff. Moreover, not only are the resultant decision-trees substantially more accurate and concise than those developed from only the primitive

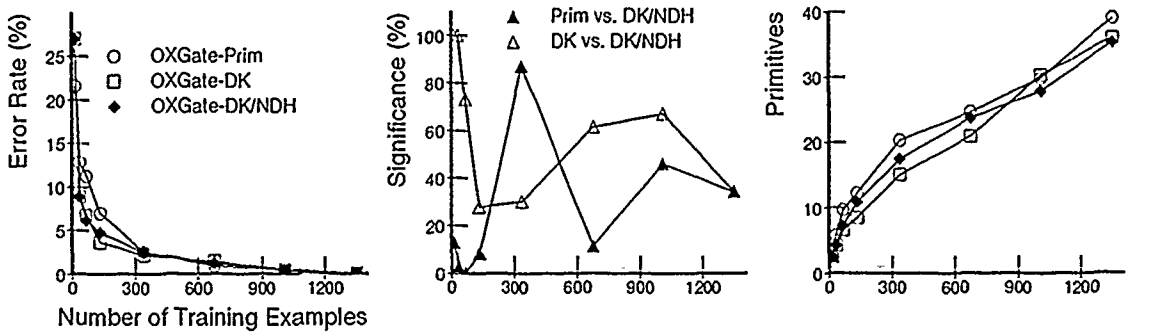


Figure 5.37 Effects of Hypothesis Ordering: Nominal Concept B. The use of constructed hypotheses (DK) exhibits a small, albeit inconsistent, improvement in accuracy and conciseness over the use of primitive hypotheses alone (Prim). Hypothesis ordering (DK/NDH) appears to have little detrimental effect on the predictive accuracy.

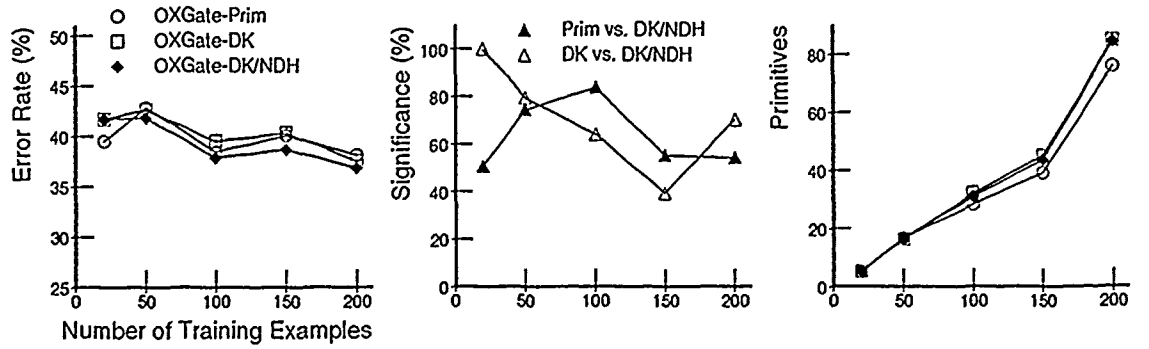


Figure 5.38 Effects of Hypothesis Ordering: Breast Cancer Concept. The significance curves indicate very little support for concluding that one approach is better than another.

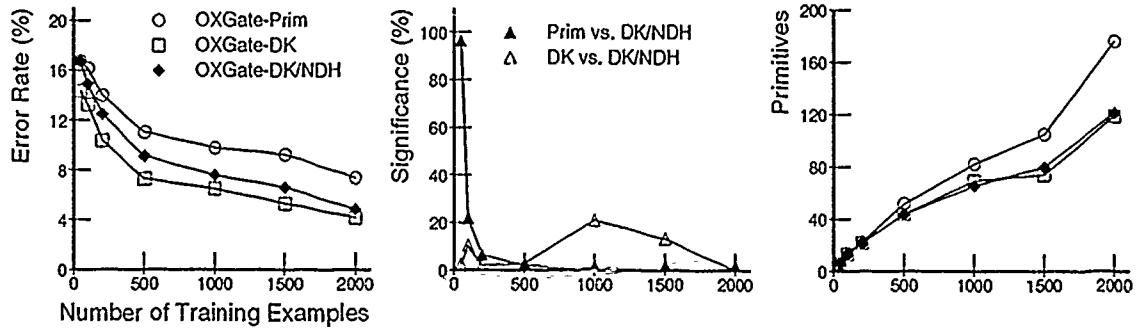


Figure 5.39 Effects of Hypothesis Ordering: NetTalk Silent Concept. Repeat of Figure 5.10, included for completeness. The use of hypothesis ordering (DK/NDH) causes some loss of predictive accuracy over the non-ordered approach (DK), yet is substantially better than using no constructed hypotheses at all (Prim).

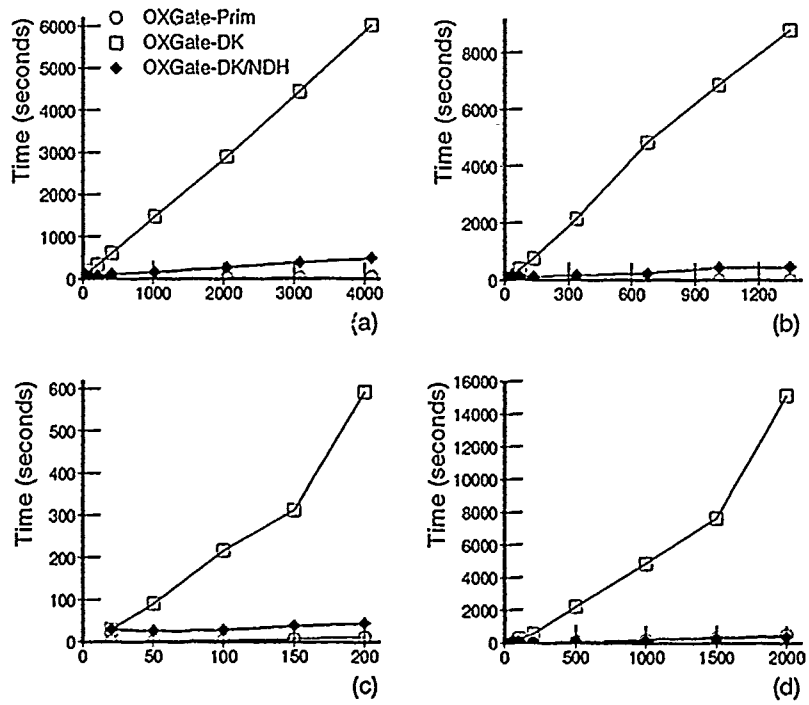


Figure 5.40 Processing Times. Key: (a) Boolean 4-term 3*DNF, (b) Nominal Concept B, (c) Breast Cancer Concept, (d) NetTalk Silent Concept.

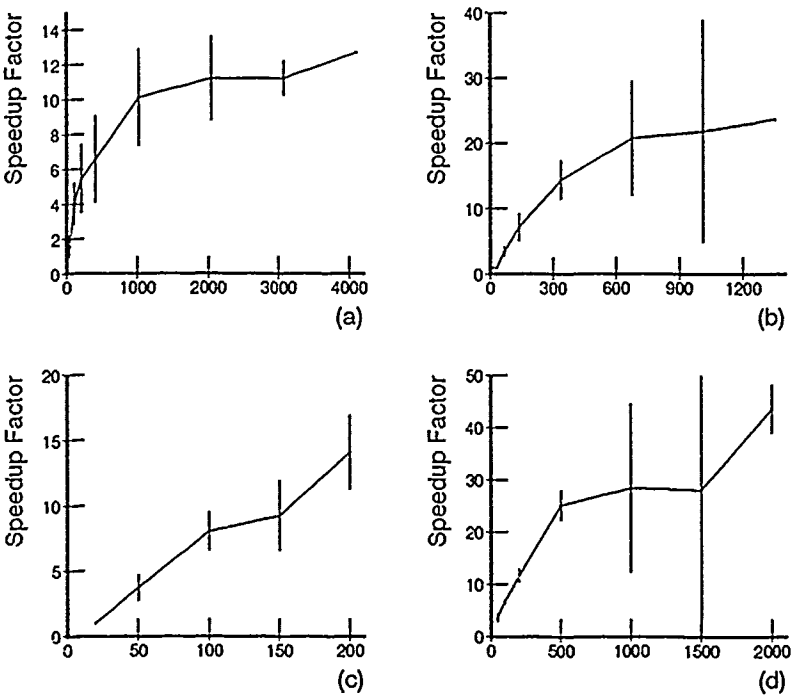


Figure 5.41 Speedup Factors. Comparison of the processing speed improvements provided by hypothesis ordering (OXGate-DK/NDH in Figures 5.36 through 5.40) over the unordered use of the constructed hypotheses (OXGate-DK). Key: (a) Boolean 4-term 3*DNF, (b) Nominal Concept B, (c) Breast Cancer Concept, (d) NetTalk Silent Concept. All data were collected over 10 runs except as noted: (a) used 5 runs with 2048 training examples, 3 runs with 3072, 1 run with 4096; (b) used 5 runs at 675, 3 runs at 1012, 1 run at 1350; (d) used 3 runs at 1000, 2 runs at 1500, 2 runs at 2000. Where fewer than 10 runs were used, the error bars appear abnormally large relative to the rest.

hypotheses, but the use of hypothesis ordering was even faster than using no constructed hypotheses, as was shown in Figure 5.11.

These four sets of experiments demonstrate two important characteristics of hypothesis ordering and its use in OXGate:

1. The hypothesis ordering mechanism is quickly able to identify hypotheses constructed from properly applicable knowledge and focus the evaluator on them.

2. The hypothesis ordering mechanism is quickly able to identify hypotheses constructed from inappropriate knowledge and eliminate them.

5.5.4 System analysis summary

This section analyzed the performance of the non-domination approach to hypothesis ordering in three areas of interest. The first area addressed the robustness of the tolerance band settings. The tolerance band values selected during early OXGate development provided an acceptable balance between processing speed gains and predictive accuracy for the Nominal concepts. Experiments in the Boolean and NetTalk domains demonstrated these settings to provide a desirable balance between accuracy and speed, confirming the robustness of the initial settings and the use of tolerance bands.

The second area of interest examined the utility of post-pruning the resultant decision-trees with the aim of determining if post-pruning would be a useful addition to or replacement for hypothesis ordering. The experiments in all four domains revealed that post-pruning did not contribute to the quality of the decision-trees produced by OXGate when a sufficiently enriched description language was used. As a supplement to hypothesis ordering, post-pruning is not needed. Without hypothesis ordering, the decision-tree assembly could become computationally expensive and post-pruning would only add to

the burden; therefore, post-pruning is not a suitable replacement for hypothesis ordering which serves to reduce the computational expense.

The third area of interest investigated the performance of hypothesis ordering in all four domains. Hypothesis ordering proved to be a powerful method of reducing the processing burden incurred with massive hypothesis generation, with little or no loss in predictive accuracy. This heuristic approach was quickly able to exploit the applicable hypotheses and filter out those developed from inappropriate knowledge. Its behavior across the diverse domains attests to the robustness of hypothesis ordering in OXGate and the use of the non-domination approach to multiple-objective evaluation.

5.6 Summary of Results

This chapter presented a sequence of experiments designed to (a) guide the development of an effective hypothesis ordering mechanism, and (b) assess its utility as a heuristic approach to managing the computational burden of the prolific hypothesis generation expected with opportunistic constructive induction.

Section 5.2 provided a comparison between the basic inductive operation of OXGate and the well-established selective induction system ID3 to ascertain whether a representational advantage exists for either system. Neither representation proved generally superior: the representational advantage depended on the concept being considered. This rough parity supports the use of the binary decision-tree representation necessary in OXGate for the incorporation of constructed hypotheses. At the same time, it provides the foundation to assess the incorporation of knowledge in subsequent experiments since this parity implies that substantial gains in predictive accuracy are not inherent in the representation but, instead, are the result of applying knowledge.

Section 5.4 investigated several design choices necessary to guide the development of the hypothesis ordering mechanism of OXGate. The information-gain confirmation measure proved to be the most effective measure tested, and has been incorporated in both the hypothesis ordering and hypothesis evaluation components. Two multiple-objective evaluation approaches to hypothesis ordering, weighted combination and non-domination, proved superior to the other approaches, including single-objective evaluation with the Quick-Look confirmation measure and single-objective evaluation using simplicity. The weighted combination approach tended to be faster than the non domination approach in the artificial domains. In the complex, real-world, NetTalk domain, the non-domination approach produced slightly better decision-trees than the weighted combination approach, and both approaches were equally fast. For its strengths, particularly its effectiveness at hypothesis ordering, as well as its uniqueness and appeal, the non-domination approach was selected as the standard method for hypothesis ordering.

Once the standard method of hypothesis ordering was established, several approaches to compensate for concept dispersion were addressed: retaining the primitive hypotheses, recycling the rejected hypotheses, and artificially balancing the training data. Retaining the primitive hypotheses proved to be an inexpensive and highly effective method of compensating for the unfavorable interactions of Quick-Look sampling and concept dispersion. This has been incorporated as a permanent feature of OXGate. Recycling the rejected hypotheses provided similar improvements in decision-tree accuracy and simpler decision-trees, but at a high cost in processing time. The accuracy achieved when recycling rejected hypotheses was no better than that obtained by retaining the primitive hypotheses. This result indicates that the non domination approach to hypothesis ordering with the retention of primitives is generally effective at selecting the right hypotheses to reject. The third approach, artificially balancing the training data when using hypothesis ordering, provided some improvement in decision-tree accuracy, but since it

may skew the training data to be unrepresentative of the domain, this approach requires further study.

Following the system design experiments, the robustness of the non-domination approach for hypothesis ordering was investigated. The use of the two tolerance bands to determine the memberships of the primary, secondary, and rejected subsets proved to be well-behaved and robust. The band settings selected to provide a desirable balance between processing speed and predictive accuracy in the Nominal domain also turned out to be the most consistently applicable settings for the Boolean and NetTalk domains, confirming the robustness of the initial settings and the use of tolerance bands.

Finally, I evaluated the performance of hypothesis ordering in all four experimental domains. Hypothesis ordering, particularly the non-domination approach, proved to be a powerful method of containing the explosion of computational overhead produced with prolific hypothesis generation, without a significant decrease in the predictive accuracy and decision-tree conciseness available with the full set of constructed hypotheses. In the complex, real-world domain of the NetTalk Silent concept, not only did OXGate with hypothesis ordering produce decision-trees with considerably improved accuracy over that available with only the primitive hypotheses, but it was able to do so faster. As a heuristic approach, hypothesis ordering behaved as anticipated, providing a good balance between computing efficiency and decision-tree quality.

CHAPTER 6

USING KNOWLEDGE: PRELIMINARY INVESTIGATIONS

The most interesting and certainly the most challenging aspect of opportunistic constructive induction is the incorporation of domain knowledge into the hypothesis generation mechanism in a form suitable to produce useful hypotheses. Section 3.1.1 described the hypothesis generator and proposed some ways knowledge might be applied. This chapter presents preliminary investigations on the application of knowledge. These investigations, particularly the first, are beyond the original scope of this thesis. They are included for the insight they provide toward future research. Section 6.1 discusses learning from experience, which is tantamount to constructing hypotheses based on the knowledge of past successes. Section 6.2 demonstrates the power of applying the right knowledge for hypothesis construction. The experiments demonstrate that the correct knowledge, even a very small amount, substantially improves the performance of decision-tree induction.

Many researchers throughout the history of machine learning have recognized the value of adding some form of knowledge to assist their systems through the learning tasks. Samuel [1967] added several heuristics to guide the adaptation of the scoring function in his checkers-playing program. Lenat [1983] used problem-solving heuristics to guide the mathematical learning system AM. Michalski [1983] proposed a methodology for applying background knowledge to constrain the application of inference and generalization rules during concept induction, and demonstrated its application in the system INDUCE. More recently, several other researchers have demonstrated the effectiveness of applying small amounts of domain-specific knowledge during concept induction. Matheus [1989, 1990] uses fragments of knowledge in CITRE to constrain the generation of new

features during experiential learning and explicitly guide the generalization of these new features (Sections 2.4.3 and 6.1). Specifically, he has demonstrated the efficacy of three pieces of knowledge in learning tic-tac-toe: 1) piece type is important, 2) piece adjacency is important, and 3) features constructed from one section of the board may be applicable elsewhere.[1] MIRO [Drastal and Raatz, 1989] applies domain-specific knowledge to establish an abstract framework for induction. MIRO begins learning with the most general description language (as defined by the knowledge) and specializes the language if necessary to induce the concept description (Section 2.4.4).

Applying small amounts of knowledge to guide concept induction is not unique to symbolic concept learning. Towell *et al.* [1990] use nearly correct background knowledge to establish an initial topology for the neural network system KBANN. The network then learns from training examples to correct the deficiencies in the background knowledge. The knowledge provides expectations of the internal (hidden) nodes of the multilayered network: in essence, it suggests how to construct partial concept descriptions (constructed hypotheses). The original knowledge, although flawed, provides enough guidance for the network to learn the concept better than several knowledge-poor approaches. Abstractly, this approach is similiar to ENIGMA (Section 2.4.6) except that learning is network-based in KBANN rather than symbolic.

The systems just described are samples of a myriad of hybrid systems that combine knowledge and inductive concept learning. They share a common theme: small amounts of the correct knowledge can greatly enhance the quality of inductive learning. Domain knowledge has the potential of enriching the concept description language so completely that substantial components of the target concept, or even the entire concept description, may be found in the vocabulary. The following experiments support this expectation.

[1]Matheus implemented a generalization operator for spatial translation that also provided reflection about the major axes. Rotation about the center and reflection about the diagonals were not addressed.

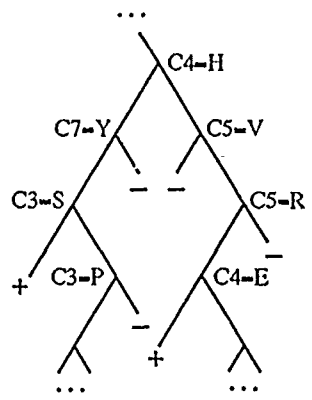
6.1 Learning from Experience

A common method of generating hypotheses is through experiential learning: creating a concept description and examining that description to develop potentially useful hypotheses for continued concept refinement. The concept is then relearned, using a modified description vocabulary consisting of the union of the newly constructed hypotheses and all or a portion of the previous vocabulary. This process is repeated until a suitably concise concept description is obtained or until some other measure of determining convergence is satisfied (e.g., a lack of improvement in predictive accuracy). The systems FRINGE [Pagallo and Haussler, 1989] and CITRE [Matheus, 1989] are two examples of decision-tree-based experiential learning systems (Section 2.4).

Both FRINGE and CITRE create hypotheses by conjoining pairs of features associated with branches leading to positive leaf nodes. This approach implements a simple heuristic: since experience shows that feature A and feature B are both needed to lead to a particular positive leaf, feature (A AND B) will lead to that leaf and may be useful elsewhere. Other binary combinations are also possible, but are not considered for the sake of simplicity. For example, Pagallo [1990] also describes the construction of new conjunctive features from branches leading to negative leaves in the system Symmetric FRINGE, and Yang *et al.* [1991] construct features using disjunction in DCFringe.

A second simplification FRINGE uses is to consider only the features associated with adjacent *fringe* branches in the tree as operands to the conjunction. The two features composing a fringe pairing are those associated with the two lowest-level branches leading to a positive leaf. In Figure 6.1, the fringe pairs are ($C7 = Y$, $C3 = S$) and ($C5 = R$, $C4 = E$). This simplification drastically reduces the number of potential new hypotheses that must be considered.

CITRE also simplifies the number of hypotheses to consider by applying one of five user-selectable biases to the operand selection. Selecting the fringe pair is one of those



All possible pairings (of those shown):

{(C4=H, C7=Y), (C4=H, C3=S), (C4=H, C3=P)
(C7=Y, C3=S), (C7=Y, C3=P), (C5=R, C4=E)}

Knowledge-guided pairings:

{(C4=H, C3=S), (C4=H, C3=P), (C5=R, C4=E)}

Generalization of knowledge-guided pairings:

{(C4=H, C3∈{S,P})}

At each test, *True* is to the left branch.

Figure 6.1 Learning from Experience. This example shows a possible portion of a decision-tree for the NetTalk Silent concept to illustrate the search for useful constructs. Positive tests leading to positive leaves are the operands (pairings) for forming binary conjunctions. Domain knowledge can be used to constrain the possible combinations. The effect of applying a domain-independent generalization operator, the merging of disjunctive regions, is also shown. The decision-tree is learned from a collection of seven-character windows (C1 through C7) on a dictionary of words. An example of such a window is from the word "symphony," with the letter "h" as the center character: s[ymp(h)ony]. This window corresponds to the pairings (C4=H,C7=Y), (C4=H,C3=P), and (C7=Y,C3=P), and supports the generalization (C4=H,C3∈{S,P}). A word that provides multiple windows with silent center characters is "haberdashery": [hab(e)rda] and [ash(e)ry_] both support (C5=R,C4=E); and [das(h)ery] supports (C4=H,C7=Y), (C4=H,C3=S), (C7=Y,C3=S), and the generalization (C4=H,C3∈{S,P}).

biases (*fringe*). Another bias, *all*, selects all possible pairings. *Root* selects the two features associated with the highest-level branches leading to a positive leaf: an example is $(C4 = H,\ C7 = Y)$ in Figure 6.1. *Root-fringe* selects the features corresponding to the single highest-level and single lowest-level branches leading to a positive leaf, such as $(C4 = H,\ C3 = P)$. The fifth bias, *adjacent*, selects features associated with any two adjacent branches leading to a positive leaf. The operand pairs selected using the *fringe* and *root* biases are a subset of those selected using the *adjacent* bias.

The application of these biases restricts the selection of operands from within a subtree leading from the root node to a positive leaf. Section B.3 defines these restrictions formally. The biases are implemented as filters that act on the positions of the features

within the tree. Knowledge may also be applied as filters, but, in addition to being concerned with the position of features (syntactic filtering), knowledge-based filters may also be concerned with the meaning of the terms (semantic filtering) or their relationship to the domain (contextual filtering). When learning from experience, OXGate limits the number of potential new hypotheses through contextual filtering by the procedural application of domain knowledge.

Figure 6.1 illustrates a possible portion of a learned decision-tree for the NetTalk Silent concept. The given attributes are the seven characters C1 through C7. The knowledge used to guide the pairings in Figure 6.1 can be stated as: *the center character is the focus of attention, and the characters adjacent to the center provide the most information-gain* [Lucassen and Mercer, 1984]. This knowledge focuses the selection of features for construction on those related to the attribute pairs (C4,C3) and (C4,C5).

The implementation of the experiential learning mechanism in OXGate is through a post-processing procedure called *Koala.* Koala searches for all pairings of *true* hypotheses leading to positive leaf nodes, but keeps only those defined by knowledge as coded in search patterns. Invocation of the procedure takes the form

(koala *tree pattern focus-attributes focus-values*
scope-attributes scope-values trivia-tolerance)

where *tree* is the decision-tree to be learned from,

pattern is the form (template) of the hypotheses being searched for,

focus-attributes are the attributes used as the focus of the search,

focus-values are the values accepted in the focus template,

scope-attributes are the attributes used for the scope of the search,

scope-values are the values accepted in the scope template, and

trivia-tolerance is the number of times a pairing must occur.

Using the NetTalk example, the search for pairs of characters adjacent in the word to the center position would be invoked by

(koala *tree* '(EQP *attribute value*) 'C4 '* (adjacent 'C4) '* 1) (6.1)

The *pattern* "(EQP *attribute value*)" indicates only primitive hypotheses are to be considered. It is a test (*true* or *false*) to see whether the value of the attribute for the instance under consideration is the same as the stated value. The *focus-attribute* "C4" indicates the search is focused on the center window character C4. The *focus-value* "*" is a wildcard, telling Koala to accept any value for attribute C4. The focus template is created from these three parameters, preparing the upcoming search to locate any hypotheses of the form (EQP 'C4 *any-letter*). The scope of the search is defined by the next two parameters. The *scope-attributes* are defined in this case by the function "(adjacent 'C4)," indicating either attribute C3 or C5 is acceptable. The *scope-value* "*" indicates that any value will be accepted in the scope templates. From these two scope parameters and the original pattern, Koala creates two scope templates, (EQP 'C3 *any-letter*) and (EQP 'C5 *any-letter*). The final parameter, *trivia-tolerance*, is the number of times an acceptable pairing must occur before being retained (learned from experience). Here, it is set to one.

These parameters and templates are used by Koala to sort through the space of all possible pairings defined by the decision-tree. The knowledge-guided pairings will be the ones containing valid matches to the focus template and a scope template. In addition to the construction of the new hypotheses, Koala builds generalizations of the pairings as depicted in Figure 6.1. Once all of the acceptable pairings are collected, the focus template is used to guide the generalization. For each instantiation of the focus template, the pairings containing the instantiation are collected, and the scope instantiations are

grouped by *scope-attribute* to create membership functions as shown in Figure 6.1, for example,

(AND (EQP 'C4 'H) (MEMBER (WHATIS 'C3) '(S P W G T C)))

meaning "H is silent when it follows S, P, W, G, T, or C."

The operator WHATIS looks up the value of the attribute for the instance under consideration. It is assigned a cost of one (see Section 4.2.2). MEMBER tests if this value is a member of the given set, and is also assigned a cost of one. The function (EQP *attribute value*) is actually implemented as (EQ *value* (WHATIS *attribute*)), testing if the stated value matches the value found by WHATIS. The test for equality has zero cost, but since WHATIS has a cost of one, EQP also incurs a cost of one. These costs are used to determine both the simplicity of hypotheses and the complexity of the resultant decision-trees. Thus, primitive hypotheses have a cost of one, binary constructs have a cost of two (the Boolean operators have zero cost), and generalizations such as that shown above have a cost of three (simplicity of 1/3).

Another feature of Koala provides constraints on the original collection of possible pairings. By setting the global variable **tree-adjacent** true, only physically adjacent terms in the decision-tree are collected. With minor modification this variable could be used to select from any of the syntactic filters discussed on pages 155 through 156. For the experiments of this chapter, **tree-adjacent** was false, corresponding to the syntactic bias *all*.

With **tree-adjacent** set false and *trivia-tolerance* set to one, Koala limits the selection of operands for construction based only on their content. The knowledge incorporated in Equation (6.1) allows only primitive hypotheses involving the named attributes to be selected. This knowledge can be viewed as a filter applied by Koala on the set of all possible constructions derivable from the decision-tree (see Section B.3).

Koala provides a convenient mechanism for incorporating knowledge to constrain the extent of experiential learning. This is necessary to avoid overlearning when drawing upon a single decision-tree. In the example, the setting of a few simple parameters provides Koala with all the guidance it needs to learn every combination of the center window character (C4) and its nearest neighbors (C3 and C5) appearing along the way to a positive leaf. An alternative view is that the experience contained in the decision-tree constrains the application of domain knowledge by the hypothesis generator. Instead of generating all possible hypotheses indicated by the knowledge (as was done for the experiments of Chapter 5), Koala limits the constructed hypotheses to those suggested by experience to be useful. This combination of expectation and experience defines the most promising extensions to the concept description language.

6.1.1 Experimentation

One way to judge the efficacy of the knowledge used to guide experiential learning is by examining the utility of the learned constructions in subsequent induction. This experiment compares the performance of three sets of constructed hypotheses: two sets were created using knowledge for guidance, the third was not. Hypothesis ordering was disabled to isolate the effects of the knowledge itself. The experiment shows that correct knowledge focuses the experiential learning on those constructed hypotheses most beneficial to subsequent decision-tree assembly.

The first step in the experiment was to create the initial decision-tree and the three sets of constructed hypotheses. OXGate was run on the 2000 training examples of the NetTalk Silent concept, using only primitive hypotheses. This decision-tree was then used as the basis for experiential learning. The first set of knowledge-guided constructed hypotheses was created using Koala as shown in Equation (6.1), producing 46 binary conjuncts and 11 generalizations (*Learned w/DK* in Figure 6.2). The second

set of knowledge-guided constructed hypotheses *(Learned w/DK(-))* is a subset of the *Learned w/DK* hypotheses: the 37 binary conjuncts subsumed by the generalizations were removed, leaving 9 binary conjuncts and 11 generalization hypotheses. To generate the third set of constructed hypotheses, the ones created without the guidance of knowledge *(Learned wo/DK)*, Koala was invoked with wildcards for the attributes and values of both the focus and the scope. This provided all possible pairings and generalizations (247 binary conjuncts and 97 generalizations).

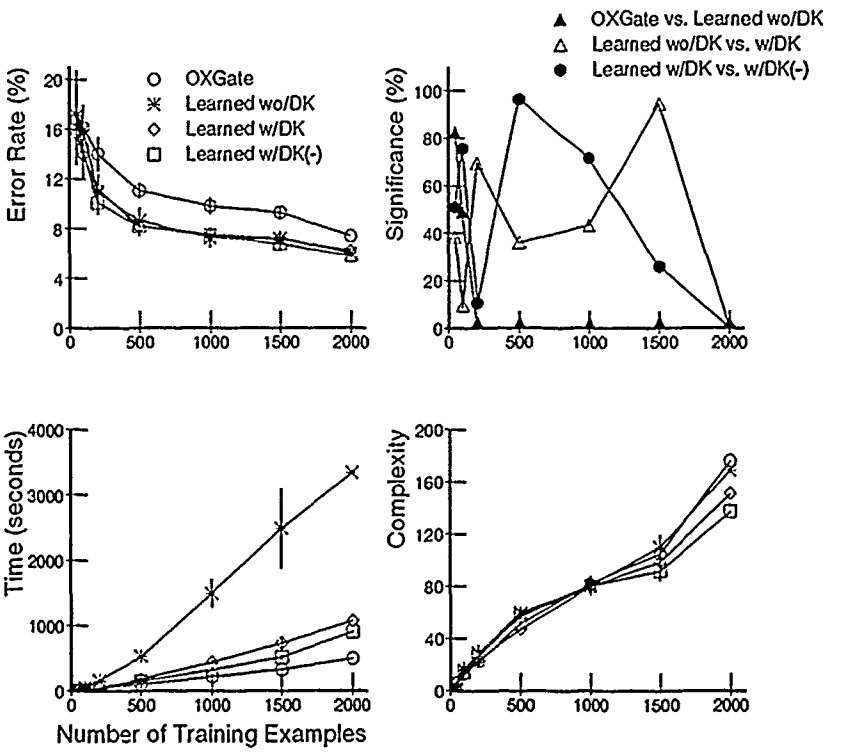


Figure 6.2 Learning from Experience: NetTalk Silent Concept.

Figure 6.2 presents the effects of a single iteration of experiential learning. A single iteration consists of assembling an initial decision-tree and constructing hypotheses from the evidence in the initial tree (as described above), and then measuring the performance of subsequent decision-trees created with the extended description language. The lower right graph displays the complexity of the resultant decision-trees. *Complexity* is

a modified form of the measure *primitives* used in the experiments of Chapter 5. It includes the number of primitive hypotheses incorporated in the decision-tree and accounts for the cost of the membership function MEMBER in the generalizations. *Complexity* is the inverse of the measure of *simplicity* defined in Section 4.2.2. As other operators of non-zero cost are introduced into OXGate these two measurements will diverge further. Where no membership functions are included, *complexity* and *primitives* are identical measurements.

The plots labeled *OXGate* show the baseline performance of the decision-trees created using only the set of primitive hypotheses. *Learned wo/DK* shows the marked improvement in predictive accuracy supplied by the addition of the 344 constructed hypotheses to the description language. Processing time also increased correspondingly. Interestingly, the complexity of the decision-trees remained nearly the same. The number of decision nodes decreased with the addition of learned hypotheses, but, in this case, the increased cost of some nodes offset the reduction in structural complexity.

The plots labeled *Learned w/DK* present the results of using the knowledge-guided constructed hypotheses, a subset of those used for *Learned wo/DK*. Although the additional knowledge did not appear to improve the predictive accuracy, the reduction in the number of hypotheses did have a substantial impact on the processing speed. This indicates that the knowledge served to remove the hypotheses of little or no utility, leaving what was needed to assemble good decision-trees. *Learned w/DK(-)* shows the results with the non-redundant subset of the *Learned w/DK* hypotheses. This more limited description language provided powerful generalizations, yet prevented overfitting with the small sample sizes, resulting in a slight increase in accuracy, more compact decision-trees, and faster processing.

6.1.2 Summary: Learning from experience

The approach used in OXGate for experiential learning, the application of Koala, is similar to that of FRINGE and CITRE: a decision-tree is developed and then evaluated to determine useful constructions. It is different in that only domain knowledge (contextual information provided by the developer or available in the hypothesis generator) is used in Koala to constrain the generation of experientially learned hypotheses. FRINGE uses the positions of the operands in the decision-tree (a syntactic bias) to select components for hypothesis construction. CITRE uses a combination of syntactic bias and user-supplied domain knowledge to guide construction. Like CITRE, Koala uses knowledge to constrain the selection of operands suggested by the decision-tree, and to produce domain-dependent generalizations. The difference between CITRE and OXGate/Koala is primarily one of focus: CITRE's principal mode of operation is the syntactically guided search for new constructions, while in OXGate the application of contextual knowledge is the primary mode of operation. Koala is an extension to OXGate for investigating the application of contextual knowledge and experiential learning in hypothesis generation.

The knowledge used in the experiment focused Koala on constructing hypotheses from primitives involving the target character C4 and those using the contextually adjacent characters C3 and C5. When multiple pairs were found using a common primitive involving C4, a generalization was also created. The experiment clearly showed an increase in predictive accuracy when hypotheses created through experiential learning were added to the description language. It also demonstrated the focusing effect of correct domain knowledge: the hypotheses most beneficial to subsequent decision-tree assembly were retained, with a substantial improvement in processing speed.

6.2 Applying the Right Knowledge

The power of applying the right knowledge for hypothesis generation is demonstrated in this section. The first two experiments show the application of two sets of knowledge in both the Boolean and Nominal domains. The knowledge was syntactic, specifying an expectation of the form of components of the concept description. The third experiment demonstrates the power of a single hypothesis constructed from correct contextual knowledge in the NetTalk domain. In these three experiments, hypothesis ordering was disabled to isolate the effects of the knowledge. The experiments of Chapter 5 demonstrate the effects of hypothesis ordering: generally, hypothesis ordering revokes the contributions of the weaker or less applicable knowledge.

6.2.1 Boolean 3-term 3DNF concepts

The first set of knowledge suggests that negations of the primitives, binary conjuncts, and binary disjuncts are useful constructions (see Appendix C). This knowledge is applied procedurally by invoking a handful of special-purpose routines. The overall control of the procedures has been partially generalized so that subsets of the hypotheses can be selectively generated by changing the keywords *NOT*, *AND*, and *OR*. For the Boolean concepts, the 24 primitive hypotheses yield 24 negations, and these 48 unary hypotheses are used to generate 1140 binary conjuncts and 1140 binary disjuncts (shown as *DK-type1* knowledge in Figure 6.3). The negations of primitives are logically redundant hypotheses in the Boolean domain, and were originally produced to help flood the hypothesis ordering mechanism for the experiments of Chapter 5. Since the hypothesis ordering mechanism is disabled for the following experiment, the negations simply serve to illustrate that, although knowledge might *suggest* generating a certain type of hypothesis, the knowledge may not be fully applicable to the domain at hand.

The second set of knowledge suggests that trinary conjunctions of the primitive hypotheses might be useful. Again, a special-purpose procedure builds the constructions. From the 24 primitive hypotheses in the Boolean domain, 264 binary conjuncts and 1760 trinary conjuncts are formed (shown as *DK-type2* knowledge in Figure 6.3).

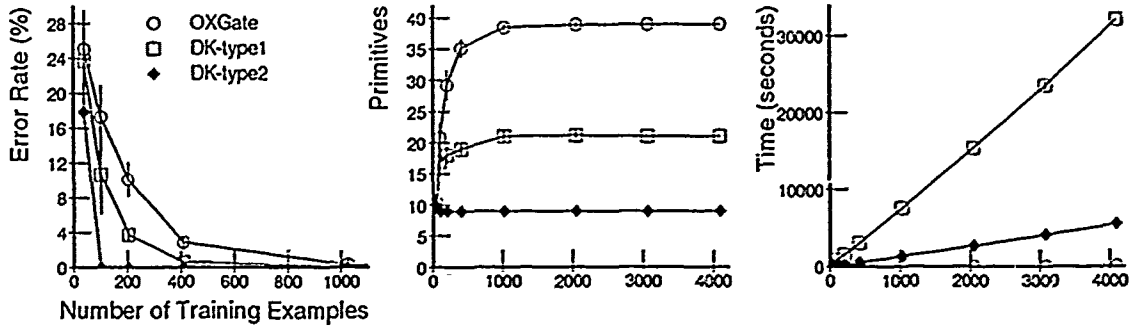


Figure 6.3 Using Correct Knowledge: Boolean 3-term 3DNF. Note: The number of training instances shown for the error-rate is truncated after 1024 examples for clarity of presentation.

Figure 6.3 dramatically illustrates the effect of proper knowledge when learning the Boolean 3-term 3DNF concepts. The plots labeled *OXGate* show the baseline performance using only the 24 primitive hypotheses. *DK-type1* shows a marked improvement, mainly from the incorporation of the binary conjuncts in the decision-trees. The processing time increases tremendously since the decision-trees still require several levels of branching, and a large number of hypotheses must be considered at every level.

The plots for *DK-type2* show that with trinary conjuncts, the decision-trees perfectly represent the concepts with only 102 training examples, about 1/10th the number required for *OXGate*. The processing time is still much larger than when using only the primitive hypotheses since the number of hypotheses to consider is much larger. The processing time is substantially smaller than when using the *DK-type1* hypotheses because convergence to the solution occurs much faster. *DK-type2* knowledge, particularly the

use of trinary conjunct, works well because it perfectly matches the syntactic structure of the Boolean 3-term 3DNF concepts:

((*trinary-conjunct*) OR (*trinary-conjunct*) OR (*trinary-conjunct*))

Comments: An alternative approach to generating and evaluating the sets of constructed hypotheses is through the use of *look-ahead*, a common reasoning strategy for decision-making [Barr and Feigenbaum, 1982]. For decision-tree induction, look-ahead involves postponing the decision to incorporate a hypothesis until the quality of the potential subtrees can be determined [Hartmann *et al.*, 1982]. A system using look-ahead performs "what-if" reasoning on each hypothesis: "What is the best subtree available if hypothesis X is used to split here?" The best hypothesis for the current node is the one producing the best subtree overall. For example, using a look-ahead of two, a system such as OXGate would pretend to split on a hypothesis, and then find the best hypotheses for splitting at the two nodes (left and right) lower in the decision-tree. The value of the hypothesis at the current node of interest is the aggregate information-gain provided by the three splits (entropy at the current node minus the remaining entropy two levels down). After considering each hypothesis at the current node, the system would select the hypothesis with the highest value, i.e., the best potential subtree.

The binary hypotheses constructed from the *DK-type1* and *DK-type2* knowledge are equivalent to portions of the subtrees considered with a look-ahead of two (Figure 6.4). Since look-ahead allows an inductive system to make more informed choices at a node, the resultant decision-trees are potentially better than those created with no look-ahead. Similarly, the binary hypotheses used in this experiment provided an enriched description language, also leading to more informed decisions and better decision-trees.

One of the drawbacks of using look-ahead is the potentially large increase in processing time with each added level of look-ahead depth, due to the number of additional

combinations to be investigated. However, as the decision-tree is developed, look-ahead becomes less expensive because the number of hypotheses to consider is reduced: some have been incorporated in the tree and are no longer available. The same effects occur with hypothesis generation from syntactic knowledge: the more complex syntactic structures yield greater numbers of hypotheses, each of which requires evaluation. Syntactic filtering of hypotheses (Section 3.1.4) can reduce the number of hypotheses considered at the lower nodes of the decision-tree, providing a similar reduction in computational expense as that found with look-ahead. In addition, the use of hypothesis ordering provides a more powerful focusing mechanism to reduce the processing time, by identifying the most promising hypotheses and removing the least promising ones. The analogous behavior with look-ahead would be to determine a priori which subtrees to consider first and which to avoid.

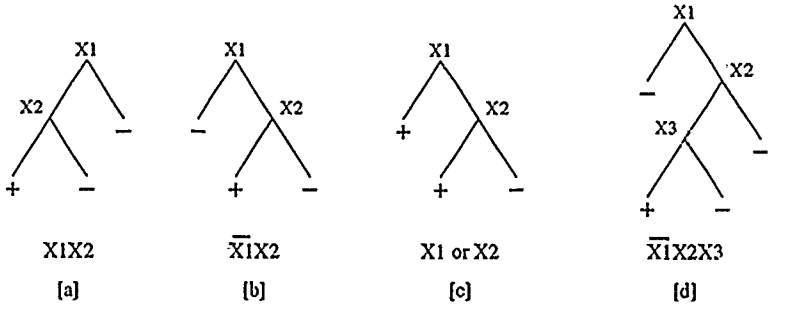


Figure 6.4 Relating Look-Ahead to Hypothesis Construction. Several potential subtrees created with a look-ahead of two ([a], [b] and [c]) and three ([d]) are shown. The equivalent constructed hypotheses are provided below the subtrees.

Using look-ahead does not always incur a significant processing burden. Ragavan *et al.* [1991] demonstrated that with the right level of look-ahead for the target concept, a more accurate decision-tree can be produced than without look-ahead, yet with little additional processing time. This is because the proper, compact decision-tree can be identified and developed quickly. Without look-ahead, the system tends to assemble large, inaccurate trees. A similar effect was demonstrated in Figure 6.3 with the hypotheses created using

the *DK-type2* knowledge: three of the trinary conjuncts were all that were needed to quickly learn the Boolean 3-term 3DNF concept. For both look-ahead and hypothesis generation, if the size of the terms in a k-term lDNF Boolean function is known (i.e., the value of l), then the optimum depth of look-ahead (l) or width of the syntactic template (also l) is specificied.

Determining the correct level of look-ahead or the correct syntactic template for hypothesis generation is problematic. Ragavan and Rendell [1991] propose a measure of concept dispersion that estimates the difficulty of learning the concept with a given set of hypotheses. They are currently investigating heuristics for applying this and other measures for predicting the proper level of look-ahead [Ragavan *et al.*, 1991]. When available, these heuristics should be incorporated into the hypothesis generator of QXGate.

6.2.2 Nominal Concept B

Figure 6.5 shows the results of applying the same approach used in Section 6.2.1 to Nominal Concept B (Appendix D). The first generation of hypotheses used the 25 primitive hypotheses to create 25 negations, 1225 binary conjuncts, and 1225 binary disjuncts (*DK-type1* knowledge in the figure). The second generation produced 229 binary conjuncts and 915 trinary conjuncts (*DK-type2* knowledge). In this domain, inconsistencies such as ((*color* = *red*) AND (*color* = *blue*)) were not generated. With this concept, *DK-type1* performed generally better for the smaller training set sizes. The processing time for *DK-type1* was substantially greater due to the larger number of hypotheses to test.

Analysis of the resultant decision-trees reveals that several of the *DK-type1* hypotheses were incorporated early in decision-tree development with small training set sizes, accounting for their improved accuracy. For instance, with 135 training instances, one decision-tree consisted of the binary disjunct ((*color* = *green*) OR (*size* = *huge*)) and

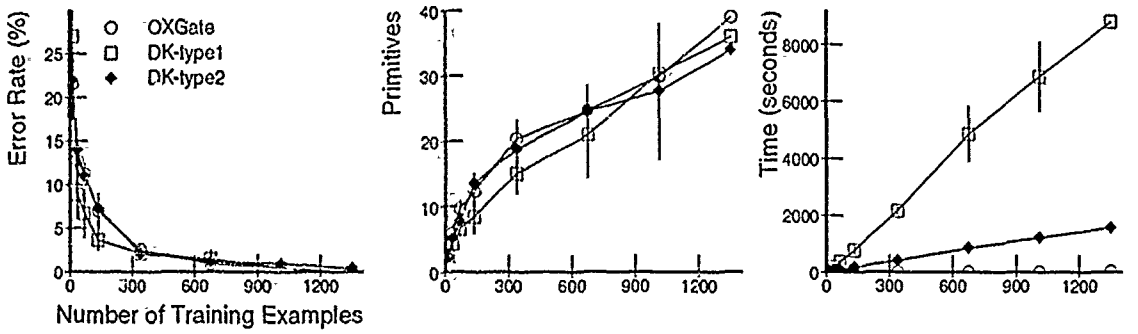


Figure 6.5 Using Partially Applicable Knowledge: Nominal Concept B.

three binary conjuncts. With the full set of 1350 training examples, the decision-trees were more complex: one consisted of seventeen constructed and three primitive hypotheses. The decision-tree was difficult to decipher because it described another of the many logically equivalent decision-trees completely representing Nominal Concept B.

The *DK-type2* knowledge provided very few useful hypotheses for learning this concept. The decision-trees developed using small training set sizes were very similar to their *OXGate* counterparts (created using primitive hypotheses only), with the occasional inclusion of one or two binary conjuncts. The decision-trees created with the full set of training data contained one trinary conjunct ((*color* = *blue*) AND (*taste* = *bitter*) AND (*shape* = *triangle*)), four binary conjuncts, and twenty-three primitive hypotheses.

The syntactic knowledge provided for *DK-type1* was moderately useful for Nominal Concept B. Since the structure of the concept does not reflect an obvious syntactic pattern, large increases in performance should not be expected. The *DK-type2* knowledge had little utility for learning this concept. The binary and trinary conjuncts did not provide the right extensions to the description language and went largely unused. Generally, the proposed syntactic knowledge was not applicable: the lack of substantial improvement in performance reflects this.

The measure of concept dispersion proposed by Ragavan and Rendell [1991] may predict that these two description languages, although substantially enriched, are not particularly useful for learning this concept. This measure, once fully understood and developed, may be very useful to guide the selection of the proper syntactic structure to use for hypothesis generation. As Breiman *et al.* [1984] relate, the development of new and useful hypotheses "is an art guided by the analyst's intuition and preliminary exploration of the data." Measures such as concept dispersion and the associated heuristics for applying them are necessary to automate hypothesis generation effectively.

6.2.3 NetTalk Silent concept

The third experiment uses a piece of contextual knowledge to generate a single powerful hypothesis: "when a character is repeated, the second is silent." This heuristic is generally true, with the exception of words like *zoology*. Since the NetTalk Silent concept is concerned about silence in the center window position (C4), this piece of knowledge translates to the equality hypothesis (EQP (WHATIS 'C3) (WHATIS 'C4)), a test to determine if (C3=C4) is true.

Figure 6.6 presents the marked improvement in all three performance measurements with the addition of the single hypothesis. *OXGate* shows the behavior when only the primitive hypotheses were used for decision-tree assembly. *Learned w/DK(-)* shows the performance when the set of 21 empirically learned (non-redundant) hypotheses described in Section 6.1 were included. *Learned w/DK(+)* presents the performance when the equality hypothesis was used in addition to the 21 learned hypotheses.

The equality hypothesis subsumes only three of the nine binary hypotheses of *Learned w/DK(-)* and a portion of one of the generalized hypotheses, yet provided a large improvement in the predictive accuracy, produced substantially simpler decision-trees, and allowed *Learned w/DK(+)* to complete faster than *Learned w/DK(-)*. This

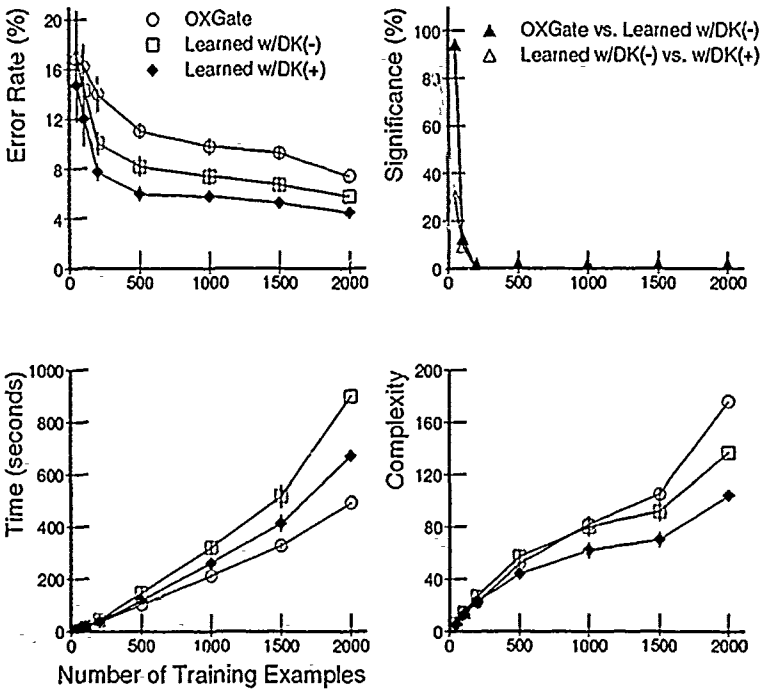


Figure 6.6 Adding Correct Knowledge: NetTalk Silent Concept.

simple addition to the description language demonstrates the power available with even small amounts of the correct knowledge.

Comments: Other pieces of knowledge could also prove useful for quick and effective decision-tree induction. Their utility is directly related to the amount of instance space they cover, i.e., the percentage of instances correctly identified by the application of the knowledge. The reason (C3=C4) is so effective is that it occurs in a large number of instances and accounts for a significant percentage of all the ways the center character could be silent. Another potentially powerful piece of knowledge suggests that the second vowel in a diphthong (e.g., ea, oi, and ou) is generally silent. The hypothesis discovered in Section 6.1, "h is silent when it follows s, p, w, g, t, or c" represents highly useful

semantic knowledge about the behavior of the letter "h" that could have been proposed a priori.

Knowledge that is more specialized is less relevant to the concept in general, but may be highly useful to uncover portions of the concept sparsely represented by the training data. One such piece of semantic knowledge might suggest that the letter "y" following a vowel will generally be silent. Another highly specialized piece of knowledge might suggest that the "q" in words with "cqu" (e.g., acquaint, lacquer, and racquet) is always silent. A similar piece of knowledge might suggest that the "u" in words ending with "que" (e.g., unique, torque, and critique) is always silent. A slight generalization of this knowledge, "que" appearing anywhere, is not applicable though. It is sometimes silent (e.g., bouquet, conquer, and etiquette) and pronounced at other times (e.g., banquet, conquest, equestrian, frequent, and eloquent).

Knowledge can also be used to describe which hypotheses should not be generated, i.e., hypothesis screening as described in Section 3.1.1. Many letters in specific combinations are almost never silent (e.g., the "k" in "rk"). This semantic knowledge could be applied as a set of filters to the output of a mechanism that generates all combinations of letters, with the net result being a set of "non-disallowed" hypotheses: a superset of the applicable hypotheses. When both approaches to hypothesis generation are used, one to generate promising hypotheses and the other to screen out the most useless, the enriched description language can be focused to include only the most potentially useful hypotheses even before hypothesis ordering is invoked.

6.3 Summary and Comments

This chapter presented some preliminary investigations on the application of knowledge to hypothesis generation. These investigations were beyond the original scope of

this thesis, but were included for the insight they provide towards future research. The experiments in this chapter demonstrate the efficacy of even small amounts of correctly applied knowledge and support the assertion that domain knowledge provides the means for effective induction of complex concepts.

Section 6.1 described the experiential learning addition to OXGate, Koala. Koala examines an assembled decision-tree, collecting promising decision points (true hypotheses leading to positive leaves) to use as operands for constructing binary conjuncts and generalizations of those conjuncts. Knowledge is readily incorporated in the invocation of Koala to constrain the construction of hypotheses to those which are both predicted useful by the knowledge and supported by experience. Abstractly, this interaction may be viewed as applying knowledge to limit the tentative conclusions drawn from a single learning session. Conversely, it may also be viewed as using experience to guide and constrain prolific hypothesis generation based on what OXGate "thinks it knows." Either view sustains the suggestion that experiential learning is a unique and useful addition to the set of hypothesis generation mechanisms.

The experiment with the NetTalk Silent concept clearly showed an increase in predictive accuracy when hypotheses created through experiential learning were added to the description language. It also demonstrated the focusing effect of correct domain knowledge: the hypotheses most beneficial to subsequent decision-tree assembly were retained, with a substantial improvement in processing speed. These results are in accord with those obtained by Matheus [1990] for the game of tic-tac-toe, where the addition of knowledge about piece adjacency, piece type, and spatial translation focused feature construction during experiential learning to produce highly accurate decision-trees within relatively low processing times.

The second part of the investigation, Section 6.2, demonstrated the power of applying the right domain knowledge during hypothesis generation by examining the benefits of

applying small amounts of correct knowledge. The knowledge used in the experiments on the Boolean 3-term 3DNF concepts and Nominal Concept B was syntactic, specifying expectations of the form of components of the concept description. Where the expectation matched the actual form of the concept, such as the trinary-conjunction form in the Boolean experiment, the generated hypotheses provided the means for extremely rapid convergence to the concept description with small sample sizes. Nominal Concept B is a complex structure with few repeated syntactic forms; consequently, the proposed knowledge provided only modest improvements in decision-tree performance.

The success of the trinary-conjunction form of knowledge for the Boolean 3-term 3DNF concepts is directly attributable to the perfect match between the syntactic form of the hypotheses (trinary conjuncts) and the disjoint terms in the concepts. This use of the correct syntactic template reflects the use of the correct level of look-ahead, providing the hypothesis evaluator with the means to make well-informed decisions.

The third experiment of Section 6.2 demonstrated the potency of correct contextual knowledge with the NetTalk Silent concept. When the single, well-founded, hypothesis (C3=C4) was added, which generalized some of the results from experiential learning, OXGate achieved a substantial and consistent improvement in each of the three performance measurements.

Domain knowledge has the potential of enriching the concept description language so completely that substantial components of the target concept, or even the entire concept description, may be found in the vocabulary. Clearly, applying the right knowledge for hypothesis construction provides the basis for highly efficient decision-tree induction. Recognition of the power of background knowledge is the foundation of the entire class of explanation-based (or analytic) learning systems, which rely heavily on knowledge for concept induction (Section 2.4.6). There exists a strong synergism between analytic and empirical (similarity-based) learning: good domain knowledge reduces the amount of

training data required to learn a concept and provides rapid convergence to a compact representation (top-down reasoning), and the regularities in the training data can compensate for deficiencies in the domain knowledge (bottom-up reasoning). Hybrid systems such as MIRO (Section 2.4.4), ENIGMA (Section 2.4.6), and OXGate take advantage of both approaches as needed to compensate for sparse or noisy data, and incomplete or inconsistent knowledge. The fact that over thirty papers addressed the combination of empirical and explanation-based learning at the Sixth International Workshop on Machine Learning (1989) reflects the importance of integrating both approaches.

CHAPTER 7

CONCLUSION

7.1 Thesis Summary

This thesis lays the foundation for the investigation of opportunistic constructive induction: using fragments of knowledge to propose potentially useful new terms during the inductive process. The primary objective of this research effort is to develop a mechanism to allow the use of domain knowledge in an unrestricted, opportunistic fashion while maintaining a manageable computational load. This primary objective was achieved by accomplishing the following secondary objectives:

1. Penetration of the mechanics of induction to incorporate flexible search guidance, including an interleaving of deductive and inductive mechanisms. (Chapter 2)

2. Development of a modular system architecture to implement the four components of the opportunistic constructive induction process: hypothesis generation, hypothesis ordering, hypothesis evaluation, and hypothesis incorporation. The prototype opportunistic constructive induction system OXGate provides a domain-independent development and testbed environment for future work in applying domain knowledge to guiding decision-tree construction. (Chapter 3)

3. Development of a robust hypothesis ordering mechanism to manage the potentially enormous computational burden produced by the uninhibited exploration of hypothesis space. (Chapter 4)

4. Investigation into the development of a robust hypothesis generation mechanism that uses fragments of domain knowledge and an assessment of the current state

of the inductive process to explicitly create new hypotheses intended to enrich the concept description language for further induction. (Chapters 3 and 6)

The phase of research described in this thesis is a portion of a larger envisioned effort: so far it has only scratched the surface of the hypothesis generator. Since the hypothesis generator is expected to produce large numbers of hypotheses, the hypothesis ordering mechanism was first developed to contain the effects of prolific hypothesis generation for both tractable experimentation with knowledge and eventual real-world operation. The current implementation consists of the overall OXGate architecture, baseline hypothesis ordering and hypothesis evaluation modules, and partial implementations of the hypothesis incorporation and central blackboard mechanisms. The hypothesis ordering and hypothesis evaluation components are essentially complete, providing the fundamental capability for exploration of the hypothesis generation mechanism and its interaction with hypothesis incorporation.

The experiments accompanying and supporting the development of OXGate investigated two major fronts: the costs and benefits of hypothesis ordering, and the effects of select pieces of domain knowledge. The principal investigations of this thesis concentrated on the use of hypothesis ordering; the preliminary investigation of the application of knowledge to hypothesis generation was beyond the original scope of this thesis.

The hypothesis ordering mechanism acts as a filter between the hypothesis generation and hypothesis evaluation phases of the constructive induction process. Its function is to make initial estimates of the utilities of hypotheses, present the most promising ones for rigorous evaluation, and reject the seemingly useless hypotheses. For this it uses a competitive mechanism based in part on small samples of the training data. This thesis explored several multiple-objective evaluation functions as the basis of hypothesis ordering, and established the non domination method as a psychologically satisfying, functionally robust, and computationally practical approach. Hypothesis ordering was

not intended to optimize predictive accuracy, but rather to serve as a practical approach to creating a manageable testbed environment and pave the way for operation in the real world. Hypothesis ordering with the non-domination method proved to be an effective heuristic method that lives up to its expectations: it sacrifices a small amount of predictive accuracy for large improvements in processing speed.

The preliminary investigation of the application of knowledge to hypothesis generation showed that even small amounts of correct knowledge provide powerful guidance for constructive induction. The experiments examined several variations on the type of knowledge used for hypothesis generation and the stage of induction in which the knowledge was applied. In Chapter 6, contextual knowledge (knowledge about content) was used by the post-processing procedure Koala to constrain the conclusions drawn when learning from experience. This combination of expectation and experience produced a small set of very useful hypotheses for subsequent learning sessions. In other experiments (Chapters 5 and 6), syntactic (structural) and contextual knowledge was used by procedural mechanisms to generate hypotheses prior to decision-tree induction. Hypotheses generated with proper knowledge provided the means for extremely rapid convergence to the concept description with small sample sizes. This investigation established that domain knowledge has the potential of enriching the concept description language so completely that substantial components of the target concept or even the entire concept description may be found in the vocabulary.

Although the limitations of the implementation and experiments certainly warrant conservativism in claims of success and promises of future capabilities, I am confident the approach presented in this thesis is a robust and effective mechanism for utilizing domain knowledge to guide the constructive induction process. Domain knowledge provides the means for effective induction of complex concepts, and hypothesis ordering makes the operation of OXGate practical even with prolific hypothesis generation.

7.2 Specific Contributions

The novel machine learning aspects and contributions of this research are:

- The refinement of a conceptual framework for the inductive process that encourages the incorporation of deductive processes using background knowledge to suggest fragments of the concept description. This framework maps directly into an implementation architecture integrating inductive and deductive mechanisms in an opportunistic fashion.

- The investigation of the use of a hypothesis ordering mechanism to act as a filter between the generation and test phases of new term creation. Hypothesis ordering is implemented with a competitive mechanism in which small samples of the training data are used to focus the system's attention on the most promising hypotheses and to reject the most useless. This investigation explored several multiple-objective evaluation functions as the basis of the filtering operation, and established the non-dominance method as a robust and computationally practical approach.

- The establishment of the groundwork for a diverse and flexible hypothesis generation mechanism. A hypothesis generator is any mechanism that produces a testable hypothesis given the current state of the inductive process.

7.3 Suggested Future Work

This thesis lays the foundation of a much larger research effort. The prototype system OXGate provides a tool for the next stage, the development of the hypothesis generation component. Two major undertakings are involved in the implementation of the hypothesis generator. The first is the development of automated mechanisms for hypothesis generation as described in Section 3.1.1. The second is the development of cueing

mechanisms for the state changes resulting from hypothesis incorporation to trigger the application of domain knowledge in the hypothesis generator.

The hypothesis generation component was implemented only to the extent necessary to perform controlled experiments on the effectiveness of certain pieces of domain knowledge. The generator was not developed to the degree necessary to become an autonomous and integral part of OXGate. Since OXGate is incomplete, the full opportunistic capabilities of a completed system architecture have not been demonstrated. Specifically, the experiments did not require that knowledge be invoked to generate additional hypotheses based on a partial or tentative concept description. However, the experiments did examine several variations on the types of knowledge used and the stage of induction in which the knowledge was applied. In the experiments of Chapter 5, declarative knowledge was used to create hypotheses before beginning the initial decision-tree induction. This approach is similar to the situation in which the system generates hypotheses based on the recognition of the problem domain. Another set of experiments (Section 6.1) applied both declarative and procedural knowledge to a completed decision-tree, to create hypotheses for future use (learning from experience). These experiments demonstrate the feasibility and flexibility of applying domain knowledge for hypothesis generation.

Several other areas of future investigation were identified during the course of this research. They can be divided into three categories: evaluation measures, hypothesis ordering considerations, and interesting extensions of the basic operation of OXGate.

Evaluation measures

- An alternative concave measurement of node impurity has been suggested by Breiman *et al.* [1984]. Instead of using the entropy or information-gain of the node, they propose the computationally simpler method of forming the product of the positive and negative purities. Using this approach to estimate the quality

of a split simply involves calculating the difference of the node impurity and the weighted sum of the impurities of the branches. The formulas are identical to those of information-gain presented on page 75. As the basis of a cost-complexity measure, it may be useful in product combination with simplicity for both hypothesis evaluation as well as ordering.

- Another area for future work is the development of a practical and syntactically attractive approach to operator cost assignment for calculating hypothesis complexity/simplicity. In this thesis, the operators AND, OR, and NOT are used with costs of zero, while WHATIS and MEMBER have costs of one. Preliminary testing during the evolution of OXGate exhibited no noticeable, consistent benefit from assigning non-zero costs to the Boolean operators. However, as discussed in Section 4.2.2, constructive operators should be assigned costs commensurate with their computational difficulty.

- Section 3.1.3 discusses the use of the product of confirmation and simplicity as the basis for hypothesis evaluation. Theoretically, this approach should produce well-behaved decision-tree induction; however, early empirical investigations contradicted this expectation. This empirical disagreement with well-founded theory warrants continued investigation.

Hypothesis ordering considerations

- In the current implementation of hypothesis ordering, three dimensions were used for multiple-objective evaluation: confirmation, simplicity, and primitiveness. Other measures are also possible, such as the degree of confidence ascribed to a hypothesis by its domain knowledge-based generator, or the amount of combined support that disjoint knowledge sources provide a hypothesis. Past behavior is also a promising evaluation measure, using the history of the hypothesis on similar

concepts or domains, the behavior of the family of hypotheses to which it belongs, or the past utility of the operator(s) used to construct the hypothesis.

- Four methods of combining the evaluation measures were investigated in Chapters 4 and 5. The linearly weighted combination approach and the application of a modified non-domination method were the best-behaved. Other combination methods also warrant examination, such as model-based decision analysis, the use of fuzzy sets. or the Dempster-Schaeffer method of uncertain reasoning.

- Section 4.2.1.1 described several approaches to resolving the problem of the Quick-Look sample not adequately representing highly disjunctive concepts. Increasing the size of the sample improves the likelihood of representing the islands, but at a significant computational cost. One possible approach to avoid maintaining too large a Quick-Look sample is to adjust the sample size according to the concept dispersion or concept variation [Rendell and Seshu, 1990]. Such an approach requires the development of mechanisms for assessing the complexity of concept and dynamically scoping the sample size.

- Another approach for handling highly disjunctive concepts is to balance the training data (Section 5.4.4.3). While this approach appears to be an effective addition to hypothesis ordering, the effects of intentionally skewing the training data need further investigation.

- Even with concepts of low complexity, when using small sets of training examples, hypothesis ordering becomes very sensitive to the particular examples randomly selected (Sections 5.3 and 5.5.3). A mechanism to assess the behavior of hypothesis ordering and disable it for smaller training sets would alleviate this sensitivity.

- Section 5.4.4.1 demonstrated the benefit of retaining the full set of primitive hypotheses during induction. However, for a complex concept with a very rich initial description language, after several useful hypotheses have been constructed, there may be no need to retain many of the primitive hypotheses any longer. It may be beneficial to identify and eliminate the unneeded primitives.

Extensions of OXGate

- When selecting a hypothesis for incorporation, sometimes it is necessary to choose from among several equally or near equally credible hypotheses. This choice may have a substantial impact on the ability of OXGate to generate the proper hypotheses for good concept induction. One approach to avoid this sensitivity is to instill OXGate with the ability to entertain multiple world-views. Each world-view represents the state of the system after incorporating one of the hypotheses. After the hypothesis generation has proposed more hypotheses and induction has proceeded further, OXGate can commit to one of the world-views and discard the rest. This approach is a form of look-ahead: it is computationally expensive, but could potentially produce better decision-trees.

- The experiential learning program Koala (Chapter 6) is currently relatively inflexible with respect to the choice of patterns. The template building mechanism used by Koala does not allow structural "wildcarding" in the pattern: only content wildcards are allowed. It does not have CITRE's flexibility to pick a node for use as an operand regardless of its structure. Such a capability would provide OXGate with the ability to construct hypotheses of any level of complexity during experiential learning.

- The experiments of Section 5.4.4.2 show that retaining and reintroducing some of the hypotheses rejected by the hypothesis ordering mechanism may be

advantageous for creating accurate decision-trees. A promising alternative to recovering the rejected hypotheses is to rely on the hypothesis generator to regenerate certain ones when there is sufficient reason to do so. The hypothesis generator should be made intelligent enough to recognize when the decision-tree assembly has reached an impasse, and attempt alternative approaches to hypothesis generation such as the selective regeneration of previously rejected hypotheses.

- A user or developer interface to the hypothesis generator would provide the ability to modify easily the knowledge available to OXGate. The user could readily experiment with the effects of particular pieces of knowledge, apply knowledge incrementally to guide OXGate through induction of a particularly difficult concept, or provide other types of guidance to the system such as dynamically modifying the evaluation method and parameters. This interface would be a powerful addition to the capabilities of OXGate as a flexible development environment.

APPENDIX A

DEFINITIONS

Abduction: The act of producing a new hypothesis. The *extralogical* production of a general statement in the presence of concrete instances. [Watanabe, 1985]

Accuracy: The measure of predictive performance of the resultant classifier on a set of test data independent of the training data. It is the number of correctly identified examples divided by the total number of examples in the test set. [Breiman *et al.*, 1984] (See also *Resubstitution Accuracy.*)

Attribute: A variable describing a measurable property of an instance. An attribute may be Boolean (binary), integer, real, or nominal valued. (See also *Feature.*)

Bias: A preference for certain areas of hypothesis space over others [Mitchell, 1980, Utgoff, 1986]. Biases are necessary for effectively searching for a concept description. They can be *context-insensitive*, *context-independent*, or *context-sensitive* (Section 2.2.2).

Binarization: The process of converting a non-binary attribute to a single feature. The possible values of the attribute are grouped into two distinct subsets, thereby converting an n-way decision into a binary one. [Cestrik *et al.*, 1987]

Concept: An intensional description of a class of objects, i.e., a condensed (nonenumerated) description intended to identify members of a particular class and discriminate them from non-members. [Hunt *et al.*, 1966, Matheus, 1989]

Concept Dispersion: The degree of disjunction inherent in the representation of the concept in instance space. A complex concept has several distinct areas of positive

examples of the concept spread throughout instance space, making it difficult or impossible for a selective induction algorithm to learn the concept.

Confirmation: The degree of confidence placed in a hypothesis on the grounds of the empirical evidence.

Constructive Induction: A form of inductive concept learning in which new terms are generated to enhance the instance description language. [Dietterich and Michalski, 1983]

Credibility: The quality or inductive probability of a hypothesis, defined as the product of its confirmation and plausibility, divided by the prior probability of the evidence. [Watanabe, 1985]

Deduction: The act of drawing a conclusion from a set of declarations through logical reasoning.

Evidential, Extra-Evidential/Extralogical: Factors affecting the evaluation of a hypothesis. Evidential factors are based on empirical evidence (training data). Extra-evidential/extralogical factors are based on aspects other than evidence, including simplicity, elegance, harmony with a larger theoretical structure, and relationship to other competing hypotheses. [Watanabe, 1985]

Feature: According to Breiman *et al.* [1984], a feature is a real-valued variable manufactured from the function of the measured variables. Matheus [1989], Michalski [1983] and other sources use *feature* loosely, equating it to *attribute* as a variable with many possible values. In this thesis, a feature is defined as a special case of attribute whose value can be only *true* or *false*, i.e., limited to a Boolean-valued variable (Section B.2).

Feature Construction: The application of constructive operators to existing features resulting in the definition of one or more new features. [Matheus, 1989]

Generalization: The modification of a feature or hypothesis to make it less specific to the training instances, with the intent of making it more applicable to the true concept.

Hypothesis: A statement of arbitrary complexity, defined as a function of terms from the original instance description language. A hypothesis must be *testable*: it can be determined to be "true" or "false" for a given instance by instantiating the ground attributes in the hypothesis description with the values present in the instance data. (See also *Primitive Hypothesis.*)

Hypothesis Evaluation: Determination of the credibility of a hypothesis by testing against the full set of available training data.

Hypothesis Generation: The creation and proposition of candidate hypotheses for use in inductive learning, i.e., the proposition of the original set of primitive hypotheses, and the construction of new terms.

Hypothesis Incorporation: Acceptance of a hypothesis as a portion of the concept description. In decision-tree induction, incorporation involves adding the hypothesis as a decision node in the tree, sorting the training data according to the hypothesis, making the new state of the decision-tree available to the rest of OXGate, and determining if the learning task has been satisfactorily completed.

Hypothesis Ordering: The heuristic means of constraining the number of hypotheses passed along for hypothesis evaluation. Hypothesis ordering is used to identify the most promising of the candidate hypotheses, and to reject or suppress those deemed useless.

Induction: The act of proposing a general conclusion from a limited set of evidence.

Inductive Concept Learning, Inductive Learning: Learning a concept description from a set of training examples, with the intent of finding a sufficiently general description to effectively predict the classification of previously unseen examples.

Incomplete Instance Description Language: A language is *incomplete* when the set of attributes is insufficient to completely discriminate between noise-free cases with different outcomes [Michalski *et al.*, 1986]. In this situation, attributes constructed from this language would also be insufficient to overcome the deficiency.

Koala: The experiential learning procedure used in OXGate. Koala peruses a completed decision-tree looking for hypotheses with which to construct new hypotheses. The name *Koala* reflects the behavior of the procedure: it climbs the entire tree, but is particular about selecting only certain branches leading to the choicest leaves.

Multiple-Objective Evaluation/Multiple-Criteria Decision Making: The act of formulating a decision based on several evaluation measures or criteria. [Yu, 1985]

Non-Dominance, Non-Dominated Hypotheses: A modified method of multiple-criteria decision making applied to hypothesis ordering in which the preferred hypotheses are those not substantially surpassed in every evaluation dimension by another hypothesis.

Opportunistic: Taking advantage of the current situation to further an existing goal. In a system that reasons opportunistically, the determination of which knowledge to apply is made dynamically, one step at a time, resulting in the incremental generation of partial solutions. The choice of the knowledge to apply is based on the current state of the solution. [Barr *et al.*, 1989]

Opportunistic Constructive Induction: Concept induction that applies knowledge at any opportune time during the inductive process to create new terms for the description language. (See also *Opportunistic* and *Constructive Induction.*)

OXGate: OXGate is the name of the implementation developed for this thesis. The name is derived from a term used in the science fiction novel *The Jesus Incident* by Larry Niven and Jerry Pournelle. In the novel, the Ox gate is a mysterious port in a shipboard computer system from where deeply archived information and well-kept secrets periodically emerge. From the point of view of the hypothesis evaluation mechanism in OXGate, generally useful hypotheses simply appear on the blackboard with no indication of their origin; hence, they emerge from the "Ox gate."

Parity Problem: The general situation exemplified by the exclusive-or (XOR) operator: no attribute involved in the parity situation provides a means of discriminating between positive and negative examples. Either blind guessing, look-ahead, or feature construction is required to proceed with concept induction. [Seshu, 1989]

Post-Pruning: A method of generalizing a completed decision-tree by removing decision nodes which are statistically only weakly supported by the training data. [Breiman *et al.*, 1984]

Predictive Accuracy: (See *Accuracy*)

Pre-Pruning: A method of generalizing a decision-tree during assembly by performing a statistical evaluation of the expected error of adding a particular decision node, comparing it to the estimated error of not adding the decision node, and adding the node if appropriate. [Breiman *et al.*, 1984]

Primary Subset: An output of the hypothesis ordering component: the portion of the pool of currently available hypotheses appearing to be the most promising for further decision-tree induction.

Primitive Hypothesis: A primitive hypothesis is an attribute-value pair in which the attribute is a member of the original instance description language. Only the primitive hypotheses are operational: they are the only hypotheses directly testable against the data.

Prior Probability: The prior (a priori) probability of a hypothesis is interpreted as the extra-evidential likelihood of the hypothesis. It is the degree of confidence attached to the hypothesis on the grounds of considerations outside the empirical data. [Watanabe, 1985]

Quick-Look: A heuristic method of estimating the confirmation of hypotheses by testing against a small random subset of the training examples. It is related to hypothesis filtering [Etzioni, 1988] and, very loosely, the use of windowing in ID3 [Wirth and Catlett, 1988].

Rejected Subset: An output of the hypothesis ordering component: the portion of the pool of currently available hypotheses appearing to be useless for current and future decision-tree induction.

Replication Problem: The recurrence of portions of the concept description (subtrees) throughout the decision-tree (Figure 2.8). When these subtrees are identified, they can be added to the description language as constructed hypotheses, yielding a more concise and accurate decision-tree in subsequent induction. [Pagallo and Haussler, 1989]

Resubstitution, Resubstitution Accuracy: *Resubstitution* refers to testing the classifier on the training data. The *resubstitution accuracy* is the percentage of training instances classified properly. *Maximal resubstitution accuracy* implies that 100% correct classification of the training data is not always possible due to a noisy data set or an incomplete instance description language. [Breiman *et al.*, 1984]

Secondary Subset: An output of the hypothesis ordering component: the portion of the pool of currently available hypotheses left over after removal of the primary and rejected subsets. It represents the set of hypotheses potentially useful later during the decision-tree assembly.

Selective Induction: Inductive concept learning using only the original instance description language to formulate the concept description, without the benefit of constructive operators.

APPENDIX B

HYPOTHESIS REPRESENTATION

The decision-tree construction process underlying OXGate proceeds by the selection and piecewise incorporation of *testable* hypotheses, statements which when tested against an instance are either "true" or "false."[1] Regardless of how complex constructed hypotheses become, as long as their evaluation on an instance results in the determination of their truth or falsehood, they can be used by OXGate for assembly of the decision-tree. Therefore, the decision-tree produced by OXGate is always a binary tree.

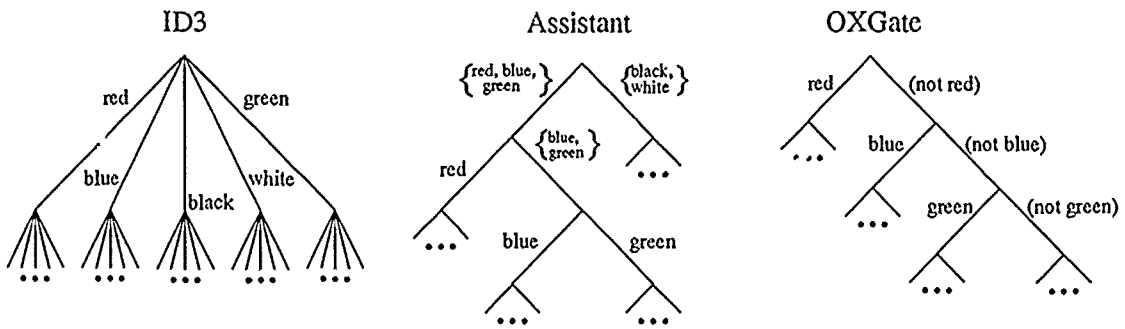


Figure B.1 Contrast of Three Representations. For nominal or discretized numerical scales, such as the attribute *color*, ID3 performs n-way splitting across an attribute, Assistant performs binarization, and OXGate produces a strictly binary decision-tree representation.

The binary tree of OXGate is an alternative representation to the n-ary trees produced by ID3 (Figure B.1). In the case of binary attributes, the resultant trees will be identical.

[1] In OXGate the "don't know" condition on a tested attribute is treated as false. Other approaches could be used, such as assigning conditional probabilities to each possible value of the attribute (Assistant86 [Cestrik *et al.*, 1987]) or simply selecting the most probable value (CN2 [Clark and Niblett, 1987]). In addition, if an instance cannot be identified as a positive example of the concept, OXGate considers it to be a negative example. Other approaches such as the Bayesian classification in Assistant86 could be incorporated into OXGate, but the exact treatment of unknowns is not germane to the theme of this research and is not addressed. See [Quinlan, 1989] for a discussion of approaches.

For nominal attributes of arity greater than two, the preference of one representation over the other is a function of the particular concept being described (Section B.1). For other attribute scales, such as real numbers, the superiority of one representation over the other is subject to interpretation: to use these scales, ID3 must receive from some external source a translation of the scales in a discretized (nominal) form. OXGate can use the real-valued scales directly by postulating ranges over the values or other mathematical functions, but, in essence, this requires the same process of finding suitable breakpoints in the original scale and transforming the representation to a usable form. In both systems, this change of representation clouds direct comparisons between OXGate and ID3 on the strengths of the decision-tree representations alone.

Assistant and its descendant Assistant86 [Cestrik *et al.*, 1987], derivatives of the ID3 family of decision-tree learning systems, also use a binary decision-tree representation. Assistant uses a process called *binarization* to group the possible values of the attribute being considered at a particular decision point into two distinct subsets, thereby converting an n-way decision into a binary one (Figure B.1). For real-valued attributes, Assistant preprocesses the scales into a finite number of subintervals and uses a heuristic method to avoid testing all possible groupings. This approach is abstractly equivalent to that of OXGate, although OXGate retains the distinct advantage of being able to accept suggestions by the knowledge sources on possible approaches to the partitioning and grouping functions. With nominal scales, Assistant always performs binarization, heuristically seeking the most useful subsets of values. In OXGate, the nominal values could be grouped into subsets provided some knowledge or impetus exists to do so, but its basic mode of operation is to use simple attribute-value pairs as individual decision points. This type of "fine-grained binarization" used by OXGate is also found in CITRE [Matheus, 1989], STAGGER [Schlimmer, 1987], and the original family of Concept Learning Systems (CLS) [Hunt *et al.*, 1966].

B.1 Functional Equivalence and Justification

One of the obvious criticisms OXGate faces is the claim that attributes may need to be evaluated more than once during the classification of a test instance while ID3 evaluates each attribute at most once. While on the surface this seems a valid criticism, it loses its potency as an argument against OXGate when one considers the implementation details involved in acquiring the value of an attribute for a specific instance, storing the value in a readily accessible location, and comparing the value against the decision point conditionals until the match is found. It would be a simple matter to equip OXGate with the ability to cache the test results for future use if speed was a primary concern.

For the attribute *color* in Figure B.1, ID3 tests the five values for the attribute and is finished with it. Assistant tests five attribute-value pairs at the top level, three at the second level and two at the third level for a total of ten tests. OXGate tests five attribute-value pairs at the top level, four at the second level, and three at the third level for a total of twelve tests. Although the binary representation of Assistant results in fewer tests than in OXGate, Assistant must consider fifteen unique subset combinations of the tested values at the top level, seven combinations at the second level, and three combinations at the third. The computational complexity of determining the best subsets may outweigh the cost of the additional tests incurred by OXGate.

Although the representation used in ID3 appears superior to the binary representations in the previous example, the algorithmic and decision-tree complexities of the three approaches vary with the concept. At times, the binary decision-tree representation is superior to the ID3 decision-tree. In the example shown in Figure B.2, the Assistant and OXGate representations are equivalent, and neither suffers the problem of the replication of subtrees shown by ID3 or its associated lack of understandability [Pagallo and Haussler, 1989]. The average number of tests needed to classify an example (2.167), as well as the worst-case number of tests (3), is identical across representations.

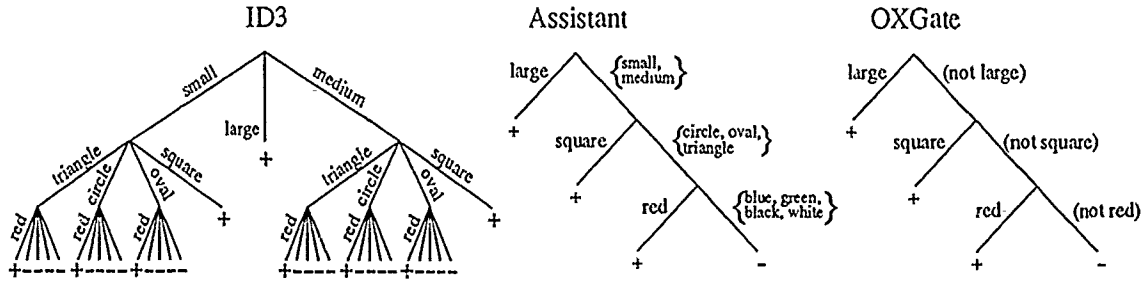


Figure B.2 Comparison of Decision-Trees. The structured splitting method of ID3 (mandatory n-way splitting) puts it at a distinct disadvantage for the concept $\big((color = red)$ OR $(size = large)$ OR $(shape = square)\big)$.

The preference of one decision-tree representation over another depends on the complexity of the concept and domain being considered, as well as the choice of metrics for evaluating tree quality (e.g., comprehensibility, lack of replication, average number of tests during classification). Section 5.2 presents experiments to ascertain if either the ID3 or OXGate decision-tree representations possess an intrinsic representational advantage over the other. Several artificial and two real-world concepts were used for the tests. The binary representation used in OXGate provides neither generally better nor worse intrinsic performance than the ID3 representation, yet, in allowing the flexible incorporation of generated hypotheses, it provides the foundation for the opportunistic use of domain knowledge.

B.2 Representation Formalism

In this section, a formalism and notation is presented to establish a frame of reference for describing the application of biases and knowledge in Section 6.1. This section presents the development of a feature-based description language from an original attribute-based language, and concludes with a description of decision-trees as a set of binary relations.

B.2.1 Attributes and features

Let $\tau(Q)$ be defined as a test on Q (an attribute, feature, class, or leaf node) to determine the value of its instantiation. The original attribute-based description language $\mathcal{A}$ is a set of original or *ground* attributes, and is defined by

attribute a_i, $i = 1, 2, \ldots$,

attribute-based description language $\mathcal{A} = \{a_i\}$,

j^{th} value of attribute a_i : $v_i^{(j)}$, $j = 1, 2, \ldots$,

and the set of values of attribute a_i : $\mathcal{V}_i = \left\{v_i^{(j)}\right\}$ (i.e., $\tau(a_i) \in \mathcal{V}_i$)

To convert the attribute-based description language to a feature-based language suitable for the assembly of the fine-grained binary decision-trees used by OXGate, create for each nominal attribute a_i a set of primitive features (primitive hypotheses) $\mathcal{X}_i = \left\{x_i^{(j)}\right\}$ such that there is a primitive feature $x_i^{(j)}$ for every possible attribute-value pair:

$$x_i^{(j)} \triangleq \left(\tau(a_i) = v_i^{(j)}\right), \quad \text{where } \tau\left(x_i^{(j)}\right) \in \{T, F\}$$

For numerical-valued attributes (real or integer), it is usually disadvantageous to attempt to create a feature for each possible value of the attribute. Binarization is accomplished by selecting ranges of values, and the feature becomes a test whether the instantiation of the attribute falls within the range. For these attributes a_i, create a set of primitive features $\mathcal{X}_i = \left\{x_i^{(k)}\right\}$:

primitive feature $x_i^{(k)} \triangleq \left(\tau(a_i) \in g_k(\mathcal{V}_i)\right)$, where

$g_k(\mathcal{V}_i) \subset \mathcal{V}_i$ (defined with $\leq, <, >, \geq$, etc.),

$\bigcup_k g_k(\mathcal{V}_i) = \mathcal{V}_i$, $\tau\left(x_i^{(k)}\right) \in \{T, F\}$, and

$\forall m \forall n \left[(m \neq n) \Rightarrow \left(g_m(\mathcal{V}_i) \cap g_n(\mathcal{V}_i) = \emptyset\right)\right]$

This binarization of the attributes defines a feature-based description language $\mathcal{X} = \cup_i \mathcal{X}_i$. Essentially, a feature is defined as a test (*true* or *false*) to determine whether the instantiation of a particular attribute is a member of a particular subset of possible values. This approach also applies to the binarization method of Assistant on nominal-valued attributes with two exceptions: the relational operators defining which subsets of $\mathcal{V}_i$ constitute $g_k(\mathcal{V}_i)$ are not applicable to nominal-valued attributes, and the requirement for disjoint subsets is unnecessary.

B.2.2 Decision-trees

For inductive concept learning of a decision-tree, a set of training instances $\mathcal{I}_c$ is provided, where each instance $I_p \in \mathcal{I}_c$ is described by an attribute vector $\vec{a} = (a_1, a_2, \ldots, a_n)$, corresponding to the attribute-based description language $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$. The vector $\vec{a}$ can be recast as a feature vector $\vec{x} = \left(x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(j_1)}, \ldots, x_n^{(1)}, x_n^{(2)}, \ldots, x_n^{(j_n)}\right)$ corresponding to the feature-based description language $\mathcal{X} = \left\{x_1^{(1)}, \ldots, x_n^{(j_n)}\right\}$. To simplify the notation, $\left\{x_1^{(1)}, \ldots, x_n^{(j_n)}\right\}$ shall be mapped one-to-one onto $\{x_1, \ldots, x_m\}$ with the understanding that this description language $\mathcal{X} = \{x_1, \ldots, x_m\}$ is derived from the original attribute language in the manner presented in the previous section.

Each instance I_p can then be described by a tuple $(\vec{x}_p, \varphi_p)$ in which $\vec{x}_p$ is the instantiation of the feature vector $\vec{x}$ for I_p, and φ_p is its classification, i.e., $\varphi_p \in \{+, -\}$.

The result of inductive concept learning under the conditions described above is a binary decision-tree $\mathcal{T}(\mathcal{X})$ over the feature-based language $\mathcal{X}$. To describe the tree, first let $\mathcal{X}_c$ be the subset of features used as decision points in the construction of $\mathcal{T}(\mathcal{X})$ for the training instances $\mathcal{I}_c$:

$$\mathcal{X}_c \subset \mathcal{X} \text{ such that } \mathcal{T}(\mathcal{X}_c) = \mathcal{T}(\mathcal{X})$$

An individual feature $x_i \in \mathcal{X}_c$ may be used multiple times in the tree. Let a node be defined by using a feature as a decision, and let $\mathcal{N}_c$ be the set of nodes forming the tree $\mathcal{T}(\mathcal{X}_c)$. The mapping from $\mathcal{N}_c$ to $\mathcal{X}_c$ is surjective, and not-injective: a node maps to a single feature, but a feature may be mapped onto by multiple nodes. The relationship between a node n_α and its associated feature x_i is depicted as

$$n_\alpha \mapsto x_i$$

Also, let $\mathcal{Z}_c$ be the set of termini (leaf nodes) in $\mathcal{T}(\mathcal{X}_c)$:

$$\mathcal{Z}_c = \{z_1, z_2, \ldots\} \quad \text{and} \quad \tau(z_i) \in \{+, -\}$$

Then, the tree $\mathcal{T}(\mathcal{X}_c)$ may be described by the set of nodes $\mathcal{Y}_c = \mathcal{N}_c \cup \mathcal{Z}_c$ and a set of binary relations (edges) $\mathcal{R}_c$ over $\mathcal{Y}_c$ providing directed connections between the nodes.

B.2.3 Feature construction

Section 6.1 describes the use of learned decision-trees as the basis for feature construction when learning from experience. FRINGE and CITRE both construct new features as binary conjunctions of operands formed from existing features, described below. The selection of the features to use for construction is addressed in Section B.3.

Typically, the notation used for a relation in $\mathcal{R}_c$ is $\left(\langle n_a, n_b \rangle \in \mathcal{R}_c\right)$, indicating that a directed edge exists from node n_a to n_b. For the purpose of using the information contained in the structure of the decision-tree $\mathcal{T}(\mathcal{X}_c)$ to select features for use in feature construction, it is necessary to distinguish between the left and right branches emanating from a decision node. The relations in $\mathcal{R}_c$ cannot be described simply as $\left(\langle n_a, n_b \rangle \in \mathcal{R}_c\right)$ without taking into account the result of the test of the feature at that node. Therefore,

define $\mathcal{R}_c = \mathcal{R}_l \cup \mathcal{R}_r$ (union of the left and right branches), such that for $n_a \mapsto x_a$

$$\begin{aligned}
&\left(\langle n_a, n_b \rangle \in \mathcal{R}_l\right) \Rightarrow \left(\tau(x_a) = T\right), \\
&\left(\langle n_a, n_b \rangle \in \mathcal{R}_r\right) \Rightarrow \left(\tau(x_a) = F\right), \\
&\text{and } \mathcal{R}_l \cap \mathcal{R}_r = \emptyset
\end{aligned}$$

Also, define $\Omega(x_a)$ to be an operand for the construction of a new feature under the conditions $\langle n_a, n_b \rangle \in \mathcal{R}_c$ and $n_a \mapsto x_a$. Where the relation is a left branch from the decision, the operand is the feature x_a. Where the relation is a right branch, the negation of the feature, $\overline{x_a}$, is used as the operand.

$$\Omega(x_a) = \begin{cases} x_a & \text{for } \langle n_a, n_b \rangle \in \mathcal{R}_l, \\ \overline{x_a} & \text{for } \langle n_a, n_b \rangle \in \mathcal{R}_r \end{cases} \tag{B.1}$$

General feature construction [Matheus, 1989] consists of the application of a constructive operator O_i to a set of operands to create a new feature x^*:

$$x^* = O_i\left(\Omega(x_j), \ldots\right) \tag{B.2}$$

The applications discussed in Section 6.1 (FRINGE, CITRE, and OXGate/Koala) use a learned decision-tree as the basis for selection of operands for feature construction. Binary conjunction is the only constructive operator (O_i) considered. As Matheus [1989] points out, the iterative application of binary conjunction in concert with the negation implicit in the nodes of a binary decision-tree (and incorporated into the operands) yields a complete Boolean representational capability. Generalization operators are also applied in CITRE and OXGate, but only after the binary conjunctions are created. Therefore, for these applications, the set of new features $\mathcal{X}_c^*$ derivable from the decision-tree $\mathcal{T}(\mathcal{X}_c)$

in a single iteration is

$$\mathcal{X}_c^* = \left\{ \left(\Omega(x_a) \wedge \Omega(x_b) \right) \mid \mathcal{T}(\mathcal{X}_c),\ x_a \in \mathcal{X}_c,\ x_b \in \mathcal{X}_c,\ a \neq b \right\}$$

An additional general limitation used by these applications is that not all possible combinations of operands are used for new feature construction. Instead, operands are selected only from paths in the decision-tree leading from the root node $n_\odot$ to positive leaf nodes. The root node $n_\odot$ is defined as the only node in $\mathcal{T}(\mathcal{X}_c)$ that is not on the subordinate end of a relationship with any other node in the tree. The feature $x_\odot$ used at the root node $(n_\odot \mapsto x_\odot)$ is defined by

$$x_\odot \triangleq \left(x_j \mid \neg\exists x_i \left[\left(\langle n_i, n_j \rangle \in \mathcal{R}_c \right) \cap (n_i \mapsto x_i) \cap (n_j \mapsto x_j) \cap (x_i \in \mathcal{X}_c) \cap (x_j \in \mathcal{X}_c) \right] \right)$$

The set of all positive leaves (+ termini) in $\mathcal{T}(\mathcal{X}_c)$ is

$$\mathcal{Z}_c^+ = \{ z_i \mid z_i \in \mathcal{Z}_c, \tau(z_i) = + \}$$

A single path leading from the root $x_\odot$ to a positive leaf z_q is a subtree $\mathcal{T}(\mathcal{X}_q)$ of $\mathcal{T}(\mathcal{X}_c)$ defined by nodes $\mathcal{N}_q$ where

$$\mathcal{N}_q \mapsto \mathcal{X}_q,\ \ \mathcal{X}_q \subset \mathcal{X}_c \text{ (includes } x_\odot),\ \ \mathcal{Y}_q = \mathcal{N}_q \cup \{z_q\},$$
$$\mathcal{R}_{q_l} \subset \mathcal{R}_l,\ \ \mathcal{R}_{q_r} \subset \mathcal{R}_r,\ \text{ and } \ \mathcal{R}_q = \mathcal{R}_{q_l} \cup \mathcal{R}_{q_r}$$

The set of features $\mathcal{X}_q$ is the subset of features from $\mathcal{X}_c$ that are used as decision points along the path from the root node to the positive leaf z_q. Therefore, the set of new features possible through construction using the operands presented in subtree $\mathcal{T}(\mathcal{X}_q)$ is

$$\mathcal{X}_q^* = \left\{ \left(\Omega(x_a) \wedge \Omega(x_b) \right) \mid \mathcal{T}(\mathcal{X}_q),\ x_a \in \mathcal{X}_q,\ x_b \in \mathcal{X}_q,\ a \neq b \right\} \qquad \text{(B.3)}$$

The complete set of new features is the union of new features constructed from each of the paths: $\mathcal{T}(\mathcal{X}_c)$ is

$$\mathcal{X}_c^* = \bigcup_q \mathcal{X}_q^*$$

B.2.4 Hypothesis construction

The construction of general features defined in Equation (B.2) is a limited form of construction. Each of the operands can return only the value of "true" or "false" when tested, limiting the types of operator and the power of the construction. When the original attributes are themselves Boolean features, this restriction is not an issue. Constructions using nominal-valued attributes are also not limited by this restriction since the nature of the attribute limits the types of operators applicable. However, this approach to construction severely limits the potential available with real- and integer-valued attributes. For example, using the illustration shown in Figure 1.1 (page 3), no amount of feature construction could create the feature $(P \times V = C)$ from the initial feature set $\{p_1, p_2, \ldots, p_n, v_1, v_2, \ldots, v_m\}$ where the features p_i and v_j represent tests for ranges of pressure and volume, respectively. The attributes themselves must be combined into a new attribute, and then a useful hypothesis (in this case the true concept description) can be created testing the new attribute against a set or range of values. Hypothesis construction of this form can be described as creating a new attribute a^* and a new hypothesis h^* by

$$a^* \in \mathcal{F}\left(\mathcal{A}'\right), \quad \text{and} \quad h^* \triangleq \left(\tau(a^*) \in g\left(\tilde{\mathcal{F}}(\mathcal{V}')\right)\right)$$

where $\mathcal{A}'$ is the set of all attributes including previously constructed ones, $\mathcal{F}$ is a function or set of operators applied to $\mathcal{A}'$, and $\tilde{\mathcal{F}}(\mathcal{V}')$ is the possible range of values corresponding to the new attribute (typically $\tilde{\mathcal{F}}$ is the same function as $\mathcal{F}$, but is applied to the ranges of the attributes). Each new attribute a^* is one of several attributes potentially created

by applying the set of operators $\mathcal{F}$ to the set of existing attributes $\mathcal{A}'$. A new hypothesis is a test to determine if the instantiation of the new attribute (the test τ on a^*) is an element of a subset of the new values.

A constructed hypothesis is essentially a feature of arbitrary complexity. Hypothesis construction subsumes feature construction (Equation (B.2)) since features are simply Boolean-valued attributes. The general form of a constructed hypothesis is a test whether the instantiation of a certain function of attributes is related in some fashion to a specific subset of values, and can be defined by

$$h^* \triangleq \left(\tau\left(\mathcal{F}(\mathcal{A}')\right) \overset{Rel}{\Longleftrightarrow} g\left(\tilde{\mathcal{F}}(\mathcal{V}')\right)\right) \tag{B.4}$$

Equation (B.4) is very general and necessarily noncommittal. It encompasses a wide variety of statements: in fact, any statement that is ultimately a function of only the ground attributes. The reason it needs to be so general is illustrated below with a few short examples. For ease of understanding the examples here and throughout the thesis, the form $(A = B)$ will be used as shorthand for $\left(\tau(A) = B\right)$, both meaning "test if attribute A has value B."

1. Let two features be $x_1 = (width = 10)$ and $x_2 = (length = 10)$, corresponding to $a_1 = width$, $v_1 = 10$, $a_2 = length$, and $v_2 = 10$. One possible construction would be the application of the addition operator, resulting in a new attribute $a_3 = (a_1 + a_2) = (width + length)$, and a new hypothesis $h_3 = \left((width + length) = 20\right)$. This hypothesis has the form $\left(\tau\left(\mathcal{F}(a_1, a_2)\right) = \tilde{\mathcal{F}}(v_1, v_2)\right)$.

2. Using the same two features with their corresponding attributes and values, another reasonable construction would be $h_3 = \left((width = length) = T\right)$ entailing a

new Boolean attribute $a_3 = (width = length)$.[2] This hypothesis has the form $\Big(\tau\big(\mathcal{F}(a_1, a_2)\big) = T\Big)$, or more simply, $\mathcal{F}(a_1, a_2)$.

3. Now consider two features $x_4 = (color = red)$ and $x_5 = (shape = oval)$. In this case it would not make sense to combine the attributes *color* and *shape*. Construction involves combinations of the features, such as the hypothesis $h_6 = \big((color = red)$ AND $(shape = oval)\big)$. This hypothesis has the form $\mathcal{F}(x_4, x_5)$, equivalent to $\Big(\tau\big(\mathcal{F}(x_4, x_5)\big) = T\Big)$. This approach to hypothesis construction is equivalent to the non-generalizing feature construction used in FRINGE, CITRE, and Koala.

4. From the preceding components, a more complex hypothesis can be constructed: $h_7 = \Big((color = red)$ AND $(shape = oval)$ AND $\big((width + length) = 20\big)\Big)$. This hypothesis takes the general form of Equation (B.4) where a complex function of the ground attributes is related in some fashion to a complex function of values.

5. Another type of hypothesis that takes the form of Equation (B.4) is the generalization created through the collection of disjoint terms. For example, the hypothesis $\Big((shape = rect.)$ AND $\big(color \in \{red, white, blue\}\big)\Big)$ is a generalization of $\big((shape = rect.)$ AND $(color = red)\big)$, $\big((shape = rect.)$ AND $(color = white)\big)$, and $\big((shape = rect.)$ AND $(color = blue)\big)$. This approach to generalization is implemented in Koala through the use of the MEMBER operator.

[2]Note the equivalence of $((width = length) = T)$ and $(width = length)$. Both are statements with identical meanings. This example illustrates one of the difficulties of describing exactly what a feature is, for if A is a Boolean attribute, then A, $(A = T)$, $((A = T) = T)$, $(((A = T) = T) = T)$,... are all equivalent statements. By the definitions used herein, all are hypotheses, but only A is a feature.

B.3 Applying Biases and Knowledge

Section 6.1 describes the application of biases and knowledge to select operands for constructing new features (FRINGE and CITRE) or hypotheses (OXGate/Koala) from a learned decision-tree. The biases incorporated in FRINGE and CITRE provide syntactic guidance, specifying locations in the tree from which to choose operands. CITRE and Koala use knowledge to impose other constraints, specifying the nature of allowed operands. This section parallels Section 6.1 and presents the application of biases and knowledge as constraints on the set of constructions possible from the decision-tree.

FRINGE uses a single bias for operand selection: *fringe*. CITRE is able to use one of five biases: *fringe, root, root-fringe, adjacent,* and *none*. These biases are described in Section 6.1. Using the notation developed in Section B.2, it is easy to show how the application of these biases refines the selection of operands from within a subtree $\mathcal{T}(\mathcal{X}_q)$, the path from the root node $n_{\textcircled{1}}$ to positive leaf z_q. Each bias can be viewed as a filtering operation by the imposition of additional constraints on the construction of a new feature $\mathcal{X}_q^*$ (defined in Equation (B.3) and repeated here)

$$\mathcal{X}_q^* = \left\{ \Big(\Omega(x_a) \wedge \Omega(x_b) \Big) \mid \mathcal{T}(\mathcal{X}_q),\ x_a \in \mathcal{X}_q,\ x_b \in \mathcal{X}_q,\ a \neq b \right\} \tag{B.5}$$

where x_a and x_b are the two features used at the selected decision points in the path, and $\Omega(x_a)$ and $\Omega(x_b)$ are the two operands derived from those features according to Equation (B.1).

One example of this filtering operation is the application of the *fringe* bias, imposing the additional constraints of

$$\langle n_a, n_b \rangle \in \mathcal{R}_q,\ \ \langle n_b, z_q \rangle \in \mathcal{R}_q,\ \ n_a \mapsto x_a,\ \ n_b \mapsto x_b,$$
$$\text{with}\ \ z_q \in \mathcal{Z}_c^+ \cap \mathcal{Y}_q\ \ \text{(i.e., the only positive leaf node in } \mathcal{T}(\mathcal{X}_q))$$

to those already incorporated in the subtree. This results in the full definition of new feature $\mathcal{X}_q^*$ as

$$\overset{fringe}{\mathcal{X}_q^*} = \left\{ \begin{array}{l|l} \Big(\Omega(x_a) \wedge \Omega(x_b)\Big) & \mathcal{T}(\mathcal{X}_q),\ x_a \in \mathcal{X}_q,\ x_b \in \mathcal{X}_q,\ a \neq b, \\ & \langle n_a, n_b \rangle \in \mathcal{R}_q,\ \langle n_b, z_q \rangle \in \mathcal{R}_q, \\ & n_a \mapsto x_a,\ n_b \mapsto x_b,\ z_q \in \mathcal{Z}_c^+ \cap \mathcal{Y}_q \end{array} \right\}$$

Similarly, the *adjacent* bias imposes additional constraints to $\mathcal{X}_q^*$ of

$$\langle n_a, n_b \rangle \in \mathcal{R}_q,\ \langle n_b, y_i \rangle \in \mathcal{R}_q,\ \text{and}\ \ y_i \in \mathcal{Y}_q$$

In addition, the *root* bias is identical to the *adjacent* bias with the restriction that the most superior node of the two relations be $n_\odot$:

$$\langle n_\odot, n_a \rangle \in \mathcal{R}_q,\ \langle n_a, y_i \rangle \in \mathcal{R}_q,\ \text{and}\ \ y_i \in \mathcal{Y}_q$$

Finally, the *root-fringe* bias demands that

$$\langle n_\odot, n_a \rangle \in \mathcal{R}_q,\ \langle n_b, z_q \rangle \in \mathcal{R}_q,\ n_b \neq n_\odot,\ n_a \neq z_q,\ \text{and}\ \ z_q \in \mathcal{Z}_c^+ \cap \mathcal{Y}_q$$

The biases just described are implemented as filters that act on the positions of the features within the tree. Knowledge may also be applied as filters, but in addition to being concerned with the position of features (syntactic filtering), knowledge-based filters may also be concerned with the meaning of the terms (semantic filtering) or their relationship to the domain (contextual filtering).

Figure 6.1 (page 156) illustrates a possible portion of a learned decision-tree for the NetTalk Silent concept. The given attributes are the seven characters (C_1 through C_7,

in order)[3] composing the window used to examine a word fragment. In accordance with the notation of Section B.2.1, this domain is described by

$$\begin{aligned}
&\mathcal{A} = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}\,, \\
&\mathcal{V}_i = \{a, b, c, \ldots y, z, _\} \quad \text{for} \quad i = 1 \ldots 7, \\
&x_i^{(k)} \triangleq \left(\tau(C_i) = v_i^{(k)}\right), \\
&\text{and} \quad \mathcal{X}_i = \left\{x_i^{(k)}\right\} \quad \text{for all } k
\end{aligned}$$

The knowledge used to guide the pairings in Figure 6.1 can be stated as: *the center character is the focus of attention, and the characters adjacent to the center provide the most information-gain* [Lucassen and Mercer, 1984]. This knowledge focuses the selection of features for construction on those related to the attribute pairs (C_4, C_3) and (C_4, C_5). The knowledge can be described as additional constraints on $\mathcal{X}_q^*$ (Equation (B.3)) of

$$\begin{aligned}
&x_a \in \mathcal{X}_4, \quad x_b \in \mathcal{X}_3 \cap \mathcal{X}_5, \\
&\langle n_a, y_\gamma \rangle \in \mathcal{R}_q, \quad n_a \mapsto x_a, \\
&\langle n_b, y_\delta \rangle \in \mathcal{R}_q, \quad n_b \mapsto x_b, \\
&y_\gamma \in \mathcal{Y}_q, \text{ and } \quad y_\delta \in \mathcal{Y}_q
\end{aligned}$$

resulting in new hypotheses of the form found in Example 3 on page 203.

The implementation of the experiential learning mechanism in OXGate is the procedure Koala. Koala searches for all pairings of *true* hypotheses leading to positive leaf nodes, but keeps only those defined by knowledge (such as that described above) as coded in search patterns. The acceptance of true hypotheses only, as opposed to their negations, embodies the additional constraints on $\mathcal{X}_q^*$ of $\langle n_a, y_\gamma \rangle \in \mathcal{R}_{q_l}$ and $\langle n_b, y_\delta \rangle \in \mathcal{R}_{q_l}$, which define the operand $\Omega(x_i)$ to equal x_i for all i.

[3]For notational convenience in relating to the other discussions of NetTalk throughout this thesis, the terms CX and C_X are assumed synonymous, i.e., $C1$ is identical to C_1, and so forth.

APPENDIX C

ARTIFICIAL BOOLEAN FUNCTIONS

```
CONCEPT: (COND ((AND (EQP 'X10 'T)
                     (EQP 'X5 'T)
                     (EQP 'X11 'T)))
               ((AND (EQP 'X2 'F)
                     (EQP 'X7 'T)
                     (EQP 'X6 'F)))
               ((AND (EQP 'X1 'T)
                     (EQP 'X3 'F)
                     (EQP 'X12 'T))))
```

DOMAIN: (X1 (T F)), (X2 (T F)), (X3 (T F)), (X4 (T F)),
(X5 (T F)), (X6 (T F)), (X7 (T F)), (X8 (T F)),
(X9 (T F)), (X10 (T F)), (X11 (T F)), (X12 (T F))

MEANING: $((X10 \wedge X5 \wedge X11) \vee (\overline{X2} \wedge X7 \wedge \overline{X6}) \vee (X1 \wedge \overline{X3} \wedge X12)$

Instance space size = 4096 instances
Coverage = 33% positive
24 primitive hypotheses available

Figure C.1 Typical Boolean 3-term 3DNF Concept.

This appendix presents the artificially created concepts with Boolean-valued attributes. In the experiments of Chapter 5, the training data used are random subsets of the instance space and the test data comprise the set of all examples in the instance space. Each run within an experiment uses a different randomly generated concept. Figure C.1 describes[1] a typical 3-term 3DNF concept: a concept with three terms joined together by disjuncts, with each term consisting of the conjunction of three primitive hypotheses.

[1] In the figures, the concepts are described in CommonLisp where COND represents the "if-then-else-if" function and EQP is a test of whether the value of the attribute for a considered instance equals the given value.

Figure C.2 describes a typical 4-term 3*DNF concept: a disjunction of four terms, where three of the terms are conjunctions of three primitive hypotheses, and the fourth term is a binary conjunction. For the experiments of this thesis, the concept generation algorithm produces only "clean" concepts, where a particular feature is used at most once in the concept.

```
CONCEPT:  (COND ((AND (EQP 'X10 'T)
                      (EQP 'X5 'T)
                      (EQP 'X11 'T)))
                ((AND (EQP 'X2 'F)
                      (EQP 'X7 'T)
                      (EQP 'X6 'F)))
                ((AND (EQP 'X1 'T)
                      (EQP 'X3 'F)
                      (EQP 'X12 'T))))
                ((AND (EQP 'X9 'F)
                      (EQP 'X8 'T))))
```

DOMAIN: (X1 (T F)), (X2 (T F)), (X3 (T F)), (X4 (T F)),
(X5 (T F)), (X6 (T F)), (X7 (T F)), (X8 (T F)),
(X9 (T F)), (X10 (T F)), (X11 (T F)), (X12 (T F))

MEANING: $((X10 \wedge X5 \wedge X11) \vee (\overline{X2} \wedge X7 \wedge \overline{X6}) \vee (X1 \wedge \overline{X3} \wedge X12) \vee (\overline{X9} \wedge X8))$

Instance space size = 4096 instances
Coverage = 50% positive
24 primitive hypotheses available

Figure C.2 Typical Boolean 4-term 3*DNF Concept.

The primitive hypotheses are of the form $(X1 = T)$, or as implemented, (EQP 'X1 'T). In the experiments of Chapters 5 and 6, sets of constructed hypotheses are used to exercise the hypothesis ordering mechanism of OXGate. The knowledge used to generate hypotheses in this domain is weak, syntactic knowledge and anticipates only the form of elements of the decision-trees. Two sets of constructed hypotheses are used, described below.

In one set of constructed hypotheses, the negations of all of the primitives are first created, e.g., (NOT $(X1 = T)$). Although this is logically equivalent to $(X1 = F)$, the negations are included to help flood OXGate with constructed hypotheses. The primitive hypotheses and their negations are then used as operands to create binary conjuncts and binary disjuncts. An example of a binary conjunct is $((X1 = T)$ AND $(X2 = F))$. An example of a binary disjunct is $((X1 = T)$ OR (NOT $(X3 = F)))$. This last example would be represented in OXGate as ((EQP 'X1 'T) OR (NOT (EQP 'X3 'F))). In this 12-featured Boolean domain, 24 primitive hypotheses are available. From these, 24 negations can be created. Using these 48 hypotheses as operands, 1128 binary conjuncts and 1128 binary disjuncts can be created for a total of 2280 constructed hypotheses. The other set of constructed hypotheses consists of 264 binary and 1760 trinary conjuncts created from the 24 primitive hypotheses, for a total of 2024 constructed hypotheses.

APPENDIX D

ARTIFICIAL NOMINAL FUNCTIONS

This appendix presents three artificially created concepts with nominal-valued attributes. In the experiments of Chapter 5, the training data used are random subsets of the instance space and the test data comprise the set of all examples in the instance space. Figure D.1 describes Concept A, a concept of medium complexity, Figure D.2 describes Concept B, a more complex structure, and Figure D.3 describes Concept C, a very simple construct, but one difficult for ID3 to represent. The concepts are described in CommonLisp where COND represents the "if-then-else-if" function and EQP is a test of whether the value of the attribute for a considered instance equals the given value.

```
CONCEPT:  (COND ((EQP 'color 'green))
                ((EQP 'color 'blue)
                 (OR (EQP 'shape 'circle)
                     (EQP 'shape 'square)))
                ((EQP 'color 'red)
                 (COND ((NOT (EQP 'size 'medium))
                        (OR (EQP 'shape 'triangle)
                            (EQP 'shape 'oval))))))

DOMAIN:   (color (red white blue blac  reen)),
          (shape (oval circle triangle square)),
          (size (small medium large)),
          (flavor (tart sweet salty bad))

MEANING:  An object is either green, a blue circle, a blue square,
          a red non-medium triangle, or a red non-medium oval.

Instance space size = 240 instances
Coverage = 37% positive
16 primitive hypotheses available
```

Figure D.1 Nominal Concept A.

```
CONCEPT: (COND ((EQP 'color 'green))
               ((EQP 'color 'blue)
                (COND ((EQP 'shape 'circle)
                       (OR (EQP 'flavor 'sweet)
                           (EQP 'flavor 'salty)
                           (EQP 'flavor 'tart)))
                      ((EQP 'shape 'triangle) (EQP 'flavor 'bitter))))
               ((EQP 'color 'red)
                (COND ((EQP 'shape 'triangle)
                       (COND ((EQP 'size 'large)
                              (OR (EQP 'flavor 'sweet)
                                  (EQP 'flavor 'salty)
                                  (EQP 'flavor 'tart)))
                             ((OR (EQP 'size 'small)
                                  (EQP 'size 'medium))
                              (EQP 'flavor 'bitter))))
                      ((EQP 'shape 'oval) (NOT (EQP 'size 'huge)))
                      ((AND (EQP 'flavor 'spicy)
                            (EQP 'size 'large)))))
               ((EQP 'size 'huge)))
```

DOMAIN: (color (red white blue black green orange purple aqua gray)),
(shape (oval circle triangle square diamond crescent)),
(size (tiny small medium large huge)),
(flavor (tart sweet salty spicy bitter))

MEANING: An object is either green, a sweet blue circle, a salty blue circle, a tart blue circle, a bitter blue triangle, a large sweet red triangle, a large salty red triangle, a large tart red triangle, a small bitter red triangle, a medium bitter red triangle, a non-huge red oval, a large spicy red *anything*, or huge if it isn't green, blue, or red.

Instance space size = 1350 instances
Coverage = 28% positive
25 primitive hypotheses available

Figure D.2 Nominal Concept B.

CONCEPT: (COND ((EQP 'color 'red))
((EQP 'size 'large))
((EQP 'shape 'circle)))

DOMAIN: (color (red white blue black green)),
(shape (oval circle triangle square)),
(size (small medium large)),
(flavor (tart sweet salty bad))

MEANING: An object is either red, large, or a circle.

Instance space size = 240 instances
Coverage = 60% positive
16 primitive hypotheses available

Figure D.3 Nominal Concept C.

The primitive hypotheses are of the form (*color* = *green*), or as implemented, (EQP 'color 'green). In the experiments of Chapters 5 and 6, sets of constructed hypotheses are used to exercise the hypothesis ordering mechanism of OXGate. The knowledge used to generate hypotheses in this domain is weak, syntactic knowledge and only anticipates the form of elements of the decision-trees. To construct hypotheses, the negations of all the primitives are first created, e.g., (NOT (*color* = *green*)). The primitive hypotheses and their negations are then used as operands to create binary conjuncts and binary disjuncts. An example of a binary conjunct is ((*color* = *green*) AND (*shape* = *oval*)). An example of a binary disjunct is ((*color* = *green*) OR (NOT (*taste* = *tart*))). This last example would be represented in OXGate as ((EQP 'color 'green) OR (NOT (EQP 'taste 'tart))).

For Concepts A and C, 16 primitive hypotheses are available with the given domain list. From these, 16 negations can be created. Using these 32 hypotheses as operands, 496 binary conjuncts and 496 binary disjuncts can be created for a total of 1008 constructed hypotheses. For Concept B, the 25 primitive hypotheses yield 25 negations, 1225 binary conjuncts, and 1225 binary disjuncts, for a total of 2475 constructed hypotheses.

APPENDIX E

NETTALK DOMAIN

The NetTalk domain database[1] is an updated version of the data set used by Sejnowski and Rosenberg in their study of speech generation using a neural network [Sejnowski and Rosenberg, 1987]. It is available for academic use from the Repository of Machine Learning Domains maintained by the University of California at Irvine. The database contains a list of 20,008 English words, along with a phonetic transcription for each character position of each word. The transcriptions include 51 phoneme representations, 5 stress/syllabic markers, and a marker for a silent character position. An additional marker is used for foreign/irregular words, providing a total of 58 distinct concepts explicitly represented in the database.

The original use of the data was to train a neural network to produce the proper phonemes, given a string of letters as input. The input to the network was a series of seven consecutive letters from one of the training words. The central letter in this sequence was the "current" one for which the phonemic output was to be produced. Three letters on either side of this central letter provide context that helps to determine the pronunciation. (A few words exist in English for which this local seven-letter window is not sufficient to determine the proper output.) For the study using this dictionary corpus, individual words were moved through the window so that each letter in the word was seen in the central position. Blanks were added before and after the word as needed.

In the network training task, 29 input units were provided to the network for each of the seven characters in the window. The output side of the network used a distributed representation for the phonemes. There were 21 output units representing

[1]

various articulatory features such as voicing and vowel height. Each phoneme was represented by a distinct binary vector over this set of 21 units. In addition, there were 5 output units that encoded the stress and syllable boundaries. This 26-bit string represented the input vector to a speech generation unit (DecTalk). The network was trained by sliding the words in the dictionary across the seven-character window. Several iterations were run using the entire dictionary before learning was declared complete. [Sejnowski and Rosenberg, 1987]

In this thesis, the NetTalk database was used differently. Comparisons between the effectiveness of the neural network approach and OXGate are neither possible nor relevant. For OXGate, a single concept (described below) was selected from the 58 explicitly represented concepts in the database. The output of OXGate was a decision-tree description of this concept while the neural network attempted to simultaneously learn all 58 concepts in the 26-bit vector representation. The mapping of the 26-bit string produced by the network to the phonemic representations was not provided in the NetTalk database, making direct comparisons between the two approaches impossible. In addition, the neural network input consisted of the entire dictionary database while OXGate used subsets of the data for training and testing.

The NetTalk Silent Concept Each of the seven character positions in the presentation window is an attribute (C1 through C7). The focus of learning is the center position, C4. Each attribute may take on one of 27 values: the 26 letters of the alphabet and the blank.[2] This domain defines a potential instance space of 27^7 ($> 10^{10}$) unique examples. The actual instance space is much sparser since the English language excludes many letter combinations. This set of attributes also defines $27 \times 7 = 189$ primitive hypotheses.

[2]Attribute C4 is also allowed to become "blank," although this situation never occurs in practice due to the nature of the windowing process.

The single concept of "silence in the center window position," known in this thesis as the *NetTalk Silent* concept, was chosen for its ease of understanding, potential for application of domain knowledge, and prevalence. Positive examples of this concept associate the "silent" phoneme marker with window position 4, or character C4. Of the more than 143,000 seven-character windows possible from the 20,008 words in the NetTalk database, a random set of 2000 instances was drawn for training and a separate set of 6000 instances was randomly drawn for testing. The training set consisted of 13.55% positive examples; the test set was 15.3% positive.

Two forms of potentially useful domain knowledge, both involving the characters on either side of position C4, provided the set of constructed hypotheses used to exercise OXGate. The first form of knowledge was the conjecture that the characters adjacent to the center position might be important. This conjecture is strongly supported by the data of Lucassen and Mercer [1984], where the mutual information between the center window position and neighboring letters is shown to be greatest with the adjacent positions, and decreases with the distance from the center. This knowledge was procedurally applied to exhaustively generate conjuncts of pairs of adjacent characters focused on C4, i.e., $((C3 = a)$ AND $(C4 = a))$, ..., $((C4 = z)$ AND $(C5 = z))$, including those using blanks. This application yielded 1458 constructed hypotheses. The second form of knowledge was more general, subsuming some of the first constructed terms, and conjectured that when the adjacent characters were the same, one of them would be silent. This was used to create two additional constructed hypotheses, $(C3 = C4)$ and $(C4 = C5)$, for a total of 1460. Other obvious forms of knowledge were not used, such as omitting occurrences of the "blank" in C4 and other impossible structures (e.g., C5 is blank when both C4 and C6 are not), and using knowledge about legal letter combinations to filter out hypotheses such as $((C3 = q)$ AND $(C4 = x))$.

APPENDIX F

BREAST CANCER DOMAIN

The Breast Cancer database consists of medical data collected on 286 patients that have undergone an operation to treat breast cancer. Of those, approximately 30% had the condition recur within five years. The database was provided for academic use by M. Zwitter and M. Soklic of the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. The Breast Cancer domain is one of three medical domains provided by the Oncology Institute that have repeatedly appeared in machine learning literature; lymphography and primary-tumor are the other two. The database is available for academic use from the Repository of Machine Learning Domains maintained by the University of California at Irvine.

The instances are described by nine attributes. Five of the attributes are integer-valued and were discretized into ranges (converted to nominal attributes) by the authors. Two attributes are nominal-valued, and the remaining two are binary (Boolean). The attributes and their possible values are described in Table F.1.

The data (286 instances) are divided into two classes: 201 instances of *no-recurrence-events*, the positive examples of the concept; and 85 instances of *recurrence-events*, the negative examples. In the experiments, a set of training data is 70% of the database (200 instances), drawn randomly without replacement. OXGate is trained on subsets of the training data and tested on the remaining 86 instances of the database. Different experimental runs use different random training sets. This approach is typical of machine learning research projects involving the Breast Cancer database.

Several efforts have investigated the Breast Cancer domain, with varying degrees of classification success. Chance, always saying the cancer will not recur, is 70% accurate.

Table F.1 Breast Cancer Domain Attribute Descriptions

Age:	the patient's age at the time of treatment	[integer]
values:	10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99	
Menopause:	the age of the patient at the beginning of menopause	[integer]
values:	lt40 (less than 40), ge40 (greater than or equal to 40), premeno (has not had menopause)	
Tumor-size:	(Note: The unit of measure was not provided)	[integer]
values:	0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59	
Inv-nodes:	the number of lymph nodes involved	[integer]
values:	0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39	
Node-caps:	(Note: The meaning of this attribute was not provided)	[binary]
values:	yes, no	
Deg-malig:	the degree of malignancy of the tumor	[integer]
values:	1, 2, 3	
Breast:	which breast the tumor was in	[nominal]
values:	left, right	
Breast-quad:	the tumor location in the affected breast	[nominal]
values:	left-up, left-low, right-up, right-low, center	
Irradiat:	whether radiation treatment was used	[binary]
values:	yes, no	

Various reports of the testing of oncologist specialists and internist non-specialists show they correctly predict the prognosis in 64-65% of the cases. Michalski *et al.* [1986] achieved accuracies over four runs of 66-68% with their system AQ15. They also tested an early version of Assistant and obtained accuracies of 67-72% depending on the pruning method. Clark and Niblett [1987] present several systems and configurations, with accuracies ranging from 65-72%. Cestnik *et al.* [1987] show an accuracy of 78% in Assistant86 with post pruning. Tan and Eshelman's system IWN boasts an accuracy of 73% [Tan and Eshelman, 1988]. Finally, Spackman [1988] claims accuracies as high as 85% for both his CRLS system and Michalski's AQ15. Since Michalski does not claim such a high accuracy in later publications, his results must be assumed to be questionable and await confirmation or refutation.

As indicated above, various methods have been tried, yet the general result is not much better than chance and often worse. The set of attributes in the Breast Cancer database appears to be inadequate to properly classify the data. It appears that a random 70% of the database is simply not representative of the remaining 30%. Two factors could contribute to this lack of success: noisy data and an inadequate description language [Clark and Niblett, 1987].

The database contains thirteen examples that could be considered obviously noisy. In these cases, one instance is a positive example of the concept and another instance with an identical description is a negative example. These thirteen anomalous examples account for an error-rate of only 2-3%. Other noise could exist in the database, but since the data are culled from existing medical records, and the data were verified after collection, it seems unlikely that the magnitude of the inherent error would be so large. A more realistic cause is an inadequacy in the description language [Cestrik *et al.*, 1987, Clark and Niblett, 1987, Michalski *et al.*, 1986].

An inadequate description language implies that the given attributes provide insufficient information. Constructive induction may enrich the language sufficiently to form an accurate classifier, but if the original attributes are simply the wrong measurements, then no amount of construction can compensate. It is quite possible that other attributes such as tissue type, condition of the immune system, and response of the particular tissue type to radiation treatment are also important. Even the patient's state of mind has a significant bearing on her recovery and continued health. It is my opinion that this particular database requires enrichment of its description language beyond that available with constructive induction.

REFERENCES

[Barr and Feigenbaum, 1982] Avron Barr and Edward A. Feigenbaum. *The Handbook of Artificial Intelligence*, volume 1. Addison Wesley Publishing Company, Inc., Reading, Massachusetts, 1982.

[Barr *et al.*, 1989] Avron Barr, Paul R. Cohen, and Edward A. Feigenbaum. *The Handbook of Artificial Intelligence*, volume 4. Addison Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[Bergadano and Giordana, 1988] Francesco Bergadano and Attilio Giordana. A Knowledge Intensive Approach to Concept Induction. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 305-317, Ann Arbor, MI, June 1988.

[Bergadano *et al.*, 1988] Francesco Bergadano, Attilio Giordana, and Lorenza Saitta. Automated Concept Acquisition in Noisy Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):555-578, July 1988.

[Bergadano *et al.*, 1990] Francesco Bergadano, Attilio Giordana, and Lorenza Saitta. Integrated Learning in a Real Domain. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 322-329, Austin, TX, June 1990.

[Blumer *et al.*, 1987] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred Warmuth. Occam's Razor. *Information Processing Letters*, 24:377-380, 1987.

[Breiman *et al.*, 1984] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, California, 1984.

[Cestrik *et al.*, 1987] Bojan Cestrik, Igor Kononenko, and Ivan Bratko. Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In *Proceedings of the Second European Working Session on Learning*, pages 31-45, Yugoslavia, 1987.

[Chan and Wong, 1990] Keith C. C. Chan and Andrew K. C. Wong. Performance Analysis of a Probabilistic Learning System. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 16-23, Austin, TX, June 1990.

[Cheng *et al.*, 1988] Jie Cheng, Usama M. Fayyad, Keki B. Irani, and Zhaogang Qian. Improved Decision Trees: A Generalized Version of ID3. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 100-106, Ann Arbor, MI, June 1988.

[Clark and Niblett, 1987] Peter Clark and Tim Niblett. Induction in Noisy Domains. In *Proceedings of the Second European Working Session on Learning*, pages 11-30, Yugoslavia, 1987.

[Dennan, 1959] William H. Dennan. *Principles of Mineralogy.* The Ronald Press Company, New York, 1959.

[Dietterich and Michalski, 1983] Thomas G. Dietterich and Ryszard S. Michalski. A Comparative Review of Selected Methods for Learning from Examples. In *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann Publishers, Inc., Los Altos, 1983.

[Diette ch, 1990] Thomas G. Dietterich. Machine Learning. *Annual Review of Computer Science*, 4:255–306, 1990.

[Drastal and Raatz, 1989] George Drastal and Stan Raatz. Learning in an Abstraction Space. Technical Report DCS-TR-248, Department of Computer Science, Rutgers University, January 1989.

[Drastal *et al.*, 1989] George Drastal, Gabor Czako, and Stan Raatz. Induction in an Abstraction Space: A Form of Constructive Induction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 708–712, Detroit, MI, August 1989.

[Erman *et al.*, 1980] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[Etzioni, 1988] Oren Etzioni. Hypothesis Filtering: A Practical Approach to Reliable Learning. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 416–429, Ann Arbor, MI, June 1988.

[Flann, 1990] Nicholas S. Flann. Applying Abstraction and Simplification to Learn in Intractable Domains. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 277–285, Austin, TX, June 1990.

[Friedman, 1937] M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[Goodman and Smyth, 1988] Rodney M. Goodman and Padhraic Smyth. Information-Theoretic Rule Induction. In *Proceedings of the 1988 European Conference on Artificial Intelligence*, Munich, August 1988.

[Gordon and Perlis, 1989] Diana Gordon and Donald Perlis. Explicitly Biased Generalization. *Computational Intelligence*, 5(2):67–81, May 1989.

[Hartmann *et al.*, 1982] Carlos R.P. Hartmann, Pramod K. Varshney, Kishan G. Mehrotra, and Carl L. Gerberich. Application of Information Theory to the Construction of Efficient Decision Trees. *IEEE Transactions on Information Theory*, IT 28(4):565–577, 1982.

[Haussler, 1987] David Haussler. Bias, Version Spaces and Valiant's Learning Framework. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 324–336, Irvine, CA, June 1987.

[Hettmansperger, 1984] Thomas P. Hettmansperger. *Statistical Inference Based on Ranks*. John Wiley and Sons, New York, 1984.

[Hume, 1990] David V. Hume. Learning Procedures by Environment-Driven Constructive Induction. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 113–121, Austin, TX, June 1990.

[Hunt *et al.*, 1966] Earl Hunt, J. Marin, and P. Stone. *Experiments in Induction*. Academic Press, New York, 1966.

[Jagannathan *et al.*, 1989] V. Jagannathan, Rajendra Dodhiawala, and Lawrence S. Baum. *Blackboard Architectures and Applications*. Academic Press, Inc., San Diego, 1989.

[Kerber, 1988] Randy G. Kerber. Using a Generalization Hierarchy to Learn from Examples. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 1–7, Ann Arbor, MI, June 1988.

[Langley *et al.*, 1986] Pat Langley, Jan M. Zytkow, Herbert A. Simon, and Gary L. Bradshaw. The Search for Regularity: Four Aspects of Scientific Discovery. In *Machine Learning: An Artificial Intelligence Approach, Vol II*. Morgan Kaufmann Publishers, Inc., Los Altos, 1986.

[Lenat, 1983] Douglas B. Lenat. The Role of Heuristics in Learning by Discovery: Three Case Studies. In *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann Publishers, Inc., Los Altos, 1983.

[Lucassen and Mercer, 1984] J. M. Lucassen and R. L. Mercer. An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 42.5.1–42.5.4, 1984.

[Manago, 1989] Michel Manago. Knowledge Intensive Induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 151–155, Ithaca, NY, June 1989.

[Matheus, 1989] Christopher J. Matheus. Feature Construction: An Analytical Framework and Its Application to Decision Trees. Technical Report UIUCDCS-R-89-1559, Ph.D. dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, December 1989.

[Matheus, 1990] Christopher J. Matheus. Adding Domain Knowledge to SBL through Feature Construction. In *Proceedings of the Eighth National Conference on Artificial*

Intelligence, pages 803–808, Boston, MA, July 1990. American Association for Artificial Intelligence.

[Michalski *et al.*, 1986] Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1041–1045, Philadelphia, PA, August 1986. American Association for Artificial Intelligence.

[Michalski, 1983] Ryszard S. Michalski. A Theory and Methodology of Inductive Learning. In *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann Publishers, Inc., Los Altos, 1983.

[Mingers, 1989] John Mingers. An Empirical Comparison of Selection Measures for Decision-Tree Induction. *Machine Learning*, 3:319–342, 1989.

[Mitchell *et al.*, 1983] Tom M. Mitchell, Paul E. Utgoff, and Ranan Banerji. Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics. In *Machine Learning: An Artificial Intelligence Approach*, chapter 6, pages 163–189. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1983.

[Mitchell, 1978] Tom M. Mitchell. Version Spaces: An Approach to Concept Learning. Technical Report STAN-CS-78-711, Ph.D. dissertation, Department of Electrical Engineering, Stanford University, December 1978.

[Mitchell, 1980] Tom M. Mitchell. The Need for Biases in Learning Generalizations. Technical Report CBM-TR-117, Computer Science Department, Rutgers University, May 1980.

[Muggleton and Buntine, 1988] Stephen Muggleton and Wray Buntine. Constructive Induction in First Order Logic. In *Proceedings of the International Workshop on Change of Representation and Inductive Bias*, pages 279–292, June 1988. (Also published as "Machine Invention of First-order Predicates by Inverting Resolution" in *Proceedings of the Fifth International Conference on Machine Learning*, pages 339-352, June 1988.).

[Muggleton, 1987] Stephen Muggleton. DUCE, an Oracle Based Approach to Constructive Induction. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 287–292, Milan, Italy, August 1987.

[Norton, 1989] Steven W. Norton. Generating Better Decision Trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 800–805, Detroit, MI, August 1989.

[Pagallo and Haussler, 1989] G. Pagallo and D. Haussler. Two Algorithms That Learn DNF by Discovering Relevant Features. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 119–123, Ithaca, NY, June 1989.

[Pagallo, 1989] Giulia Pagallo. Learning DNF by Decision Trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 639–644, Detroit, MI, August 1989.

[Pagallo, 1990] Giulia Pagallo. *Adaptive Decision Tree Algorithms for Learning from Examples.* Ph.D. dissertation, University of California at Santa Cruz, June 1990.

[Quinlan, 1985] J. R. Quinlan. Decision Trees and Multi-Valued Attributes. In *Machine Intelligence 11*, pages 305–318. Oxford University Press, 1985.

[Quinlan, 1986] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[Quinlan, 1989] J. R. Quinlan. Unknown Attribute Values in Induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 164–168, Ithaca, NY, June 1989.

[Ragavan and Rendell, 1991] Harish Ragavan and Larry Rendell. Estimating the Utility of Feature Construction in Empirical Learning. Submitted to *The Twelfth International Joint Conference on Artificial Intelligence*, 1991.

[Ragavan *et al.*, 1991] Harish Ragavan, Larry Rendell, and Der-Shung Yang. Identifying and Relieving Limitiations of Empirical Algorithms for Selective and Constructive Induction. (Unpublished manuscript, University of Illinois at Urbana-Champaign), 1991.

[Rendell and Seshu, 1990] Larry Rendell and Raj Seshu. Learning Hard Concepts Through Constructive Induction: Framework and Rationale. *Computational Intelligence*, 6(4):247–270, 1990.

[Rendell, 1985] Larry A. Rendell. Substantial Constructive Induction Using Layered Information Compression: Tractable Feature Formation in Search. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 650–658, August 1985.

[Rendell, 1989] Larry A. Rendell. Comparing Systems and Analyzing Functions to Improve Constructive Induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 461–464, Ithaca, NY, June 1989.

[Roberts *et al.*, 1974] Willard L. Roberts, George R. Rapp, and Julius Weber. *The Encyclopaedia of Minerals.* Van Nostrand Reinhold Company, New York, 1974.

[Samuel, 1959] Arthur L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3:210–229, 1959. (Reprinted in *Computers and Thought*, McGraw-Hill Book Company, Inc., New York, 1963).

[Samuel, 1967] Arthur L. Samuel. Some Studies in Machine Learning Using the Game of Checkers II. *IBM Journal of Research and Development*, 11(6), November 1967.

[Schaffer, 1990] Cullen Schaffer. A Proven Domain-Independent Scientific Function-Finding Algorithm. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 828–833, Boston, MA, July 1990. American Association for Artificial Intelligence.

[Schlimmer and Granger, 1986] Jeffrey C. Schlimmer and Richard H. Granger. Incremental Learning from Noisy Data. *Machine Learning*, 1:317–354, 1986.

[Schlimmer, 1987] Jeffrey C. Schlimmer. Incremental Adjustment of Representations in Learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 79–90, Irvine, CA, June 1987.

[Sejnowski and Rosenberg, 1987] Terrence J. Sejnowski and Charles R. Rosenberg. Parallel Networks That Learn to Pronounce English Text. *Complex Systems*, 1:145–168, 1987.

[Seshu *et al.*, 1989] Raj Seshu, Larry Rendell, and David Tcheng. Managing Constructive Induction Using Optimization and Test Incorporation. In *Proceedings of the Fifth International Conference on Artificial Intelligence Applications*, pages 191–197, Miami, FL, March 1989.

[Seshu, 1989] Raj Seshu. Solving the Parity Problem. In *Proceedings of the Fourth European Working Session on Learning*, pages 263–271, Montpellier, France, December 1989.

[Silver *et al.*, 1990] B. Silver, W. Frawley, G. Iba, J. Vittal, and K. Bradford. ILS: A Framework for Multi-Paradigmatic Learning. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 348–356, Austin, TX, June 1990.

[Smith and Rosenbloom, 1990] Benjamin D. Smith and Paul S. Rosenbloom. Incremental Non-Backtracking Focusing: A Polynomially Bounded Generalization Algorithm for Version Spaces. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 848–853, Boston, MA, July 1990. American Association for Artificial Intelligence.

[Soloway and Risemen, 1977] Elliot M. Soloway and Edward M. Risemen. Levels of Pattern Description in Learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 801–811, Cambridge, MA, August 1977.

[Spackman, 1988] Kent A. Spackman. Learning Categorical Decision Criteria in Biomedical Domains. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 36–45, Ann Arbor, MI, June 1988.

[Spackman, 1990] Kent A. Spackman. The Influence of Prior Probabilities on Inductive Learning: Bayes Bites Back. (Unpublished manuscript, Oregon Health Sciences University), 1990.

[Stepp and Michalski, 1986] Robert E. Stepp and Ryszard S. Michalski. Conceptual Clustering: Inventing Goal-Oriented Classification of Structured Objects. In *Machine Learning: An Artificial Intelligence Approach, Vol II*, pages 471–498. Morgan Kaufmann Publishers, Inc., Los Altos, 1986.

[Tan and Eshelman, 1988] Ming Tan and Larry Eshelman. Using Weighted Networks to Represent Classification Knowledge in Noisy Domians. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 121–134, Ann Arbor, MI, June 1988.

[Towell *et al.*, 1990] Geoffrey G. Towell, Jude W. Shavlik, and Michiel O. Noordewier. Adding Domain Knowledge to SBL through Feature Construction. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA, July 1990. American Association for Artificial Intelligence.

[Utgoff, 1986] Paul E. Utgoff. Shift of Bias for Inductive Concept Learning. In *Machine Learning: An Artificial Intelligence Approach*, volume 2, chapter 5, pages 107–148. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

[Utgoff, 1988] Paul E. Utgoff. ID5: An Incremental ID3. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 107–120, Ann Arbor, MI, June 1988.

[Watanabe, 1985] Satosi Watanabe. *Pattern Recognition: Human and Mechanical.* John Wiley and Sons, New York, 1985.

[Wirth and Catlett, 1988] Jarryl Wirth and Jason Catlett. Experiments on the Costs and Benefits of Windowing in ID3. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 87–99, Ann Arbor, MI, June 1988.

[Wu *et al.*, 1990] Yihua Wu, Shulin Wang, and Qing Zhou. An Integrated Framework of Inducing Rules from Examples. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 357–365, Austin, TX, June 1990.

[Yang *et al.*, 1991] Der-Shung Yang, Larry Rendell, and Gunnar Blix. A Scheme for Feature Construction and a Comparison of Empirical Methods. In *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, June 1991. (To appear).

[Yu, 1985] Po-Lung Yu. *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions.* Plenum Press, New York, 1985.

VITA

PII Redacted Major Gregg Harold Gunsch, United States Air Force, [redacted]. He attended the University of North Dakota in Grand Forks, North Dakota, where he received a Bachelor of Science degree in Electrical Engineering in May 1979. Upon graduation, he entered Officer Training School, Lackland Air Force Base, Texas, and received his commission in August, 1979. His first assignment was with the 394th ICBM Test Maintenance Squadron, Vandenberg Air Force Base, California. There he served in support of the Minuteman Operational Test and Evaluation Program where he provided expertise for troubleshooting missile and support system faults whose resolutions were beyond the scope of established technical procedures.

In the summer of 1982 he began his graduate studies at the Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton, Ohio. He received his Master of Science in Electrical Engineering in December 1983. His thesis applied expert systems technology to simulating a military planning task. Following graduation, he was assigned to the Avionics Laboratory at Wright-Patterson Air Force Base where he investigated the use of artificial intelligence technologies in the cockpit of fighter aircraft as a means of easing the information-processing burden of the pilot.

Major Gunsch began his doctoral studies in Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign in the fall of 1987. His dissertation, which examined a mechanism for applying domain knowledge during machine learning, was completed in April 1991, and his Ph.D. was conferred in May 1991. He is currently assigned as faculty at the Air Force Institute of Technology where he teaches graduate level computer engineering and artificial intelligence courses.

Academic and Professional Honors

- Member, Eta Kappa Nu, Delta Rho Chapter, University of North Dakota, 1979
- Distinguished Graduate, Officer Training School, 1979
- Air Force Commendation Medal, 1982
- President, Eta Kappa Nu, Delta Xi Chapter, Air Force Institute of Technology, 1983
- Distinguished Graduate, Master of Science in Electrical Engineering, Class GE-83D, Air Force Institute of Technology, 1983
- Member, Tau Beta Pi, Ohio Eta Chapter, Air Force Institute of Technology, 1983
- Air Force Commendation Medal, First Oak Leaf Cluster, 1987

Publications and Presentations

Gregg H. Gunsch and Robert V. Hebert. A Proposed Military Planning Task Simulator Using the ROSS Language. Defense Technical Information Center Report AD-A138 060, M.S. thesis, Air Force Institute of Technology, Dayton, OH, 1983.

Gregg H. Gunsch. Simulating Military Planning in the ROSS Environment. Presented at the *National Aerospace Engineering Conference*, Dayton, OH, 1984.

Gregg H. Gunsch and Larry A. Rendell. Opportunistic Constructive Induction: Using Fragments of Domain Knowledge to Guide Construction. In *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, 1991. (To appear.)