Concept Acquisition Through Representational Adjustment

(Technical Report Number 87-19)

Jeffrey C. Schlimmer

Department of Information and Computer Science
University of California, Irvine 92717
ArpaNet:schlimmer@ics.uci.edu

July 31, 1987

University of California
Irvine

Concept Acquisition Through Representational Adjustment

(Technical Report Number 87-19)

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Information and Computer Science
by
Jeffrey Curtis Schlimmer

Committee in charge:

Professor Richard H. Granger, Jr., Chair
Professor Dennis Kibler
Professor Pat Langley

1987

Contents

Acknowledgments

Without the help and encouragement of friends, this dissertation would not have been possible. Two outstanding members of the AI group at UCI, Kurt Eiselt and Doug Fisher, both aided by their insightful comments and encouraging suggestions.

Some of the greatest credit goes to three of the AI professors at UCI: Rick Granger, Dennis Kibler, and Pat Langley. Each has contributed to this dissertation through guidance, counsel, and careful criticism. Collectively their influences have led to a strongly empirical approach for analyzing the potential of proposed methods. Additionally, when the time came, they served as faithful dissertation committee members, improving this thesis through careful reading and comments.

A few individuals are due specific credits for their assistance. Michal Young provided much of the early formulation of the two basic learning methods. In many ways, this dissertation is the fruition of those seeds. Ross Quinlan pointed out a natural refinement to the matching process in a review of an earlier paper. Dennis Volper verified the analysis leading to 19 possible concepts in terms of three, binary-valued attributes in Sections 3.5 and 5.5. Mike O'Donnell, the University of Chicago, suggested considering alternative operators for the search through the space of possible chunks. Doug Fisher helped develop a measurement of the ease of concept acquisition. Steve Hampson, Don Rose, and Karen Wieckert all helped by carefully reviewing sections of this thesis. Tim Morgan, Richard Johnson, Mary Hegardt, Craig Rolandelli, and Hieu Tran, members of the department's support group, worked diligently to provide a consistent and reliable computing environment.

Abstract

Though the experiences of life exhibit unceasing variety, people are able to find constancy and deal with their world in a regular and predictable manner. This thesis promotes the hypothesis that the necessary abstractions can be learned. The specific task studied is inducing a concept description from examples. A model is presented that relies on a weighted, symbolic description of concepts. Though the description is distributed, novel examples are classified holistically by combining each portion's contribution. Each new example also refines the concept description: internal weights are updated and new symbolic structures are introduced. These actions improve description quality as measured by classification accuracy over novel examples.

Initially the concept description is highly distributed, being composed of many simple components. As learning progresses, sophisticated descriptive structures are added, and eventually the description is coalesced into a few highly predictive components. This qualitative change in the representation of the concept is a unique feature of the model.

The model extends previous work by allowing for noisy examples, unknown values, and concept change over time. To bolster claims of robustness, several experiments illustrating the model's behavior are reported. Key results illustrate that the model should scale-up to larger tasks than those studied and have a number of potential applications.

Chapter 1

Introduction

The experiences of life are seldom identical. No two visual images are the same; every verbal utterance is distinct. Yet amidst this unceasing variety we find a constancy that lets us deal with our world in a regular and predictable manner.

Indeed, this search for constancy has its purposes. It is not just that intelligent agents desire an orderliness around them. Rather, the need to cluster experience into rough groups is essential for predicting future events. Without the ability to form abstractions of experience, the similarities between life's events would go unnoticed and there would be nothing to gain from experience. Instead we all find that lessons learned in prior situations aid in helping us to decide what to do in the situations currently at hand, though the degree to which each individual utilizes this seems to vary.

This thesis is a study of the type of mechanisms that are involved in the abstractive process. A hypothesis of this work is that abstractions can be learned. It may be that humans are poor at forming abstractions, and useful ones have already been predetermined by their cognitive organization. The learning versus instinct issue may never be completely resolved, but it is clear that even if we could preprogram a mechanistic agent, it is undesirable to do so. Time and again, static solutions presented to solve particular intelligence problems have proven inadequate given even the slightest changes in the environment. A formulation of a model for intelligent behavior that relies on a learning capability at least holds out the promise of adaptiveness in the inevitable, unforeseen situations.

In order to understand the development behind the ideas presented here it may be useful to briefly consider the overall framework they are couched within. The general approach is to formulate a possible processing model, and then test this model by implementing it in a computer program. There are actually several steps to this iterative process of model development.

1. Identify desirable behavior

2. Formulate a model of how this behavior might arise given the available inputs
3. Transform this model into a specific computer program
4. Test whether the program exhibits the behavior
5. Determine whether the behavior arises from the model or just the program

These points are relatively straightforward, and collectively they constitute an approach for developing and testing a model. Each is developed in the following sections.

1.1 The learning task

The first step involves identifying desirable behavior, or in this context, identifying the inputs and outputs of interesting learning tasks. This is a loose use of the word "behavior," for the task may be to emulate the ability of humans in similar situations or to develop a commercially useful process that is computationally reasonable.

The task of learning concepts from examples has broad applicability. In general, given experiences and some feedback about them, form an abstract description that effectively predicts the feedback for novel experiences of the same type. For instance, given a mushroom and the information that it is poisonous, construct a description of "poisonousness" that will allow correctly guessing the edibility of other mushrooms. As subsequent chapters describe, this task has wide and varied instantiations.

In machine learning terminology, this task is called *learning from examples.* The mushrooms (or more generally experiences) are called *examples* because they either typify a concept (positive examples) or its opposite (negative examples or nonexamples). Central to the formulation of this learning task is the presence of feedback or identification of the class of each example: whether or not each example is positive or negative. The output of such a process is an abstract description that allows one to predict the identity of new examples. The earlier illustration relied on the concept of "poisonous," examples of specific mushroom samples, and an identification of each as either a positive example (edible) or a negative example (poisonous).

By way of contrast, a similar but distinct task formulated by machine learning researchers has a slightly different goal and does not rely on any explicit identification of the examples. Instead of attempting to identify the class of new examples, the point of the abstract description is to group the examples usefully, in order to aid in understanding the interrelations between examples or improve inference processes (Fisher, 1987). This task has been called *conceptual clustering*, reflecting its heritage in numerical clustering techniques and its predisposition to form descriptions based on conceptual level considerations.

1.2 Constraints on model formulation

The second step in this process is perhaps the least constrained. Given any specification of input and output behavior, there are an infinite number of plausible, intervening models. Therefore, some constraints must be used, either implicitly or explicitly, to narrow the space of possible models and to allow the researcher to explore a relatively small number. In some cases the choice of model is driven by a preference for simplicity or by apparent harmony between its parts. More often, these qualities are not readily apparent, and the researcher mush rely on other properties to favor one approach over another. Regrettably, these constraints often remain implicit and are based primarily on intuition.

One source of explicit constraint arises by considering the computational basis for the model. These constraints typically take the form of a bound on time or space processing. For instance, a model may avoid examining an exponential number of possibilities and limit its space requirements.

Psychology can also lend explicit constraints on the formulation of models. In many cases, clear experimental results indicate that, for humans at least, certain processing models are implausible. Keeping in mind that humans currently outperform artificially intelligent agents in many tasks, information about their underlying processes can guide in the formulation of viable models. At the very least, these constraints are guaranteed to include working models. Potentially, they may also eliminate many fruitless alternatives.

Constraining a model with psychological findings does not necessarily mean that it is to be a theory of human learning. Humans and computers may impose distinct computational constraints. Blending these two complementary sets can yield an effective model suitable for implementation on current computational devices.

As Chapter 7 describes, the learning method developed in this thesis is strongly constrained by a pair of psychological results from classical and instrumental conditioning. Though not an explicit model of animal or human learning, the model is consistent with experimental results and is more effective at its task as a result.

1.3 Transforming the model into a program

Conceptually, transforming the model into a program is usually straightforward. However, sometimes in the process of implementing a model, inconsistencies in its specification become apparent. This is a common situation in professional software development. User's requests are typically riddled with inconsistencies, and it is the job of the programmer to blend them. This property need not annoy the researcher; it can serve as an aid. Forcing a model into a consistent, impartial implementation clarifies its formulation, and in many cases leads to its

simplification. This is a primary contribution of computer science to psychology, since without an impartial model interpreter, psychological theories have struggled with cohesiveness and in some cases have been difficult to test. Of course, another contribution is the ability to test the implemented model.

The model of learning from examples developed here has been implemented in a computer program named STAGGER. Because of the tight interaction between forming the model and implementing it in a program, the two are collectively described in the chapters that follow. Each point where implementing helps refine the model is specifically noted to illustrate the interplay between forming a model and building its implementation.

1.4 Testing the program

The second to the last point is often neglected by researchers. Regrettably, even within computational disciplines, few test their programs thoroughly. It is easier to build a program that partially implements a model and will only work on a few cases. The ill effect of this undisciplined strategy is that poor models lurk with effective ones among those that are undertested.

Extensive testing can yield empirical results suitable for bolstering, or disconfirming, the analytical understanding that led to the model's original formulation. With this powerful tool, researchers are free to explore complex situations that are not readily amenable to theoretical analysis. Empirical results can be used to guide the search for analytical results.

One approach to testing a program that learns from examples involves measuring classification performance over novel examples. This arises because abstraction is an essential quality of a learning model. If the program only provides the correct identification of previously processed examples, then it does not provide any assistance in the fundamental task of abstracting away the intensive variety in experience; future performance may not be improved. Therefore, it is important to cross-validate the model by testing over a distinct set of examples.

A second approach to testing involves varying specific properties of the input in order to assess performance in the boundary conditions. For instance, if the examples may be noisy (sometimes the input to the model mistakenly identifies an edible mushroom as poisonous), the model should be tested to determine its behavior under varying amounts of noise; Besides examine the limiting cases (e.g., noise free and completely noisy), it may be useful to assess the differential effects of the middle cases. Usually it is desirable for a model's performance to degrade gently as the quality of its input deteriorates.

Several chapters of this thesis are primarily concerned with gathering and interpreting empirical testing results. Following the two guidelines above, the quality of derived concept

descriptions is tested by assessing its effectiveness on novel examples using the technique of continuous cross-validation. Similarly, several experiments are aimed at examining the limiting case behavior of STAGGER.

1.5 Evaluating the model

The final step is mapping the performance of the implementation back into components of the model. This step is especially important if the goal is to develop theories of human cognition. Researchers must convince others the program's effective behavior is a direct result of the model. Even for those not primarily motivated to understand human cognitive abilities, programs are nothing more than a means of testing the computational model. If it cannot be shown that the formulation of the model is primarily responsible for the effective functioning of the implementation, then there is little reason to believe that the work contributes anything more than a specific program.

For the model I discuss here, agreement between analysis and test results indicates that STAGGER is a faithful implementation of its model. At each point in the evaluation, I assess whether the results arise serendipitously from the implementation, or whether any reasonable realization of the model would do as well. To do this, I vary a number of implementation dependent qualities and observe the effects on output. Further, in describing the model's implementation, I explicitly identify programming considerations that initially appeared outside the scope of the model. These were varied during testing to assess the impact of their inclusion in STAGGER.

1.6 Overview

1.6.1 The main contribution

There are three contributions of the work represented by this thesis. First, it presents a method that is sufficient for a version of the learning from examples task. Specifically, little work has explored the effects of noisy data, unknown values, and the possibility that the concept may change over time. As I argue in later sections, relaxing the task along these lines makes the method more robust and more widely applicable for realistic learning applications. STAGGER is demonstrated to be effective despite these relaxations through intuitive analysis and empirical investigations.

Second, the interaction between the two methods used by STAGGER illustrate dynamic representational adjustment. One of the methods readily acquires some types of concepts, but is completely unable to describe others. In response, a secondary method is developed

that alleviates many of the former's limitations by changing its representational basis. This interplay between the two methods is a natural extension of work on bias adjustment (Utgoff, 1986; Utgoff & Mitchell, 1982). The limitations of the simpler method and the effects of adding a cooperating, secondary method are first presented at an analytic level and then demonstrated with a comprehensive empirical study.

A third contribution, related to the second, is the demonstration of useful cooperation between a numeric learning method and a symbolic one. Recent interest has come to focus heavily on models that rely on the numeric strengths of connections between simple processing units to represent information rather than on the more explicit, symbolic structures typically used in machine learning systems (e.g., [Anderson, 1986]). STAGGER combines a connectionist method with one that directly manipulates symbolic structures. In doing so, the model illustrates at least one source of useful interaction between the two approaches, relying on the strengths of each to address the learning task.

1.6.2 Organization

The thesis as a whole does not explicitly follow the organization of this introduction. Instead, chapters describing incremental developments of the learning model are interleaved with chapters illustrating its operation.

The second chapter introduces the first of the two learning methods in STAGGER. The method is based on a collection of predictive elements that are learned incrementally. Each of the elements is a dually weighted pattern, with the more relevant patterns for concept prediction accruing stronger weights. The weight learning method in STAGGER adjusts this description of the concept by modifying element weights based on a Bayesian statistic computed from characteristics of new examples. First, the issues of representation and performance are addressed by describing the initial set of descriptive elements, the semantics of their weights, and how these weights are used in matching. The chapter then explores learning issues by considering the Bayesian weight learning method.

The third chapter describes these processes in a step-by-step manner using four different concept acquisition tasks. First, a small, artificial concept is used to present the basic operations. Then the discussion turns to a simple spatial learning task. By borrowing a task from connectionist work, this domain intuitively cements Bayesian weighting in its connectionist heritage. A third demonstration is drawn from the field of information retrieval in order to depict the weight method's operation with a large number of concepts. Finally, the chapter closes by considering weight learning performance on a comprehensive set of tasks. Like single-layer connectionist approaches, Bayesian weighting is limited, clearly motivating the need for an auxiliary learning method.

Thus, the fourth chapter introduces a supplementary process that alleviates the shortcom-

ings of the weighting method. Functionally, it expands the concept description by conjoining, disjoining, or negating existing elements. The resulting complex elements are called *chunks*, for they serve a codification role in the overall picture of processing. This Boolean chunking process effective rewrites the input to the Bayesian weighting method, allowing the latter to function effectively in more situations that it could alone.

The worth of adding Boolean chunking to Bayesian weighting is clearly illustrated in Chapter 5. Following the role of Chapter 3 for Chapter 2, this chapter first describes the operation of chunking in a simple but concrete example. Then two naturally constrained concept acquisition tasks are presented. Each raises issues regarding the quality of examples and requires relaxing the problem of learning from examples. These tasks, together with those surveyed in Chapter 3, indicate that the task of learning from examples is pervasive. The last section repeats the comprehensive evaluation of Chapter 3 for the combined methods. Together they are suitable for a wider range of tasks than Bayesian weighting alone.

Although Chapter 5 discusses a relaxed version of the learning from examples task, it does not fully consider the effects of poor quality data on the resulting concept descriptions. Chapter 6 serves this purpose by systematically studying the effects of each type of degradation seen in Chapters 3 and 5. The results indicate that performance degrades gently as the input deteriorates and in many cases is close to the theoretical optimum.

Chapter 7 discusses psychological constraints for this learning task, and their effect on STAGGER's formulation. In particular, two results from the classical and instrumental conditioning literature directly affect the form of the Bayesian weighting method. Several desirable qualities observed in the prior chapter result from conformance to these constraints. On the other hand, operation of the Boolean chunking method may serve as a detailed hypothesis about subjects' performance in configural learning situations.

Given this perspective on the functioning and derivation of the two cooperative learning methods used by STAGGER, Chapter 8 examines other work in machine learning that addresses the task of learning from examples. However, the choice of systems for review is not primarily task based, but is an attempt to consider the general benefits of using cooperative learning methods. At some level, each system surveyed employs two learning methods: weight and representational adjustment.

In closing, this thesis attempts to be a thorough description of the functioning and capabilities of STAGGER: a robust model for the task of learning from examples.

Chapter 2

Bayesian Weight Learning

2.1 Introduction

Given a group of objects, each labeled according to its values for some pre-defined attributes, STAGGER constructs a general description of them. Abstract descriptions can be useful for classification and retrieval tasks arising in problem solving, for they allow rich experiences to be coalesced into useful groups. To do this, STAGGER relies on a collection of predictive elements that are learned incrementally. Each of the elements is a dually weighted pattern. The most relevant patterns have strong weights while irrelevant ones are weakly weighted. STAGGER adjusts this concept description by changing the weights attached to elements and by adding new pattern elements based on new data. In this chapter I describe the initial set of concept description elements, the semantics of their weights, and how a concept description is used to determine whether a new object fits it. Following this, I describe the process of adjusting element weights. I postpone a description of new element addition until Chapter 4. The discussion is detailed, and none of the operations essential to understanding the methods are omitted. After describing each of the processes involved and their motivation, in the following chapter I illustrate their operation through a series of detailed examples.

2.2 Concept representation

STAGGER represents what it knows about a concept as a collection of elements. Each element consists of a pattern and a pair of weights. The pattern matches some subset of the possible examples; for STAGGER, patterns consist of Boolean functions of attribute-values. A pair of weights serve to capture the relative importance of the pattern to the overall description of the concept: one indicates how much the concept depends on the pattern matching an example's features, and the other indicates reliance on "mismatching." In the following subsections, I

Table 2.1: Initial concept description elements.

ATTRIBUTE		VALUE
size	=	small
size	=	medium
size	=	large
color	=	red
color	=	green
color	=	blue
shape	=	circle
shape	=	square
shape	=	triangle

describe first the pattern portion of a concept description element and then its weights.

2.2.1 Concept description elements

Initially, the concept description elements are single patterns for each possible attribute-value pair. For instance, Table 2.1 depicts the initial elements for simple objects describable in terms of size, color, and shape. This choice for an initial set corresponds to a preference for simple descriptions, and if this set of elements proves inadequate for effective classification, STAGGER will modify it.

In general the elements need not be only simple attribute-value pairs; any Boolean combination ($\wedge$, $\vee$, $\neg$) of simple attribute-value pairs is allowable. In Chapter 4, I describe mechanisms in STAGGER that automatically add new conjunctions, disjunctions, and negations of existing elements in order to further refine the concept representation. Other possible methods might add relational or algebraic patterns.

Each new object is examined to determine whether or not it is an example of the concept or not. In this process STAGGER attempts to match each element pattern with features of the object; some patterns will match and some will not. In order to combine this mixture of information, each concept description element is dually weighted to indicate its predictiveness. In the following subsection I discuss the interpretation of these weights and their initial values.

2.2.2 Element weights

Each concept description element is a dually weighted pattern. One weight indicates the relevance of this pattern's matching a new object while the other indicates the importance of the failure of this pattern to match. Using two weights in this manner allows computing a degree of match between the concept description and the object, as I will describe in Section 2.3.

The first weight is an indication of the degree to which the element's matching predicts the concept. This weight is a Bayesian statistic and is termed *logical sufficiency* (LS).[1] LS is defined as the ratio of two conditional probabilities: the element will match given an example, and the element will match a nonexample.

$$\text{LS} = \frac{p(Match|Example)}{p(Match|\neg Example)} \tag{2.1}$$

LS ranges from zero to positive infinity. An LS value less than one (approaching zero) indicates a negative correlation, unity indicates independence, and a value greater than one indicates a positive correlation between the description element and the concept. An essential factor in LS is the term $p(Match \wedge \neg Example)$ in the denominator of Equation 2.1. If $Match$ is sufficient for $Example$ then by implication $\neg Match \vee Example$ is true, and its converse $Match \wedge \neg Example$ is false. LS measures the falseness of this latter statement. Appendix A provides further motivation for the form of LS.

The second element weight indicates degree to which failure to match predicts absence of the concept. It is termed *logical necessity* (LN) and is also the ratio of two conditional probabilities.

$$\text{LN} = \frac{p(\neg Match|Example)}{p(\neg Match|\neg Example)} \tag{2.2}$$

LN also ranges from zero to positive infinity but has the inverse interpretation of LS: An LN value less than one indicates a positive correlation, and a value greater than one indicates a negative correlation.[2] For both Bayesian weights a value of one indicates that the concept description element is irrelevant to the concept.

It can be shown that when one element weight is irrelevant the other is also, or $\text{LS} = 1 \Leftrightarrow \text{LN} = 1$. Conversely, when one weight is relevant the other will be too, or $\text{LS} > 1 \Leftrightarrow \text{LN} < 1$ and $\text{LS} < 1 \Leftrightarrow \text{LN} > 1$. However, in general a pattern is not equally sufficient and necessary ($\text{LS} \neq 1/\text{LN}$). For instance, if $p(M|E) = 3/10$ and $p(M|\neg E) = 1/10$ then $\text{LS} = (3/10)/(1/10) = 3$ and $\text{LN} = (1 - 3/10)/(1 - 1/10) = 7/9$.

Initially each concept description element is assigned the pair of unbiased weights $\text{LS} = 1$ and $\text{LN} = 1$. Table 2.2 lists some initially weighted concept description elements for simple objects.

An alternative form of this concept description conveys pictorially the LS and LN strengths of various elements. Figure 2.1 depicts the initial concept description graphically. The small, outer circles represent the individual concept description elements, and the thickness of the

[1]The formulae in Equations 2.1 and 2.2 were originally derived in work on the Prospector mineral exploration system (Duda, Gaschnig, & Hart, 1979).

[2]LN could be defined as the inverse of Equation 2.2 and thereby afford a comparable interpretation of its value. However, the current asymmetry simplifies the matching process to be described below.

Table 2.2: Initially weighted concept description elements.

PATTERNS			WEIGHTS	
ATTRIBUTE		VALUE	LS	LN
size	=	small	1	1
size	=	medium	1	1
size	=	large	1	1
color	=	red	1	1
color	=	green	1	1
color	=	blue	1	1
shape	=	circle	1	1
shape	=	square	1	1
shape	=	triangle	1	1

Table 2.3: Concept description elements after processing one example.

PATTERNS			WEIGHTS	
ATTRIBUTE		VALUE	LS	LN
size	=	small	4/3	2/3
size	=	medium	2/3	4/3
size	=	large	2/3	4/3
color	=	red	4/3	2/3
color	=	green	2/3	4/3
color	=	blue	2/3	4/3
shape	=	circle	4/3	2/3
shape	=	square	2/3	4/3
shape	=	triangle	2/3	4/3

solid and dashed lines emanating from each element correspond to the magnitude of the LS and LN weights, respectively. In this case the uniformly thin lines reflect that all of the weights are unbiased.

This initial concept description represents a naive state of understanding. After a match-update cycle in which a single object is processed, the weights associated with each of the elements are updated as described in Section 2.4. Table 2.3 depicts the changes after processing a positive example: a **small, red circle**. Note that the LS weight is larger than before for the **size = small** pattern, reflecting its predictive value. Conversely, the LS weight for **size = medium** has dropped, for this pattern did not predict the example.

As a graphical depiction (Figure 2.2), this concept description conveys the emerging importance of the **size = small**, **color = red**, and **shape = circle** elements, for they have thicker (more weighty) LS and LN lines toward the concept center.

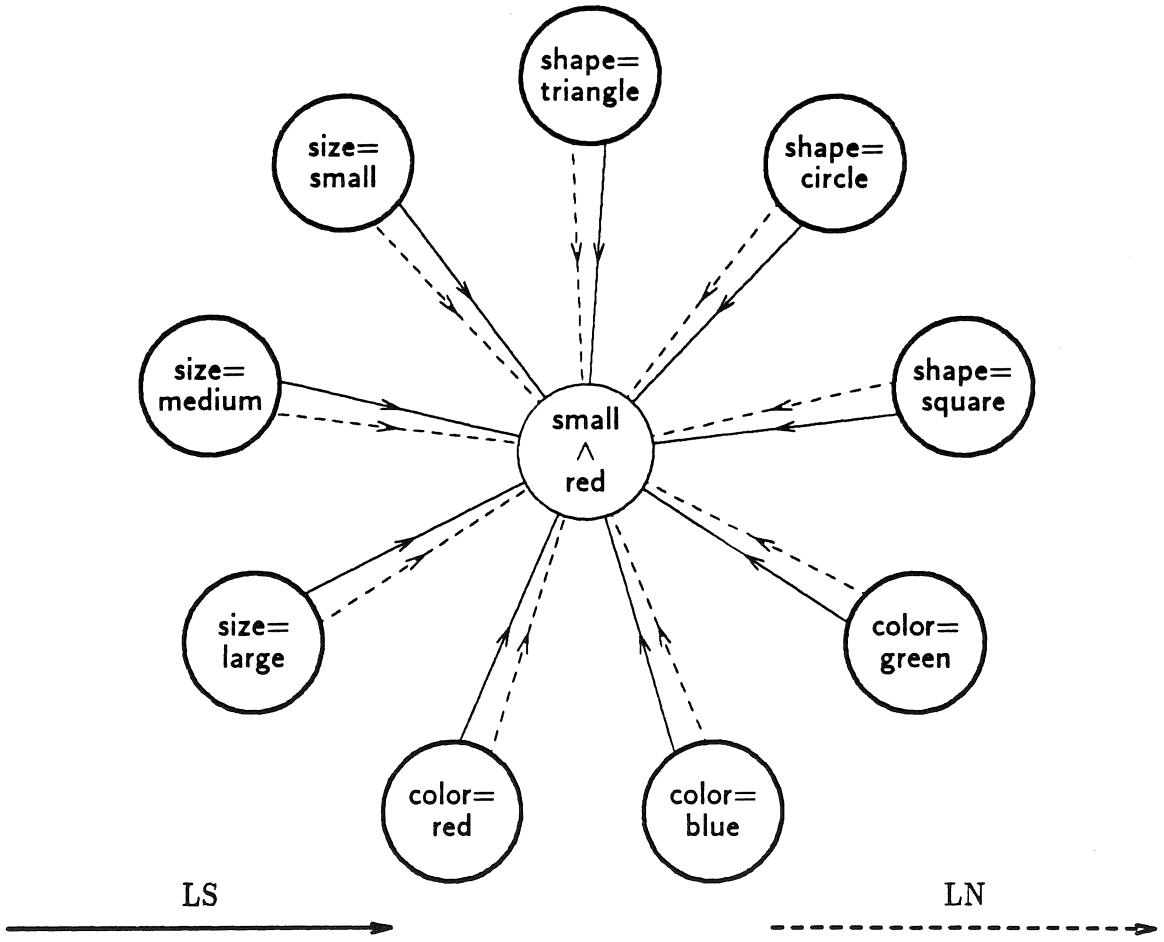

Figure 2.1: Depiction of initially weighted concept description elements.

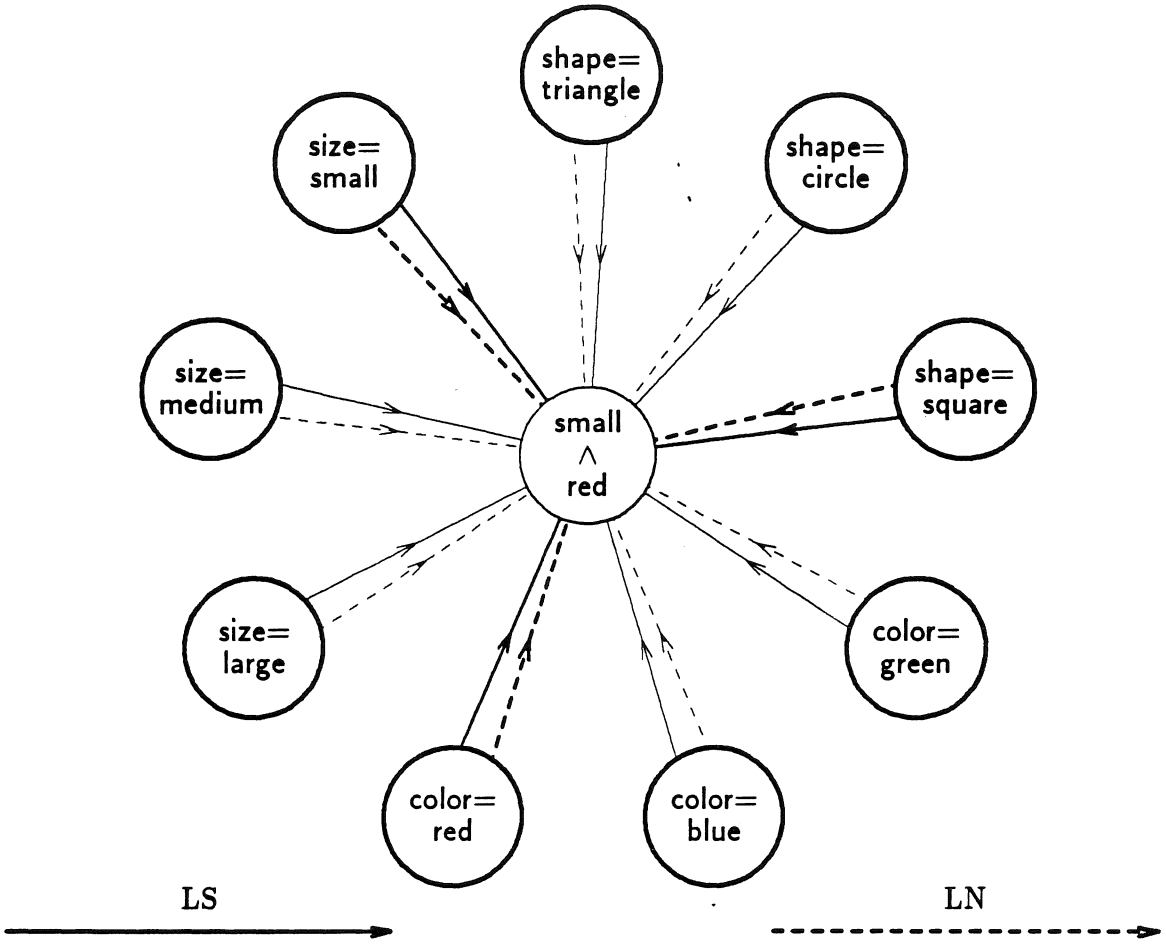

Figure 2.2: Concept description after processing one example.

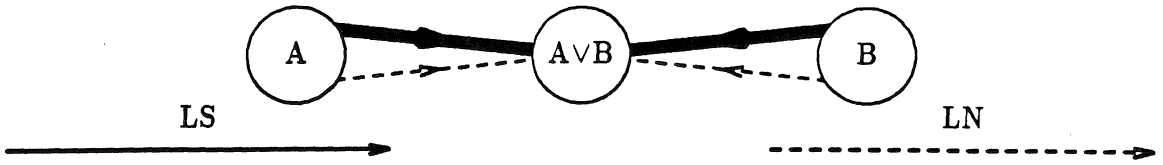

Figure 2.3: Idealized disjunctive concept description.

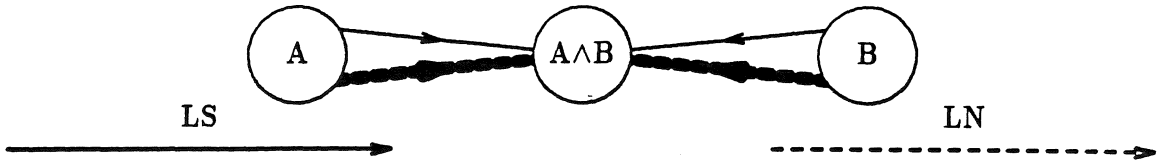

Figure 2.4: Idealized conjunctive concept description.

These Bayesian weights have a strong correspondence to the Boolean concepts of disjunction, conjunction, and negation. For instance, the LS weight represents the sufficiency of the element for the concept. If an element is sufficient, it is a disjunct of the concept description. Thus, to represent a concept description of $A \vee B$, both A and B would have large LS weights. Figure 2.3 depicts an idealized disjunction. The thickness of the solid lines corresponds to strong LS weights.

The LN weight, on the other hand, represents necessity, and an element with a small (significant) LN weight is a conjunct of the concept; $A \wedge B$ is represented by a LN weight $<$ 1 for both A and B. In Figure 2.4 the thickness of the dashed lines convey the significance of the LN weights. For either weight, a negated element is one with an inversely predictive weight. Thus, $\neg A$ can be represented by $\mathrm{LS}_A < 1$ or $\mathrm{LN}_A > 1$.

Representing the concept as a set of relatively independent, weighted patterns suggests a natural method for matching the concept as a whole against some new example. In the next subsection I describe that method and then discuss some criteria by which evidence combination methods may be examined.

2.3 Matching examples with concepts

One strategy for matching would be to translate the weights into an explicit rule; $\mathrm{LS}_A > 1$, $\mathrm{LS}_B > 1$, and $\mathrm{LN}_C < 1$ could be mapped into $(A \vee B) \wedge C$. Instead of this type of discrete strategy, the weights are used in a summative manner to generate a holistic expectation. As I alluded in the previous section, combining element weights allows STAGGER to use the mixture of information resulting from patterns that match and those that do not. In brief, the sufficiency weight increases expectation of a positive example when a pattern is matched, and the necessity weight decreases expectation when a pattern is unmatched. Taken together with

an indication of how frequently positive examples occur, the increases from LS weights and decreases from LN weights yield an indication of the quality of match between the concept and some new example. If that match is high, then STAGGER expects the object to be an example of the concept. Following a description of this procedure for computing an expectation of the concept, I conclude this section with a discussion of six criteria presented by Cohen (1977) for evidence combination schemes.

2.3.1 STAGGER's matching procedure

STAGGER utilizes each of the concept description element weights when it matches a new example with the concept description. First, STAGGER tries to match each element's pattern with the example; typically the pattern will match completely or will fail to match. At the individual pattern level, partial matches are counted as mismatches. The LS weights of all matching patterns and the LN weights of all unmatched patterns are multiplied together. This product is further scaled by the likelihood of an example: the prior odds of a positive example. The resulting product is the posterior odds in favor of a positive example. This expectation may be easily converted into the probability of a positive example by noting that $p = odds/(1 + odds)$.

$$Odds(Example|Features) = Odds(Example) \times \prod_{\forall Matched} \text{LS} \times \prod_{\forall Unmatched} \text{LN} \qquad (2.3)$$

A high matching score is taken as the basis for predicting a positive example. Conversely, a low score directs STAGGER to expect a nonexample. A score close to unity indicates about equal odds of the example being either positive or negative, leaving STAGGER to make a random guess.[3] Note that this process of combining pattern matching evidence is completely cooperative, for each of the elements contributes its expectation to the total outcome. A form of competition does exist in STAGGER's learning mechanisms, as I describe in Section 4.3.

As I noted before, these weights have a strong correspondence to Boolean disjunction, conjunction, and negation. Recall that an idealized OR is represented by large LS weights for each of the disjuncts. In Equation 2.3 this has the effect that a positive example is expected if any of the disjuncts are matched, for their large LS weights can increase expectation substantially; any disjunct is sufficient for expectation of an example. Conversely, conjuncts have small LN weights. If any of them are unmatched and their LN weights are multiplied with the current matching score, then their omission will prevent expectation of a positive example. Each of the conjuncts are therefore acting as a necessary part of the concept description.

[3]The multiplicative form of this matcher and the use of LS, LN, and prior odds are all due to the original work on Prospector (Duda, Gaschnig, & Hart, 1979).

A negation, represented by either a small LS or a large LN weight, shows its influence by decreasing expectation if matched and increasing expectation if it is omitted.

This matching process can be generalized to allow for the possibility that the attribute-value pairs are fuzzy, i.e., they take on values from zero to one. For instance, the three values of an attribute may have the intensities $\{1/3, 1/2, 1/6\}$. To accommodate this possibility, instead of using only one weight to influence expectation, both are used: the LS weight in proportion to the value intensity and the LN weight in proportion to one minus the intensity. With these modifications the matching Equation 2.3 becomes:

$$odds(Example|Features) = odds(Ex.) \times \prod_{\forall elements} \left[\begin{array}{c} Intensity \times \text{LS} \\ + \\ (1 - Intensity) \times \text{LN} \end{array} \right] \qquad (2.4)$$

In the limit, Equation 2.4 reduces to Equation 2.3; if the attribute-value intensity is one, only the LS weight will be used, and conversely, if the value intensity is zero, only the LN weight influence expectation. This generalized matching form will be exercised in Section 3.3, though typically the more discrete form of Equation 2.3 will be used.

The use of dual weights in computing expectation naturally allows for the possibility that some of the values of the example may not be specified and thus may make it impossible to determine whether an element matches or not. In single weight systems, either the solitary weight is used only when the element matches, or some mathematical inverse of the weight is used when the pattern is unmatched; the *unmatchable* case has no effect on expectation. In STAGGER, when the element cannot be matched due to unknown values, both of the element's weights are used to adjust a confidence range. The larger of the two weights is used to raise the upper bound on confidence while the smaller of the two reduces the lower bound. In this manner, STAGGER is able to specify not only the degree of expectation, but also its precision. For instance, if STAGGER were to try matching a **medium circle** with the concept description elements listed in Table 2.3, because there is no specification of the example's color, the three elements corresponding to the color attribute's values (**color = red**, **color = blue**, **color = green**) would be unmatchable. Their LS and LN weights would be used to expand the range of expectation, with the larger weights (LS=4/3, LN=4/3, LN=4/3, respectively) raising the upper bound and the smaller (LN=2/3, LS=2/3, LS=2/3) dropping the lower bound. Using both weights in this manner is equivalent to computing all possibilities and then reporting on the greatest and smallest possible expectation. This bounding on expectation is similar to the approaches followed by others (Quinlan, 1982; Shafer, 1976; and Shortliffe, 1976) and is further illustrated in Section 5.4.

2.3.2 Prototypicality effects

Using a number of patterns to determine whether or not an example is an instance of the concept differs from most machine learning systems in which a single characterization completely directs concept prediction (see Chapter 8). However, this matching process is similar to the matching used by Smith and Medin's (1981) general processing algorithm. In the latter, concept membership is calculated by summing the weighted values of each matching feature. Both of these calculations have the desirable property that the prototypicality of each new example is automatically computed. An example with a high matching score is highly prototypical; one with a smaller score is less so.

The mapping between a high matching score and a highly prototypical example suggests a way to generate a prototypical example: for each attribute, list the most predictive value. This construction relies on the assumption that the more predictive an attribute-value pair, the more prototypical it is of a positive example.

This simple method of combining the matching of various patterns has some desirable properties, as we have seen in this subsection. However, what can be said of the validity of the matching computation? In the next subsection I evaluate STAGGER's matcher with respect to criteria offered by Cohen (1977).

2.3.3 Evidence combination criteria

Cohen (1977) suggests six criteria for evaluating evidence combination schemes. He bases his assessment on a careful evaluation of the decision processes a British or American jury follow in civil and criminal legal cases. The advantage of this perspective is that it yields a tangible field of decision making for which a rational agent can readily verify the outcomes. Indeed, each juror is instructed to use their normal, everyday decision making processes, deciding the case as if it were their own personal matter. Of the six criteria Cohen discusses, four are applicable to many forms of decision making, while two entail assumptions that limit their scope to the courtroom. Cohen focuses mainly on alternative accounts of rational decision making, especially as they are applied in a prescriptive manner to the juror. Nevertheless, examining them in the light of concept matching leads to some interesting observations.

The first criterion Cohen examines is the effect of a conjunction of evidences on the conclusion. For instance, the prosecutor often seeks to establish the defendant's guilt on the basis of a number of points: motive, opportunity, method, etc. In a traditional Bayesian probabilistic method, combining the points of evidence relies on the general multiplication rule of the form $p(A \wedge B) = p(A) \times p(B|A)$. Now consider that in order to prove guilt, the prosecutor has to establish a probability of guilt at least $1/2$. Given the multiplication rule, the addition of each point of evidence ($p(B|A)$) *decreases* probability of guilt even if each bit of evidence

carries a probability greater than 1/2. On the other hand, by omitting a point of evidence, the prosecution seems to have a better chance of conviction. If we map evidence onto concept elements and guilt onto expectation of an example, it is possible to examine the behavior of STAGGER's matching procedure with respect to this criterion. By combining the probability ratios through multiplication, STAGGER reaches a more palatable conclusion: those points of evidence which are less than certain (Bayesian weight < 1) decrease anticipated guilt, while those which are more certain increase the probability of guilt. With additional evidence, STAGGER becomes more certain of the need for a conviction; thus the qualitative nature of the evidence combination appears to be correct. This is the underlying motivation for dealing with probability ratios and odds in the matching process described above.

Second, Cohen considers the effect of transitive chains of logic. In courts of law, each point in a chain of reasoning must be established beyond all reasonable doubt before the next link can be drawn. Though traditional probabilistic methods are too strict with conjunctive reasoning, they are too lenient with transitive evidence. Using the general Bayesian combination rule, if B may be shown from A, and C may be shown from B, then given A, the computed probability for C is

$$p(C|A) = \frac{p(B|A) \times p(C|A \wedge B)}{p(B|A \wedge C)} \tag{2.5}$$

This allows for the possibility that $p(B|A)$ could be relatively small and yet the probability of $p(C|A)$ could be boosted to a high value either through a large $p(C|A \wedge B)$ or a small $p(B|A \wedge C)$. Probability shows what is known from probable facts when what is needed is showing what is probable from known facts. For instance, if a person was accused of stealing secrets from a nearby government installation by tapping into their microwave transmissions, it would be jumping to conclusions to assume that the suspect had access to the necessary equipment (microwave antenna, receiver, ...) as a way of supporting their guilt. This conclusion is absurd because the hypothesis of equipment access is barely plausible, and the strength of the subsequent inference does not overcome the weakness of its premise.

STAGGER's transitive reasoning properties are demonstrated through additional concept description patterns that map expectation of one concept onto an element in another's description. One approach would be to assign each pattern a numeric value reflecting the odds in favor of its concept, but that would be like allowing weak links in a chain of reasoning to prove a strong conclusion. Rather, each of these patterns is assigned a true or false value, more closely representing the role of facts in a transitive, legal evidence chain (Schlimmer & Granger, 1986a).

A third criterion Cohen considers is that a belief of guilt is not necessarily one minus the belief of innocence, or that $p(guilty) \neq 1 - p(\neg guilty)$. For instance, suppose that at a certain rodeo, 499 people paid to enter the gate, yet there are 1,000 spectators in the stands. Now, consider a typical spectator in the stands. The probability that he paid admission is 0.49, and

using the rule of negation, the probability that he crashed the gate is 0.51. Thus the organizers seem to be justified in pressing this suspected gate crasher for the price of admission. Taken to the extreme, they could demand payment of each spectator, collecting 1,499 admissions for 1,000 spectators. In civil law decisions, the official phrase is "balance of probabilities," which implies that the probability of guilt need only be greater than 1/2. If jurors used traditional probabilistic decision techniques there would be a 49% chance of convicting an innocent person. This threshold could be raised to 8/10, but that would still result in 20% wrongly convicted and place a still heavier burden of proof on a procedure already struggling with conjunctive evidence.

Although STAGGER has so far been presented as having a single concept description for both positive and negative examples, in fact, for multiple concept learning tasks, there is a concept description for each concept. This means that STAGGER may expect an example to be both positive and negative (though one more than the other). For instance, in Section 3.3 STAGGER utilizes four concept descriptions that indicate when to apply four directional operators, one for each major point of the compass. The **move-north** and **move-south** operators are complementary concepts, and instead of assigning the inverse of one direction's expectation to the opposing direction, individual concept descriptions are utilized, with STAGGER preferring the operator that has the strongest expectation. In general, the matching procedure (Equation 2.3) does not assign the inverse of odds to all alternative outcomes.

Fourth, Cohen notes that probabilistic systems have difficulty with the ability to show proof beyond reasonable doubt in the same manner that jurors have come to expect. For instance, he notes that:

> We are more inclined to hold that a proposed conclusion falls short of certainty because there is a particular, specifiable reason for doubting it, than to hold that it is reasonable to doubt the conclusion because it falls short of certainty.
>
> What seems to be needed in practice for assessment of proof in a criminal trial is a list of the various points that all have to be established, and of the various let-outs that all have to be barred, in relation to each element of the crime, if guilt is to be proved beyond reasonable doubt.[4]

STAGGER's matching procedure and the associated roles of the dual weights attached to each concept element serve to identify both the various patterns that must be established, as well as those that must be barred for expectation of the concept. A highly significant necessity weight (LN $\ll 1$) serves to identify those patterns that must match all examples of the concept; an inversely significant necessity weight (LN $\gg 1$) identifies let-out patterns that must not match examples. The sufficiency weight serves a similar representational purpose,

[4](Cohen, 1977, p. 82)

for it identifies those patterns whose matching in and of itself indicates an example, as a confession would in a legal case.

Cohen's fifth and sixth points have a restricted applicability. Though they reflect general judgmental abilities, they are influenced further by the role a juror must serve. He points out that it is difficult to determine an acceptable basis for probabilities in the legal situation. For instance, one basis for estimating probabilities is a system of betting odds. A juror might be instructed to state the probability of belief in the guilt of a defendant in terms of the odds he would be willing to accept in a wager. Cohen discounts this option, stating that rational men do not bet on situations where the outcome cannot be discovered. While this may be true of the juror's situation, it is not for many decision-making tasks, including concept acquisition. Cohen's final criterion specifies that corroborative or converging evidence usually increases confidence in the final conclusion even if each of the pieces of evidence is less than certain. This unwillingness to accept coincidence may be accommodated by probabilistic methods if prior probabilities are properly utilized. Yet, as Cohen points out, jurors are specifically instructed to examine the individual case on its merits and to weight only the evidence presented in the court room. However, outside the court room the juror undoubtedly uses knowledge about the prior probability of certain events. For instance, the average man effectively avoids street salesmen, knowing of prior annoying experiences.

Cohen's six criteria for evidence combination are strongly based on the behavior of a rational juror who weights legal evidence. The juror's behavior contradicts many properties found in traditional probabilistic systems, though these actions are not counterintuitive. Inasmuch as STAGGER incorporates a decision making process in its matcher, it seems to perform reasonably well in light of Cohen's criteria. The viability of this matcher is in large part due to the quality of its original formulation (Duda, Gaschnig, & Hart, 1979).

This section gives a representation for learned concepts in terms of weighted patterns and an overview of their operational use in matching. However, without some process of adjustment the concept cannot represent subsequent experience. In the next section I discuss processes that adjust element weights and refine the concept.

2.4 Weight learning

In addition to representing concepts in a distributed manner and using Bayesian measures to compute a prototypical expectation, STAGGER incrementally modifies both the weights associated with individual concept description elements and the structure of the elements themselves. These two latter abilities allow STAGGER to adapt its concept description to better reflect the concept. I defer discussion of changing pattern content until Chapter 4 and focus on Bayesian weight learning in this section. First, I discuss two alternative methods for

Table 2.4: Matching a characterization to an example.

EXAMPLE	PATTERN	
	MATCHED	UNMATCHED
POSITIVE	Confirming (C_P)	Infirming (I_P)
NEGATIVE	Infirming (I_N)	Confirming (C_N)

adjusting the elements' weights. One is based on counting the number of times each pattern has successfully and erroneously predicted examples. The other is based on a probability estimating method that incrementally incorporates the effects of the most recent match or mismatch between pattern and example. Finally, I briefly address a criticism levied at Bayesian representations by Shafer (1976).

2.4.1 Bayesian weighting by counting matching types

There are at least two ways to incrementally adjust the weights attached to the concept description elements. One way is to keep count of the number of times a pattern has correctly and erroneously predicted an example. Alternatively, estimates of the conditional probabilities themselves might be maintained. In this subsection I describe the count-based method and leave the estimation method until the following subsection.

One way of computing the weights for each concept description element is to count each of the four possible matching events: either the pattern matches the example or not, and either the example is a positive example or not. Using the terminology developed by Bruner, Goodnow, and Austin (1956), a matching pattern in a positive example is called *positive confirming* (C_P). The inverse, an unmatching pattern in a negative example, is called *negative confirming* (C_N). In each of these cases, the example confirms the validity of the pattern. The other two cases, where there is a mismatch between the pattern and the example, are termed *positive infirming* (I_P) and *negative infirming* (I_N). Both of these matching types discredit the predictive value of the pattern as an indicator of the concept. Table 2.4 lists these matching events and their abbreviated names in a two-by-two design.

Note that a positive infirming mismatch may also be termed an *error of omission* (since the pattern was omitted), while a negative infirming match is sometimes termed an *error of commission.* However, in the context of this thesis it will be useful to distinguish between the *prediction* STAGGER makes and the *matching* of individual concept description elements, the former being an occasion when the program predicts a positive example that is actually a negative example, and the latter being an element matching in a negative example.

Initially, each of the counts is set to one. Table 2.5 lists the initial concept description

Table 2.5: Initial concept description elements and counts.

Patterns			Counts				Weights	
Attribute		Value	C_P	I_N	I_P	C_N	LS	LN
size	=	small	1	1	1	1	1	1
size	=	medium	1	1	1	1	1	1
size	=	large	1	1	1	1	1	1
color	=	red	1	1	1	1	1	1
color	=	green	1	1	1	1	1	1
color	=	blue	1	1	1	1	1	1
shape	=	circle	1	1	1	1	1	1
shape	=	square	1	1	1	1	1	1
shape	=	triangle	1	1	1	1	1	1

elements and counts for simple objects.

The Bayesian weighting measures LS and LN may be easily calculated for each concept description element by keeping counts of the possible situations listed in Table 2.4. The formulae for LS and LN in terms of the counts of each situation are:

$$\mathrm{LS} = \frac{C_P(I_N + C_N)}{I_N(C_P + I_P)} \qquad \mathrm{LN} = \frac{I_P(I_N + C_N)}{C_N(C_P + I_P)} \tag{2.6}$$

Derivations of these formulae appear in Appendix B.

These weights are easily modified following matching. For instance, processing a positive example, a **small, red circle**, results in the concept description listed in Table 2.6, which is identical to Table 2.3 save the additional specification of the underlying counts.[5] Note that the C_P count has been incremented for size = small, for this pattern was matched in the positive example. Similarly, the I_P count is larger for the unmatched pattern size = medium. None of the negative counts are changed since this was a positive example.

If patterns yield a fuzzy match, this count-based method may be generalized like matching Equation 2.3. Instead of updating one of the counters for each pattern, corresponding to an all-or-none match, two counters are updated, one proportional to the intensity of the attribute-value and the other to one minus the value intensity. For instance, if a nonexample has just been processed, then the value intensity would be added to the negative infirming (I_N) count, and one minus the intensity would be added to the negative confirming (C_N) count. For more complex patterns, the intensity is determined using the minimum intensity for a conjunctive pattern and the maximum intensity of a disjunct. A negated pattern has one minus the intensity of its argument. This revised match counting procedure is listed below in

[5]This table appears again in Section 3.2 as Table 3.3. However, there the counts are decayed, as described near the end of this section.

Table 2.6: Concept description elements after processing one example.

Patterns			Counts				Weights	
Attribute		Value	C_P	I_N	I_P	C_N	LS	LN
size	=	small	2	1	1	1	4/3	2/3
size	=	medium	1	1	2	1	2/3	4/3
size	=	large	1	1	2	1	2/3	4/3
color	=	red	2	1	1	1	4/3	2/3
color	=	green	1	1	2	1	2/3	4/3
color	=	blue	1	1	2	1	2/3	4/3
shape	=	circle	2	1	1	1	4/3	2/3
shape	=	square	1	1	2	1	2/3	4/3
shape	=	triangle	1	1	2	1	2/3	4/3

Equation 2.7.

$$\left.\begin{aligned} C_{P_{i+1}} &\leftarrow C_{P_i} + \mathit{Intensity} \\ I_{P_{i+1}} &\leftarrow I_{P_i} + (1 - \mathit{Intensity}) \end{aligned}\right\} \text{ if } \mathit{Example}$$

$$\left.\begin{aligned} I_{N_{i+1}} &\leftarrow I_{N_i} + \mathit{Intensity} \\ C_{N_{i+1}} &\leftarrow C_{N_i} + (1 - \mathit{Intensity}) \end{aligned}\right\} \text{ otherwise} \tag{2.7}$$

Again, in the limit Equation 2.7 reduces to the simpler technique of simply retaining matching counts. If a value intensity is one and this is a nonexample, then only the I_N count is changed. Conversely, if the intensity is zero, only the C_N count is modified. As with the generalization of Equation 2.3, the simple spatial learning example of Section 3.3 will demonstrate its use. However, for most of the tasks explicated here, the simpler direct-count method will suffice.

The matching process utilizes not only the weights attached to each concept description element, but it is pre-biased by the prior odds in favor of a positive example. This quantity may also be computed from counts of these matching event types by dividing the sum of positive evidence for any element by the sum of the negative evidence for that same element.

$$\mathit{Odds}(\mathit{Example}) = \frac{C_P + I_P}{I_N + C_N} \tag{2.8}$$

Over the course of time, STAGGER will have processed a large number of examples, and the corresponding counts of matching types will also grow. In order to retain the count's semantics and to avoid arbitrarily large counts, the matching counts (C_P, I_N, I_P, and C_N) are decayed slightly prior to processing each example. Decaying discards some of the experience reflected in the counts; if the decay is too large, records of relevant correlations may be lost.

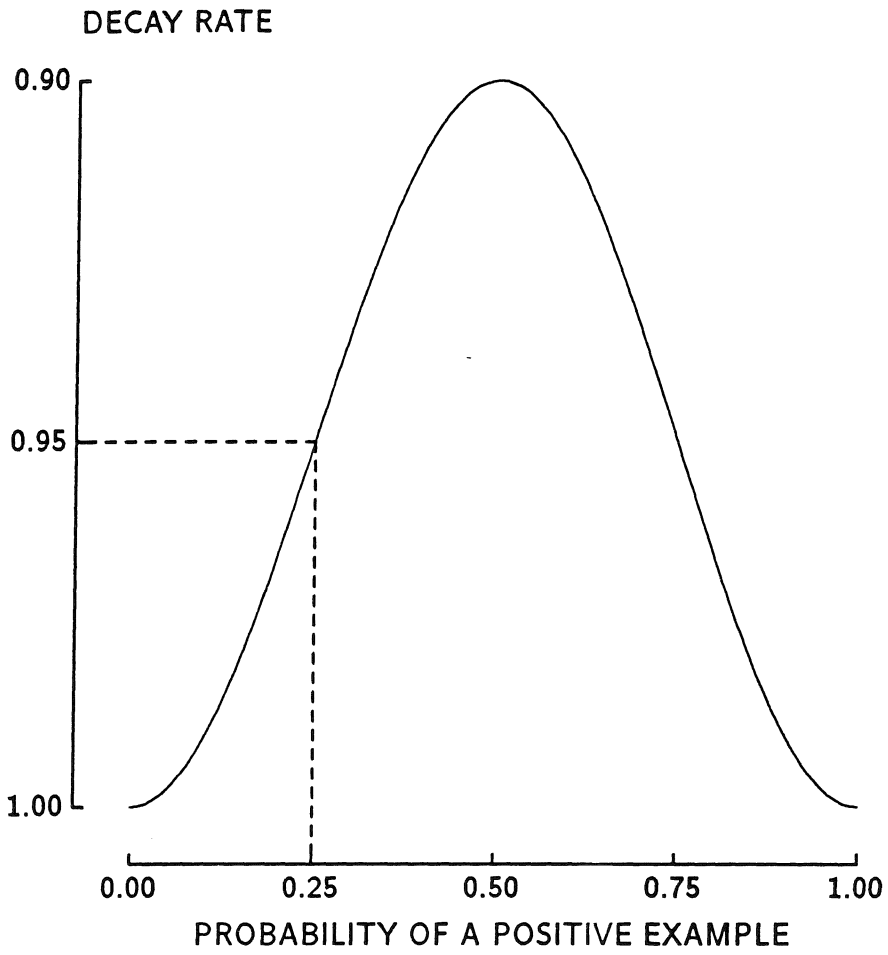

Figure 2.5: Count decay as a function of the likelihood of a positive example.

On the other hand, a tiny decay leaves unnecessarily large counts. To balance these extremes, the specific decay rate is a function of the distribution of "relevant" events. If almost all of the examples are positive, the decay should be minimal, so that some record of negative examples is retained. Similarly, if the likelihood of a positive example is small, the decay is also minimal, and the counts will retain the effects of positive examples. However, when the examples are evenly distributed between positive and negative, the counts can safely decay at a maximal rate. Figure 2.5 depicts decay as a function of the probability of a positive example. For instance (referring to Figure 2.5), if STAGGER estimates that the prior probability of a positive example is 0.25, then each count will be replaced by 95% of its previous value. The important property of the function in Figure 2.5 is that decay is minimal at low and high probabilities and is maximal at $p = 0.50$. This results in the highest retention of experience when the examples are skewed and an accurate depiction of recent experience when they are even.

Decaying counts lends some insight to the importance of the decision to initialize the

matching counts to one. This policy embodies an assumption about the variability of the initial examples to be processed. Consider the extreme cases. If the counts are initialized to 1,000,000 then the first few examples have little effect on the element weights; incrementing a counter would make a small change in its absolute value. On the other hand, if the counts are initialized to 1/1,000,000 then adding one to a counter would increase its size by six orders of magnitude. As an assumption about example variability, initializing to large counts tolerates more noisiness, since it gives little credit to early examples. Conversely, small initial counts have the effect of assigning importance to the first examples, implicitly assuming that they are all representative of the concept. Over time, the decision of initial count value may have little effect, for as the counts are decayed, prior experience, including the initial count values, is discarded to make room for current matching records.

Decaying the matching counts does not change the value of the Bayesian weights, yet it does change their susceptibility to change. The former statement follows because the weights are a ratio of the counts; multiplying each value by a constant does not change the resulting computation. This multiplication does enhance the effect resulting from incrementing one of the counts, for as the existing counts become smaller through decay, incrementing a counter has an increasing effect. This phenomenon is depicted in Figure 2.6 for a typical set of counts and the resulting LS weight. Notice that as this set of counts is successively decayed, incrementing one of the counts has a successively larger effect on the computed weight's value. This effect is diminished with increasing amounts of prior experience (larger counts), and appears consistently across both Bayesian weights and a large range of values. At a higher level, the decayed counts reflect primarily recent experience. The span of experience retained is a function of the rate of decay, which is in turn a function of the distribution of positive and negative examples.

In summary, then, by keeping counts of the matching event types between patterns and examples, it is possible to compute the Bayesian weights for those elements. A small decay is introduced to limit the growth of the counts. This decay is a function of the likelihood of a positive example. In the following subsection I describe an alternative method for adjusting the elements' weights that is not based on explicitly counting each matching type, but instead directly estimates the conditional probabilities involved.

2.4.2 Bayesian weighting by comparing matching types

Rather than count each type of match between examples and concept description element patterns, the Bayesian weights may be adjusted by directly estimating the conditional probabilities. Following a technique used by Hampson (1983) and Hampson and Kibler (1983), these estimates are incrementally modified according to the most recent type of matching event. The LS and LN weights are then computed from the ratio of the relevant conditional

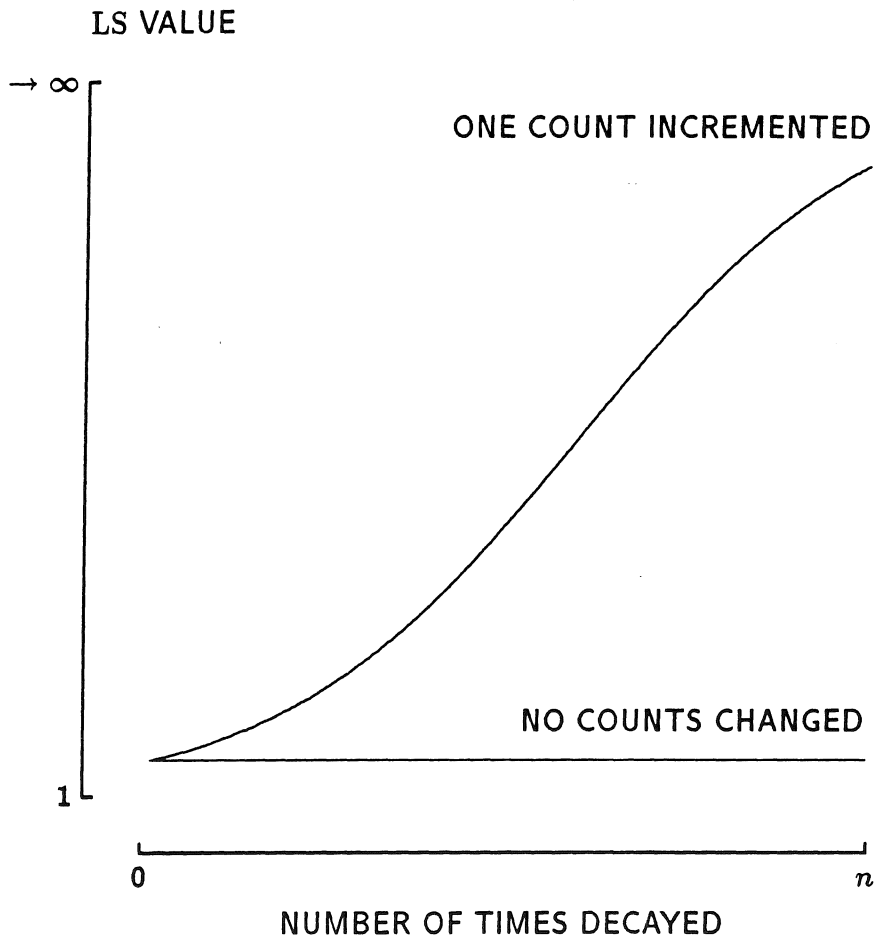

Figure 2.6: The effect of decay on weights and their ability to change.

probabilities.

Incrementally modifying a probability is based on the observation that small adjustments, if made in the proper direction, will converge on the desired value. Because these values are probabilities, each of the adjustments will be scaled by the previous value. For instance, consider estimating the probability of a positive example. After each positive example, the estimate should be increased. Because the estimate should never become greater than one (since it is a probability), it is increased by one minus the estimate, or $p(Example)_{i+1} \leftarrow p(Example)_i + r[1 - p(Example)_i]$ for some learning rate parameter $r \in [0,1]$. If all of the examples were positive, the estimate would soon approach one. On the other hand, after each negative example, the estimate is reduced, and because it should not become less than zero, the estimate is subtracted from itself, or $p(Example)_{i+1} \leftarrow rp(Example)_i - p(Example)_i$. Following an extended sequence of negative examples, the estimate approaches zero.

$$p(Example)_{i+1} \leftarrow p(Example)_i + \begin{cases} r \times [1 - p(Example)_i] & \text{if } Example \\ r \times [-p(Example)_i] & \text{otherwise} \end{cases} \tag{2.9}$$

This equation is essentially the Widrow-Hoff or LMS rule utilized in a number of simple weight models (Duda & Hart, 1973). Its key distinction from the Perceptron rule (Rosenblatt, 1958) is that the latter only modifies weights in response to classification errors. For Equation 2.9 and its relatives, all inputs influence weight values.

The parameter r in Equation 2.9 may be characterized as a learning rate, for setting it equal to one results in rapid changes in the estimate, while setting it to zero results in no changes whatsoever. Like the decay rate for the count-based method, r can also be made a function of the inputs to the learning mechanism. Mackintosh (1975) has demonstrated that by doing so, a simple model can account for a number of subtle learning effects in classical conditioning.

Estimates of the conditional probabilities used in the definition of LS and LN may be computed in a similar manner. Following a positive example, the value for $p(Match|Example)$ will be incremented if the pattern matches, or decremented if it does not. After a negative example, the value for $p(Match|\neg Example)$ is updated. Equation 2.10 lists update rules for the two conditional probabilities used to calculate LS. It is not necessary to maintain a separate estimate for $p(\neg Match|Example)$ and $p(\neg Match|\neg Example)$ since they are algebraically equivalent to $1 - p(Match|Example)$ and $1 - p(\neg Match|\neg Example)$ respectively.

Table 2.7: Initial elements and probability estimates.

PATTERNS			PROBABILITIES		WEIGHTS	
ATTRIBUTE		VALUE	$p(M\|E)$	$p(M\|\neg E)$	LS	LN
size	=	small	0.5	0.5	1	1
size	=	medium	0.5	0.5	1	1
size	=	large	0.5	0.5	1	1
color	=	red	0.5	0.5	1	1
color	=	green	0.5	0.5	1	1
color	=	blue	0.5	0.5	1	1
shape	=	circle	0.5	0.5	1	1
shape	=	square	0.5	0.5	1	1
shape	=	triangle	0.5	0.5	1	1

$$p(M|E)_{i+1} \leftarrow p(M|E)_i + \begin{cases} r \times [1 - p(M|E)_i] & \text{if } Example \wedge Matched \\ r \times [-p(M|E)_i] & \text{if } Example \wedge \neg Matched \end{cases}$$

$$p(M|\neg E)_{i+1} \leftarrow p(M|\neg E)_i + \begin{cases} r \times [1 - p(M|\neg E)_i] & \text{if } \neg Example \wedge Matched \\ r \times [-p(M|\neg E)_i] & \text{if } \neg Example \wedge \neg Matched \end{cases} \quad (2.10)$$

These probability estimates are initialized at 1/2, though it can be shown that any value will suffice. Table 2.7 lists the initial concept description elements and conditional probability estimates for simple objects.

As with the count-based method, the weights are easily modified following matching. For instance, processing a positive example, a **small, red circle**, results in the concept description listed in Table 2.8. Note that slight changes in the conditional probability estimates result in larger changes in the LS and LN weights.

It is not necessary to keep explicit count of each pattern's matching events in order to compute LS and LN. An alternative method draws on work by Hampson (1983) and Hampson and Kibler (1983) to maintain estimates of $p(Matched|Example)$ and $p(Matched|\neg Example)$. The Bayesian LS and LN weights are then computed from these conditional probabilities. This method does require an external parameter that can be made a function of the input. In the following section I make a brief comparison of the two methods, and in doing so discuss a criticism levied at Bayesian representations by Shafer (1976).

Table 2.8: Concept elements after processing one example.

PATTERNS			PROBABILITIES		WEIGHTS	
ATTRIBUTE		VALUE	$p(M\|E)$	$p(M\|\neg E)$	LS	LN
size	=	small	0.67	0.5	1.33	0.67
size	=	medium	0.33	0.5	0.67	1.33
size	=	large	0.33	0.5	0.67	1.33
color	=	red	0.67	0.5	1.33	0.67
color	=	green	0.33	0.5	0.67	1.33
color	=	blue	0.33	0.5	0.67	1.33
shape	=	circle	0.67	0.5	1.33	0.67
shape	=	square	0.33	0.5	0.67	1.33
shape	=	triangle	0.33	0.5	0.67	1.33

2.4.3 Comparison of the two Bayesian weight adjustment methods

By keeping explicit counts of matching events, it is easier to represent the accumulation of evidence.[6] This representational ability allows distinguishing between a relatively new correlation and one that is the result of extensive experience. In the count-based method, accumulated evidence is reflected in the magnitude of the counters, implicitly representing experience. This results in a differential inclination to change a weight value, as depicted in Figure 2.6. Conversely, if only estimates of the relevant probabilities are maintained, additional computations must be added to represent experience. One approach is to make the learning rate parameter a function of the input and the system's performance as a whole (Mackintosh, 1975), yet the resulting dependency is complex and difficult to understand completely. Similarly, Equation 2.10 could be modified to include a second learning rate to reflect the contribution of previous experience. If the parameter r_1 were set to i and r_2 set to $i+1$, then Equation 2.11 (shown below) mirrors the accumulation of evidence by the count-based method (Equation 2.6).[7]

[6]On the other hand, one advantage of the probability estimating technique (Subsection 2.4.2) is that by avoiding the explicit counts of the other method (Subsection 2.4.1), storage requirements are reduced by a constant factor. This is only a slight advantage since it does not affect the asymptotic space requirements of STAGGER as a whole.

[7]This observation is due to Paul Smolensky.

$$p(M|E)_{i+1} \leftarrow r_1 \times p(M|E)_i + \begin{cases} r_2 \times [1 - p(M|E)_i] & \text{if } E \wedge M \\ r_2 \times [-p(M|E)_i] & \text{if } E \wedge \neg M \end{cases}$$

$$p(M|\neg E)_{i+1} \leftarrow r_1 \times p(M|\neg E)_i + \begin{cases} r_2 \times [1 - p(M|\neg E)_i] & \text{if } \neg E \wedge M \\ r_2 \times [-p(M|\neg E)_i] & \text{if } \neg E \wedge \neg M \end{cases} \quad (2.11)$$

Shafer (1976) points out that Bayesian probabilistic methods frequently have difficulty representing accruing experience. In particular, he notes that it is important to represent not only probability but also the belief in that probability, or the meta-probability. Shafer specifically cites the inability of many Bayesian decision models to adequately represent ignorance since they cannot distinguish between a naive belief that $p = 1/2$ (for some binary choice) and a confident assertion that $p = 1/2$. Experimental subjects are able to distinguish between these situations, for they actively investigate the situation when naive, but act disinterested after concluding that randomness is at work (Siegel & Domjan, 1971). I develop this idea later in Section 7.3.

Using the first method, keeping counts of event types and then computing the Bayesian weights from them, allows one to distinguish between a naive state of understanding and experience with a random world. In the former case, where no examples have been processed, the counts are equal to one and therefore LS = LN = 1. On the other hand, having seen a large number of random examples, the counts are evenly large, and LS $\approx$ LN $\approx$ 1. By counting the matching types, the first method explicitly represents belief through a function of the counts (weight) and implicitly represents degree of belief through the magnitude of the counts. The resulting distinction is that in the naive state the weights are easily changeable; whereas in the the experienced situation the weights resist modification. Again, this property is illustrated in Figure 2.6 and is discussed in Section 7.3.

The second method estimates the relevant conditional probabilities and adequately represents belief by computing the weights. Without subsequent modification, the estimates themselves do not represent the degree of belief, for the naive chance belief and experienced chance belief have identical representations: $p(Matched|Example) = 1/2$, $p(Matched|\neg Example) = 1/2$. As I mentioned earlier, it is possible to make the learning rate parameter a function of the system's performance behavior and thereby allow an implicit representation of experience. However, this method relies on a larger number of free parameters, and is therefore more difficult to explicitly test and validate. Balancing this complexity against a relatively small space savings leads me to prefer the count-based method for computing the Bayesian weights. It is this method which is used by STAGGER.

The ability to represent ignorance, as Shafer puts it, is not a strictly academic question. Several of the learning situations that arise in subsequent chapters illustrate the advantages

of a method that relies on an accumulation of prior experience to tolerate inconsistencies.

2.5 Overview

STAGGER represents concepts as a collection of individual elements. Each of these elements is composed of a pattern, which enumerates specific attribute-values, and a pair of weights, which capture the necessity and sufficiency of the pattern for the concept. The matcher uses these weights to classify new examples, and they are easily modified by maintaining the pattern's matching history. There are at least two methods for modifying these weights, and STAGGER uses one based on counting each of the matching types for each pattern.

There are several attractive properties to this type of concept representation. It naturally allows for partial matching between examples and the concept, since each element contributes its estimate to the overall expectation. Basing element weights on a statistical computation allows tolerance to noisy examples, as Section 6.2 illustrates. Further, by distributing the concept description across a number of elements, learning for each may occur in parallel. In addition to the potential increases in speed, learning is tolerant of unknown values, for some element weights may be adjusted even when others cannot (Section 6.4). This modularity of change is also useful if the concept's definition changes over time (Section 6.3).

In this chapter I discussed STAGGER's representation of concepts, the matching process, and a simple form of Bayesian weight adaptation. In the following chapter I present a number of examples of the system's operation and motivate the need for the ability to add additional patterns to the concept description. Though the techniques appear to be complex, they turn out to be relatively straightforward in practice.

Chapter 3

Examples of Bayesian Weight Learning

3.1 Introduction

The matching process and Bayesian weight adjustment method presented in the previous chapter may seem prohibitively complex. They are actually straightforward, and in this chapter I describe the operation of STAGGER in a detailed, step-by-step manner. The forum for this description is a series of four examples. The first shows the acquisition of a conjunctive concept and illustrates the details of processing each instance. The simple weight adaptation method allows STAGGER to correctly classify novel objects by forming a reasonable concept prototype. The second example is the longest and deals with a simple navigation task drawn from Barto and Sutton (1981). It illustrates the flexibility and generality of the method because the multiple concepts to be learned are real-valued and two-dimensional. Moreover, this example affords a natural discussion of the propensity of concepts to change over time, a difficulty that STAGGER easily surmounts. The third example depicts the adaptive retrieval of documents given a set of descriptive keywords. Applying STAGGER to the problems of indexing and retrieving documents within this domain illustrates its effectiveness from and immunity to poor initial encodings. Finally, in the fourth section, I present a comprehensive, artificial task in order to objectively assess the capabilities and limitations of the Bayesian weight adjusting method. The concepts which the method is unable to learn identify a deficiency in the approach: it is unable to assign predictive values to combinations of attribute-values. Together with motivations derived from the individual examples, this serves as the impetus for the development of the Boolean chunk learning method I describe in Chapter 4.

Table 3.1: Initial concept description elements.

Patterns			Counts				Weights	
Attribute		Value	C_P	I_N	I_P	C_N	LS	LN
size	=	small	1	1	1	1	1	1
size	=	medium	1	1	1	1	1	1
size	=	large	1	1	1	1	1	1
color	=	red	1	1	1	1	1	1
color	=	green	1	1	1	1	1	1
color	=	blue	1	1	1	1	1	1
shape	=	circle	1	1	1	1	1	1
shape	=	square	1	1	1	1	1	1
shape	=	triangle	1	1	1	1	1	1

3.2 A simple classification task

The processes described in the previous chapter may seem forbiddingly complex, as they involve decay rates, counter updates, and the calculation of conditional probabilities. In practice, these techniques turn out to be relatively straightforward. Their effectiveness at forming concept descriptions is not the result of some mystical mathematics, but rather arises in an intuitive manner from each of the instances processed. In this section I present a modest example describing the operation of the weight modification procedures within the confines of a simple set of objects. The concept to be learned is conjunctive, and STAGGER is able to form an accurate prototypical description that allows it to properly classify novel objects.

3.2.1 Initializing the concept description

The simple objects from which the concept is formed are describable in terms of their size, color, and shape. Initially, STAGGER has a concept description formed from an element for each of the possible values for each of the attributes: size = large, size = medium, size = small, color = red, ... Table 3.1 (which is a duplication of Table 2.5) lists these elements, their initial counts, and Bayesian weights. Each of the matching counts and the LS and LN weights are initialized to one.

As Figure 3.1 depicts, these concept descriptions have equal contribution to expectation. This copy of Figure 2.1 denotes each of the concept description elements with a small, labeled, outer circle from which emanate two lines. The thickness of these lines correspond to the magnitude of each element's LS and LN weights.

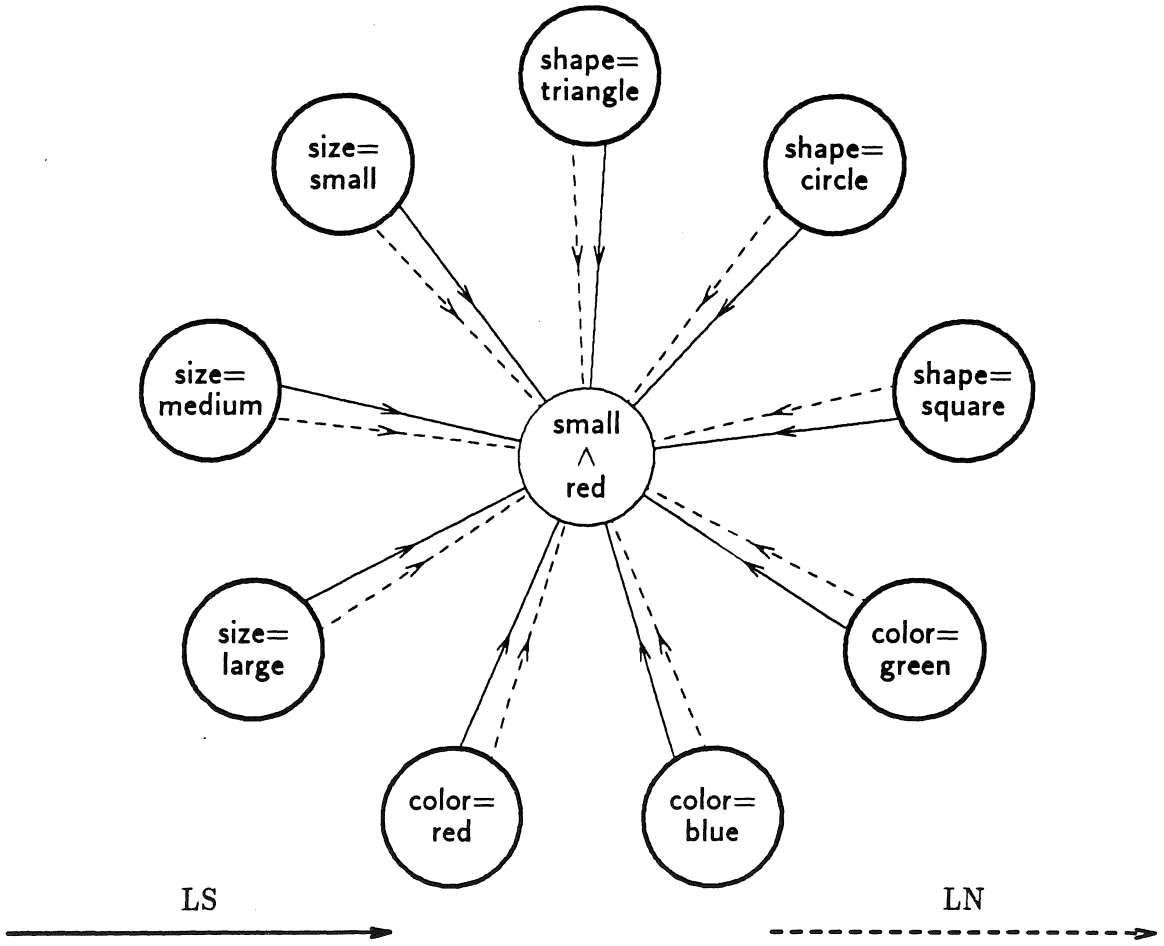

Figure 3.1: Depiction of initial concept description.

Table 3.2: Matches between small, red circle and initial concept description.

Matched Patterns			LS	Unmatched Patterns			LN
size	=	small	1	size	=	medium	1
				size	=	large	1
color	=	red	1	color	=	green	1
				color	=	blue	1
shape	=	circle	1	shape	=	square	1
				shape	=	triangle	1
		$\prod =$	1			$\prod =$	1

3.2.2 Processing the first example

The elements' weights are used to generate a holistic expectation of the identity of examples. Specifically, for each of the elements that matches with the example, the LS weight is multiplied in, and for those that do not match, the LN weight is used. Taken together with the prior expectation that this example is likely to be positive, these values yield an indication of whether the example is positive or not. For instance, consider matching the initial concept description above with a new example: a **small, red circle.** The first pattern matches. The second pattern does not, and so on. Table 3.2 lists the matched patterns against the unmatched ones along with their relevant weights.

The prior expectation of a positive example is also needed to complete the matching computation. Given the initial counts in Table 3.1, Equation 2.8 becomes:

$$\begin{aligned} Odds(Example) &= (C_P + I_P)/(I_N + C_N) \\ &= (1+1)/(1+1) \\ &= 1 \end{aligned}$$

Combining the LS and LN weights with this prior expectation in Equation 2.3 we have:

$$\begin{aligned} Odds(Ex|Features) &= Odds(Ex) \times \prod_{\forall Matched} \text{LS} \times \prod_{\forall Unmatched} \text{LN} \\ &= 1 \times 1 \times 1 \\ &= 1 \end{aligned}$$

This is a relatively uninteresting case because all of the weights are unbiased, and the prior odds of a positive example is one. Converted into a probability ($p = odds/[1 + odds]$), the expectation is $p = 0.5$ that a small, red circle is a positive example. This ambiguous matching score is neither high nor low, forcing STAGGER to randomly guess whether this is a positive or negative example. Regardless of the guess, the object is actually an example of the concept "a small and red object of any shape."

Table 3.3: Concept description elements after processing one example.

Patterns			Counts				Weights	
Attribute		Value	C_P	I_N	I_P	C_N	LS	LN
size	=	small	1.90	0.90	0.90	0.90	1.36	0.64
size	=	medium	0.90	0.90	1.90	0.90	0.64	1.36
size	=	large	0.90	0.90	1.90	0.90	0.64	1.36
color	=	red	1.90	0.90	0.90	0.90	1.36	0.64
color	=	green	0.90	0.90	1.90	0.90	0.64	1.36
color	=	blue	0.90	0.90	1.90	0.90	0.64	1.36
shape	=	circle	1.90	0.90	0.90	0.90	1.36	0.64
shape	=	square	0.90	0.90	1.90	0.90	0.64	1.36
shape	=	triangle	0.90	0.90	1.90	0.90	0.64	1.36

Given this positive example, STAGGER alters the weights attached to each element in order to accurately reflect this experience. First, each of the matching counts are decayed by a small amount based on the probability of a positive example (refer to Figure 2.5). Converting the prior expectation of a positive example into a probability yields $p(example) = 0.5$ which Figure 2.5 translates into a decay rate of 0.9. Therefore, each of the counts in Table 3.1 are replaced by 0.9 times their current value.

After this general decay, one of the counters for each concept description element is incremented. For this positive example, only the positive counters will be incremented. If an element matched (e.g., size = small), then one is added to its confirming counter C_P. Conversely, if an element did not match (e.g., size = medium), then one is added to its infirming counter I_P. Table 3.3 lists the concept description elements after decaying and updating the counts.

The new LS and LN weights in Table 3.3 are computed using the new matching counts and Equation 2.6. For instance, the weights for the first element (size = small) are:

$$LS = \frac{1.9 \times (0.9 + 0.9)}{0.9 \times (1.9 + 0.9)} = 1.36 \qquad LN = \frac{0.9 \times (0.9 + 0.9)}{0.9 \times (1.9 + 0.9)} = 0.64$$

The increased LS weight of this element is precisely what we might expect after seeing it in a positive example. Its growth is due to the incremented C_P counter. Each of the elements which were matched in this example have larger LS weights, and each of the unmatched elements have smaller ones.

The graphical depiction of the concept description in Figure 3.2 readily reveals the emerging importance of the size = small, color = red, and shape = circle elements.

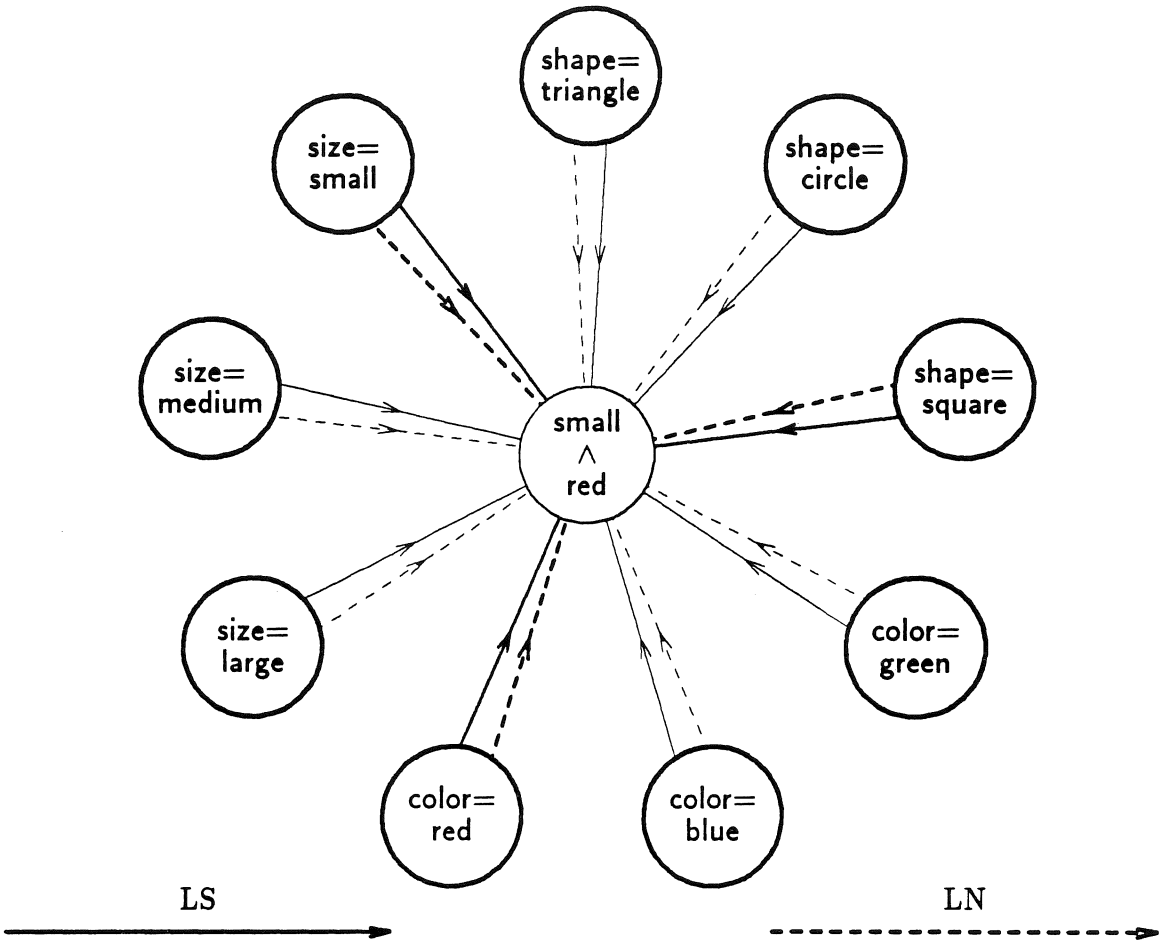

Figure 3.2: Concept description elements after processing one example.

Table 3.4: Matches between medium, red triangle after one prior example.

MATCHED PATTERNS			LS	UNMATCHED PATTERNS			LN
size	=	medium	0.64	size	=	small	0.64
				size	=	large	1.36
color	=	red	1.36	color	=	green	1.36
				color	=	blue	1.36
shape	=	triangle	0.64	shape	=	circle	0.64
				shape	=	square	1.36
		$\prod =$	0.56			$\prod =$	1.40

3.2.3 Processing subsequent examples

The first example was a bit dull because STAGGER had few expectations about it. However, for the next example the matching process is somewhat more interesting. Given a medium, **red triangle** STAGGER computes the list of matchings appearing in Table 3.4.

The expectation of a positive example is higher than it was before. By substituting the counts of the first pattern (size = small) into Equation 2.8 we have:

$$\begin{aligned} Odds(Example) &= (C_P + I_P)/(I_N + C_N) \\ &= (1.90 + 0.9)/(0.9 + 0.9) \\ &= 1.56 \end{aligned}$$

Again, this increase is due to the incremented C_P counter. Using matching Equation 2.3 and this expectation we have:

$$\begin{aligned} Odds(Ex|Features) &= Odds(Ex) \times \prod_{\forall Matched} \text{LS} \times \prod_{\forall Unmatched} \text{LN} \\ &= 1.56 \times 0.56 \times 1.40 \\ &= 1.22 \end{aligned}$$

These are better than even odds (the corresponding probability is $p = 0.55$), so STAGGER (erroneously) expects this example to be positive. Following the steps outlined above, the weights for each concept description element are adjusted by determining the appropriate decay rate, incrementing the negative counters (this is a negative example), and recomputing the weights.

For 18 more examples, STAGGER matches, updates counters, and recomputes weights resulting in the concept description listed in Table 3.5.

By this time, a graphical interpretation of the concept description (Figure 3.3 clearly reveals the salient nature of the size = small and color = red as compared to their fellow elements.

Table 3.5: Concept description elements after processing 20 examples.

Patterns Attribute Value	Counts C_P	I_N	I_P	C_N	Weights LS	LN
size = small	3.48	3.56	0.39	9.05	3.19	0.14
size = medium	0.39	4.73	3.48	7.87	0.27	1.44
size = large	0.39	4.70	3.48	7.90	0.27	1.44
color = red	3.48	3.12	0.39	9.49	3.64	0.13
color = green	0.39	5.90	3.48	6.71	0.21	1.69
color = blue	0.39	3.98	3.48	8.63	0.32	1.32
shape = circle	0.82	4.32	3.06	8.28	0.61	1.20
shape = square	3.06	4.96	0.82	7.65	2.01	0.35
shape = triangle	0.39	3.72	3.48	8.89	0.34	1.28

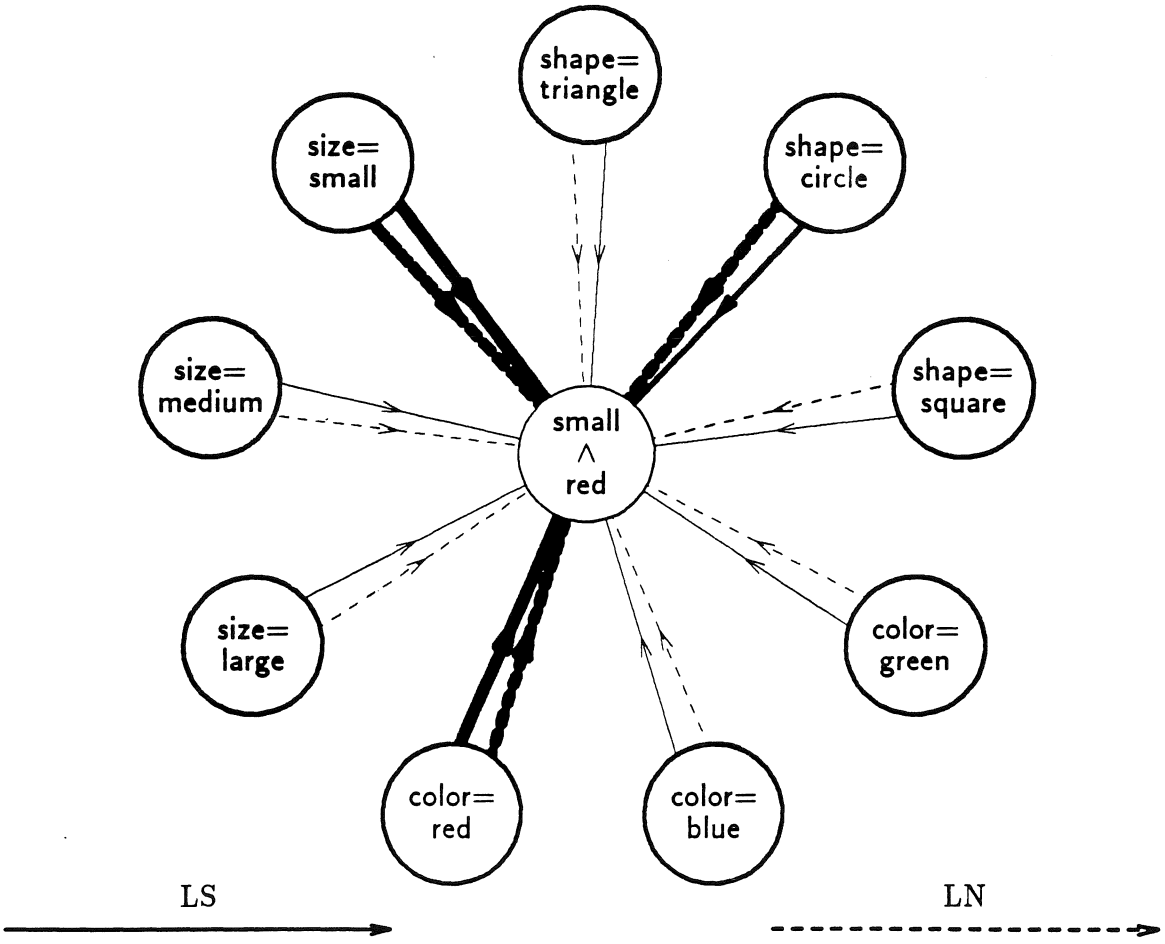

Figure 3.3: Concept description elements after processing 20 examples.

3.2.4 Interpreting the concept description

There are two good points to this concept description. First, the counts result in a relatively accurate expectation of a positive example. Using Equation 2.8 and the first pattern's counts results in an estimation of $odds(example) = 0.31$. The actual odds in favor of a positive example over the set of 20 examples STAGGER has seen thus far is $odds(example) = 0.25$.

Second, the weights reflect the concept. This set of weighted patterns correctly classifies all objects, even the novel ones which STAGGER has not yet seen. Inspecting the individual weights lends credence to this assertion. The LS weights indicate a positive correlation if greater than one, irrelevance if equal to one, and a negative correlation if less than one. The weights of **size = small**, **color = red**, and **shape = square** in Table 3.5 and the thickness of the solid lines in Figure 3.5 indicate that these elements are strongly predictive of positive examples (though **shape = square** somewhat less so). Alternatively, their high LS weight signifies that they are frequently matched in positive examples (their C_P counts are high). For the remaining elements, their weak LS weights indicate that they are poor predictors of positive examples; they are not sufficient for expectation. Examining the LN weights yields a similar set of conclusions. The same three elements that have a strong LS weight also have a strong LN weight. Since LN weights have the opposite interpretation from LS weights the smallness of them for **size = small**, **color = red**, and **shape = square** (thickness of dashed lines in Figure 3.5) indicate that these elements are usually unmatched in negative examples. They are necessary for expectation of the concept.

Interestingly, the LS and LN weights do not indicate equal significance for these predictive elements. For the elements **size = small** and **color = red**, the LN weights indicate a stronger correlation than the LS weights do. Pictorially, the dashed lines are thicker than the solid lines in Figure 3.5 for these two elements. Their imbalance signifies they are more necessary than sufficient. This is how STAGGER distinguishes between representing a conjunction (where each is necessary) and a disjunction (where each is sufficient). In matching, the LN weights operationally have a greater inhibitory effect than the contributory effect arising from the LS weights. This can be seen by comparing the inverse of LN with LS (e.g., $1/0.14 = 7.14$ is greater than 3.19 for the pattern **size = small**) or by noting that if only one of **size = small** and **color = red** is matched, the LN weight of the unmatched one will override the LS weight of the matched one (e.g., $0.14 \times 3.64 = 0.51$ which is less than even odds).

By way of reference, an ideal representation for a conjunctive concept would be for each of the conjuncts to have strong LN weights ($\ll 1$) so that when one conjunct was omitted by an example, expectation would be curtailed by the missing element's small LN weight. Conversely, ideal disjuncts have strong LS weights ($\gg 1$); each is sufficient for concept expectation. If any of these disjuncts is matched in an example, expectation is carried over the threshold.

Table 3.6: Prototypical example.

PATTERNS ATTRIBUTE VALUE	SCORE LS × 1/LN
color = red	3.64 × 1/0.13 = 28
size = small	3.19 × 1/0.14 = 23
shape = square	2.01 × 1/0.35 = 6

3.2.5 Extracting prototypicality

Lurking within this collection of weighted elements is a description of a prototypical example. First sort each of the elements by their Bayesian weights, and then, for each attribute, take the best element. Table 3.6 lists the results for the elements in Table 3.5.

This prototype description reveals the diminutive nature of the slightly predictive **shape = square** element. It is not ranked as highly as either of the other two elements are. This is in consonance with the description of the concept as "small and red with any shape."

Though this distributed, weighted representation is useful for matching, it can be difficult to comprehend. This is a difficulty that STAGGER shares with connectionist models. One of the benefits of the Boolean formation process described in Chapter 4 is that it constructs patterns that have a more unified and readily understandable meaning. With this cooperative method, the description is eventually based on **small** ∧ **red** explicitly.

In review, STAGGER begins learning about a concept with a simple, naive set of concept description elements. After processing each example, all of matching event counters are decayed, and one counter from each element is incremented. New weights are computed from these counts, and they are used to meter matching for the next instance. These simple changes build an effective, prototypical concept description that allows STAGGER to correctly classify objects it has not seen; a useful capability in many tasks.

The general nature of STAGGER goes beyond this simple task, and it is applicable for acquiring a variety of concepts. In the following section I borrow a simple, unsupervised learning task from Barto and Sutton (1981) and show how STAGGER adapts to it.

3.3 Learning about spatial locations

STAGGER is a general purpose learning mechanism. In the previous section I described its operation on a simple concept acquisition task, but it has a wider relevance. STAGGER is applicable to a range of tasks, and in this section I borrow a simple two-dimensional mapping task from Barto and Sutton (1981). Unlike many learning from examples tasks, this one does not involve an *explicit* "teacher." Instead, STAGGER determines from the environment whether

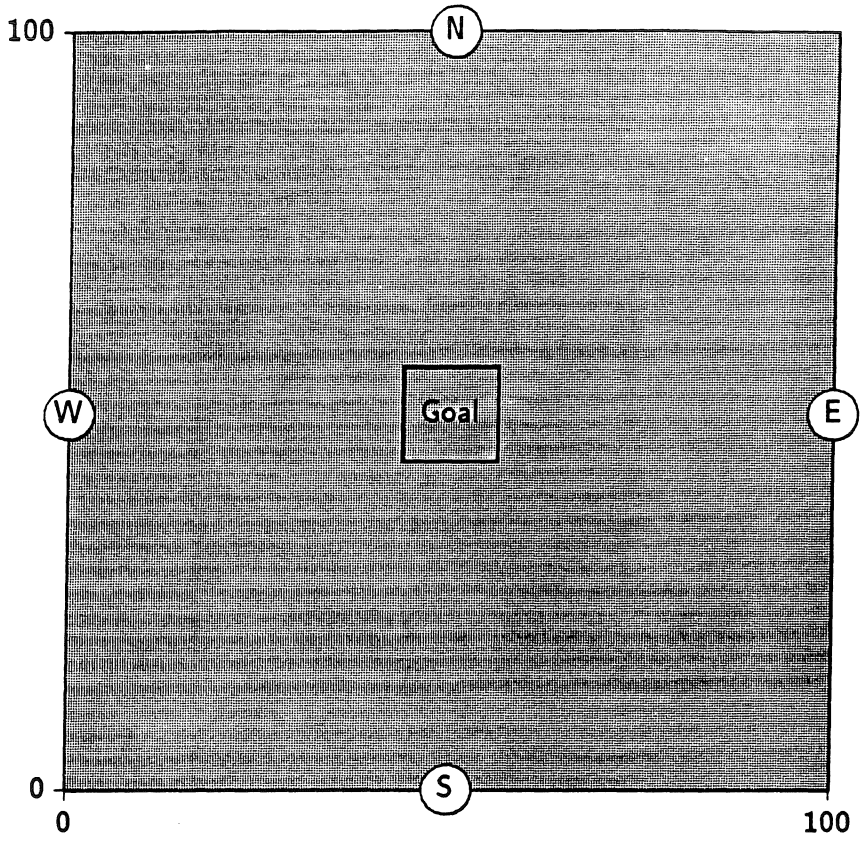

Figure 3.4: A simple two dimensional world.

or not it is making the correct actions, and in doing so, serves as its own teacher. The results of this exercise show that STAGGER is capable of representing a real-valued, two-dimensional concept and that it is tolerant of changes in a concept over time.

3.3.1 Landmark learning

Up to this point, I have described STAGGER as a concept learning program capable of acquiring abstract concepts, but it is also able to learn more concrete concepts like the spatial layout of a simple two-dimensional world. For instance, consider the simple world of Figure 3.4, which contains four distinct olfactory landmarks in a world 100 units square. An additional fifth landmark identifies a goal location.

Each of these landmarks may be sensed through the strength of their distinct odors. The intensity of each smell is inversely proportional to the distance from the landmark, and it ranges from zero to one. Thus, each location in this world is uniquely identified by the intensity of four landmark odors. STAGGER controls a simple organism in this world, directing it to move up, down, left, or right. The task is to move to the desired location from any starting point,

Table 3.7: Initial descriptions of operator preconditions.

Landmarks	Counts C_P	I_N	I_P	C_N	Weights LS	LN
Move Up						
north	1	1	1	1	1	1
south	1	1	1	1	1	1
east	1	1	1	1	1	1
west	1	1	1	1	1	1

	Move Left						Move Right					
north	1	1	1	1	1	1	1	1	1	1	1	1
south	1	1	1	1	1	1	1	1	1	1	1	1
east	1	1	1	1	1	1	1	1	1	1	1	1
west	1	1	1	1	1	1	1	1	1	1	1	1

	Move Down					
north	1	1	1	1	1	1
south	1	1	1	1	1	1
east	1	1	1	1	1	1
west	1	1	1	1	1	1

even if the goal landmark is missing. In order to do this, STAGGER learns which direction to move in terms of the four landmarks; i.e., the preconditions for each of the four directional operators. The change between the previous and current distance to the goal serves as a form of feedback, indirectly causing STAGGER to repeat actions leading to the desired location and avoid those that do not.

The initial concept descriptions for the landmark learning task consist of a set of unbiased elements: one concept for each of the four directional operators and four elements in each for the four landmarks. Table 3.7 lists these initial concept descriptions.

A graphical depiction (Figure 3.5) of these four concepts and their four simple descriptive elements clearly indicates the initially unbiased nature of the operator descriptions, for each of the weights (depicted by lines) are of equal weight.

For the most part, the learning mechanisms as presented in Chapter 2 are adequate for this task. In terms of the previous object concept learning example of Section 3.2, each of the locations in the simple world comprises an example, with the intensity of the landmarks serving as the attribute-value pairs. Instead of acquiring a single object concept, STAGGER attempts to discover appropriate conditions for applying each of the four operators. For each operator, the location may be either a positive or negative example depending on whether or not applying the operator would result in being any closer to the goal location. In practice, STAGGER matches the landmark-intensity pairs against the four operator descriptions. The

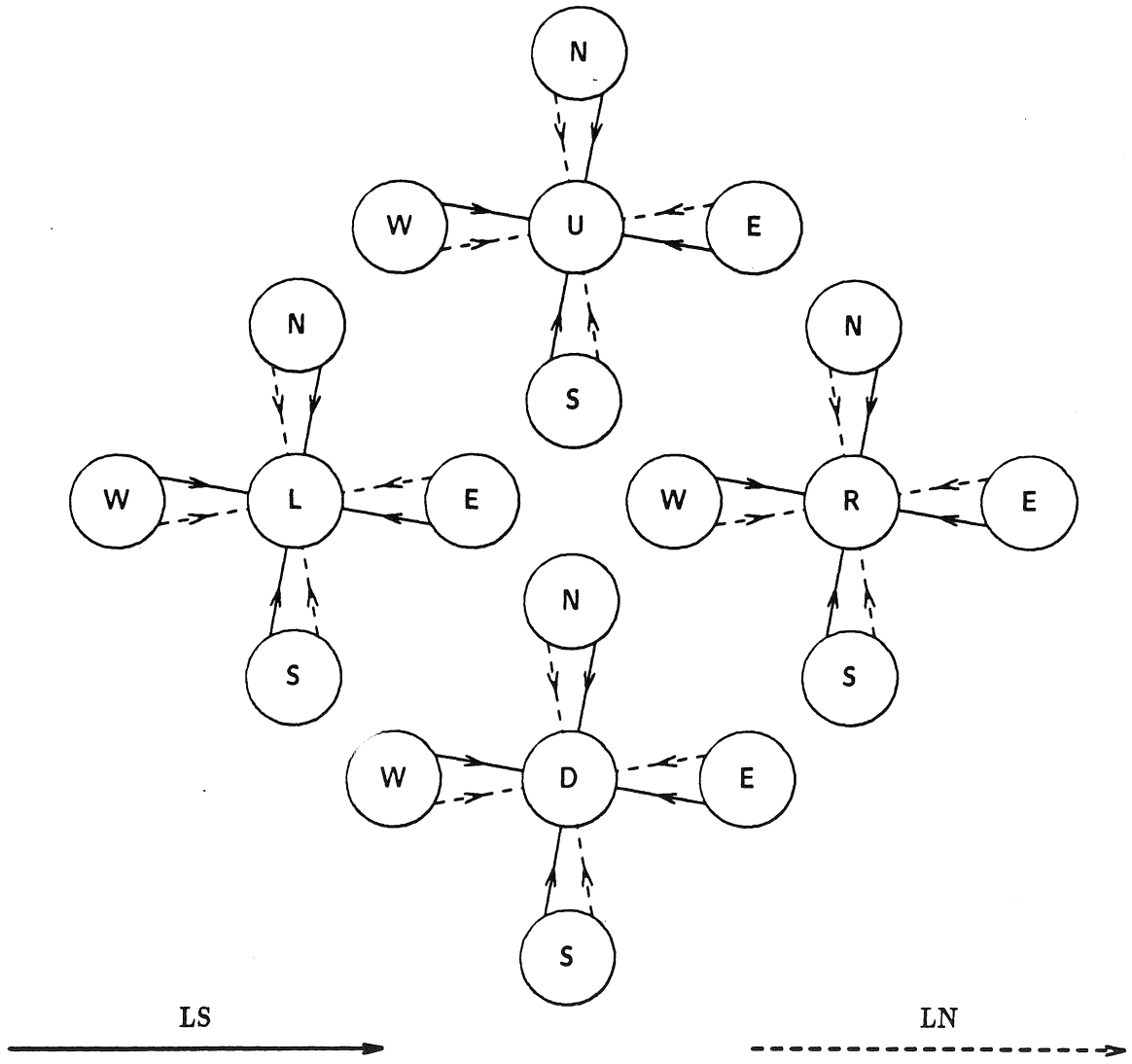

Figure 3.5: Graphical depiction of four concept descriptions.

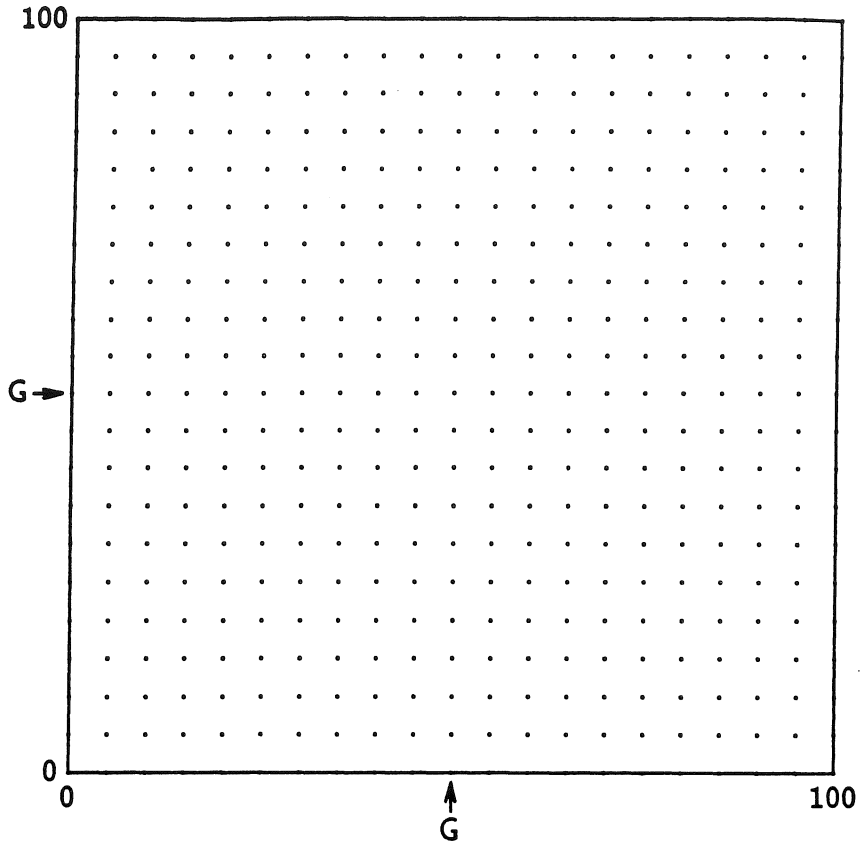

Figure 3.6: Initial operator expectation.

expectation computed from each concept is converted into actions, with the strongest actions taking precedence. If the actions bring the organism closer to the goal, then this instance is identified as positive for the operator that was applied and negative for its opposite.

Matching landmark intensities onto operator preconditions suggests a holistic presentation of the latter. Following Barto and Sutton (1981), we may draw a line for a number of locations to indicate the operators' collective expectation, with the line's direction and length arising from a summation of these expectations. Strongly expected operators swing the line their direction and lengthen it. If all of the operators have equal odds, the line reduces to a dot. Figure 3.6 depicts the operators' cumulative expectation corresponding to an unbiased set of precondition descriptions (see Table 3.7). All of the lines are dots, and this indicates that STAGGER does not initially prefer any of the operators to the others.

Given expectations corresponding to the predicted appropriateness of each of the four operators, STAGGER choses one of the operators to apply. First, the strongest of each pair of opposing operators is selected. Ties are broken via random choice. One of the two orthogonal operators is selected randomly. This latter choice keeps STAGGER from focusing too narrowly

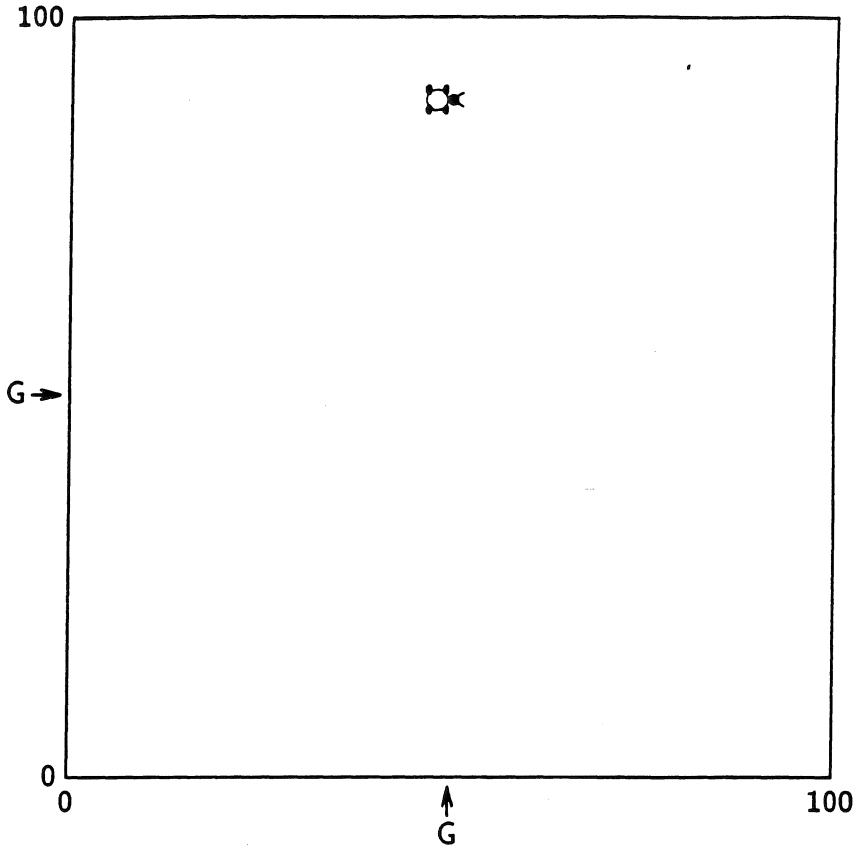

Figure 3.7: An initial starting location.

on the benefits of a single operator, allowing each of the operators' preconditions to be tried and modified.

Given these alterations to STAGGER, let us consider its overall performance for the task of moving to the desired location from any starting point. For instance, assume a starting location of (49,89) as depicted in Figure 3.7.

STAGGER senses each of the four directional landmarks as (N,S,E,W) = (0.92,0.37,0.55,0.56). Note that the North landmark has the greatest intensity since the organism is closest to it. These values are substituted into Equations 2.8 and 2.4 in order to determine the appropriateness of each operator. For instance, for the downward operator we have:

$$
\begin{aligned}
Odds(Downward) &= (C_P + I_P)/(I_N + C_N) \\
&= (1+1)/(1+1) \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
&Odds(Downward|(0.92, 0.37, 0.55, 0.56)) = \\
&= Odds(Downward) \times \prod \begin{bmatrix} 0.92 \times 1 + (1 - 0.92) \times 1 = 1 \\ 0.37 \times 1 + (1 - 0.37) \times 1 = 1 \\ 0.55 \times 1 + (1 - 0.55) \times 1 = 1 \\ 0.56 \times 1 + (1 - 0.56) \times 1 = 1 \end{bmatrix} \\
&= 1 \times 1 \\
&= 1
\end{aligned}
$$

As in the previous example, because the initial concept description is unbiased, initial expectations are evenly divided. The expectation for the other three operators is also *odds* = 1. By referring back to Figure 3.6, we could have predicted this; it shows that at each location the operators' expectations are equal. By default, STAGGER makes a nondeterministic choice. In this example, assume that the downward operator was applied. This results in a new location of (49,88) as depicted in Figure 3.8 which is closer to the desired location of (50,50).

The corresponding matching counts are then updated according to Equation 2.7. Again, considering the downward operator, this instance is a positive example because applying it moved closer to the desired location. Table 3.8 lists the results of substituting the landmark intensities from the previous location in Equation 2.7 for each of the operator preconditions.

For the downward operator, note that both the C_P and I_P counts have been updated, while neither of the other two counts have. Conversely, for the opposing, upward operator, both the I_N and C_N counts have been modified.[1] Figure 3.9 depicts the operators' net expectation following this step. The cumulative effect is that STAGGER is predisposed to repeat the previously successful operator and move downward.

At this new location STAGGER notes that the intensities for the four landmarks are (N,S,E,W) = (0.91,0.38,0.55,0.56). Given the modified concept descriptions above, for the downward operator Equations 2.8 and 2.4 become:

[1]Each of the counts for the upward and downward operators have been decayed, but each count has also been rounded to two decimals for convenience of presentation.

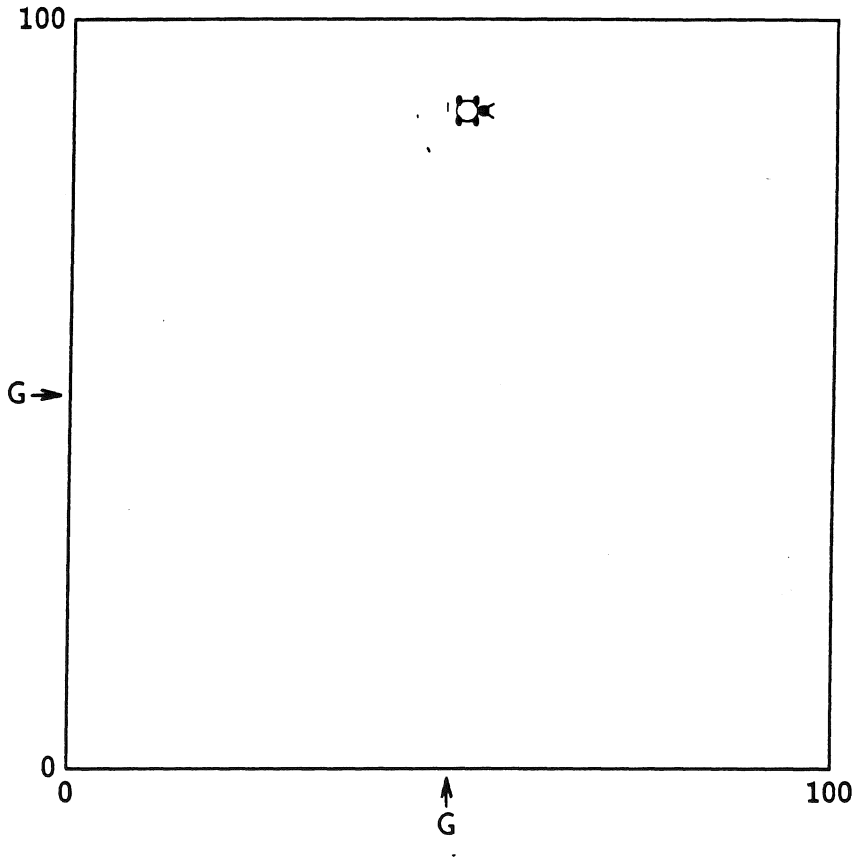

Figure 3.8: Path after 1 move.

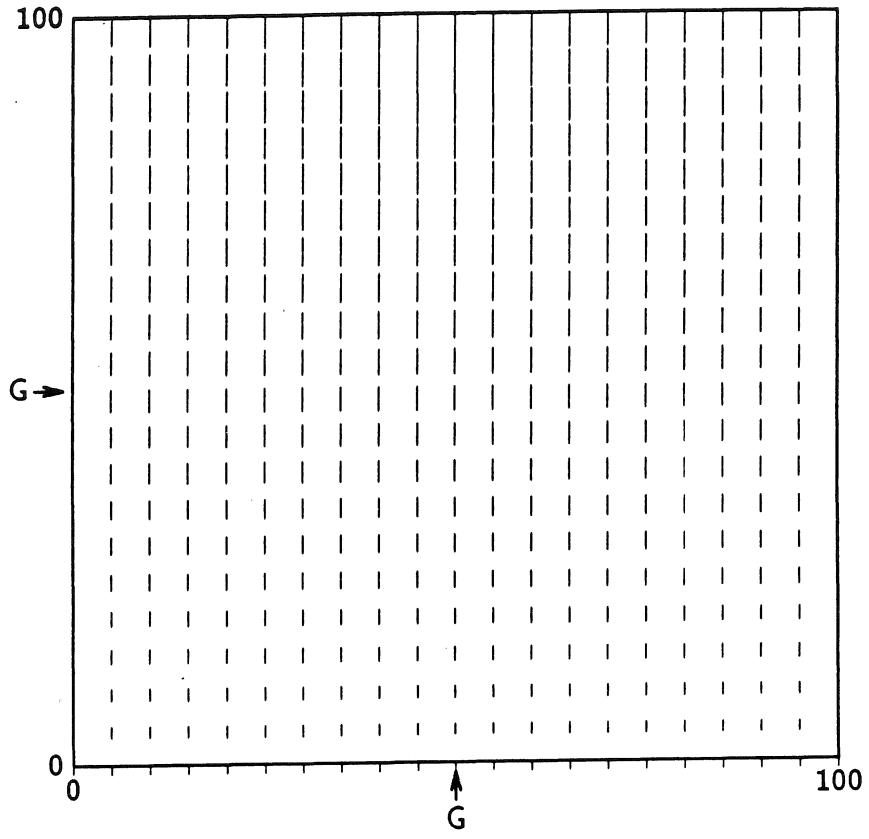

Figure 3.9: Operator expectation after 1 move.

Table 3.8: Descriptions of operator preconditions after 1 move.

LANDMARKS	COUNTS C_P	I_N	I_P	C_N	WEIGHTS LS	LN
MOVE UP						
north	1.00	1.92	1.00	1.08	0.78	1.39
south	1.00	1.37	1.00	1.63	1.09	0.92
east	1.00	1.55	1.00	1.45	0.97	1.03
west	1.00	1.56	1.00	1.44	0.96	1.04

MOVE LEFT						MOVE RIGHT					
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1

MOVE DOWN						
north	1.92	1.00	1.08	1.00	1.28	0.72
south	1.37	1.00	1.63	1.00	0.91	1.09
east	1.55	1.00	1.45	1.00	1.03	0.97
west	1.56	1.00	1.44	1.00	1.04	0.96

$$
\begin{aligned}
Odds(Downward) &= (C_P + I_P)/(I_N + C_N) \\
&= (1.92 + 1.08)/(1.00 + 1.00) \\
&= 1.50
\end{aligned}
$$

$$
\begin{aligned}
&Odds(Downward|(0.91, 0.38, 0.55, 0.56)) = \\
&= Odds(Downward) \times \prod \begin{bmatrix} 0.91 \times 1.28 + (1 - 0.91) \times 0.72 = 1.22 \\ 0.38 \times 0.91 + (1 - 0.38) \times 1.09 = 1.03 \\ 0.55 \times 1.03 + (1 - 0.55) \times 0.97 = 1.01 \\ 0.56 \times 1.04 + (1 - 0.56) \times 0.96 = 1.00 \end{bmatrix} \\
&= 1.50 \times 1.27 \\
&= 1.91
\end{aligned}
$$

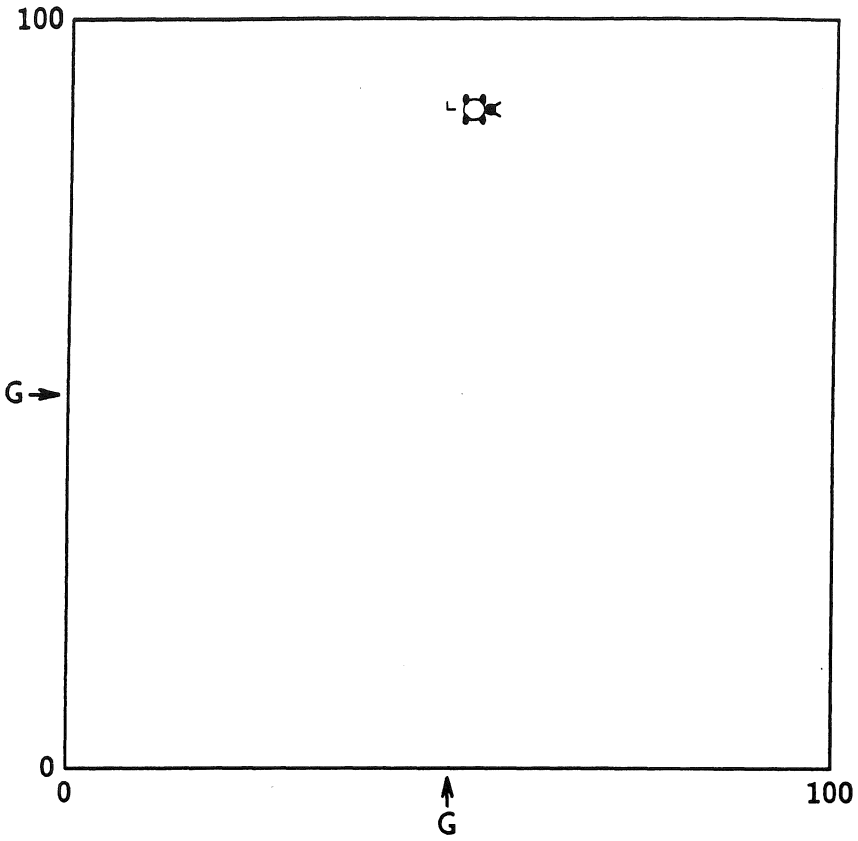

Figure 3.10: Path after 2 moves.

By a similar computation the expectation of the appropriateness for the southward, eastward, and westward operators is *odds* = 0.54, 1.00, 1.00, respectively. The strongest of each pair of opposing operators is chosen, and then a random choice is made between the resulting two options. In this case, STAGGER chooses between moving down and moving left or right (which are equally strong). For illustrative purposes, assume that the rightward operator is applied. Moving right puts the organism at (50,88), which is closer to the goal location of (50,50) as is depicted in Figure 3.10.

This is a positive instance for the rightward operator and a negative one for the leftward operator. Table 3.9 lists the operator precondition descriptions after STAGGER has updated the appropriate counters.

In this case, the C_P and I_P counters have been updated for the right and left operators, while all of the counters for the up and down operators remain unaltered. The resulting modifications to the landmarks' weights change the balance of operator expectation as shown in Figure 3.11.

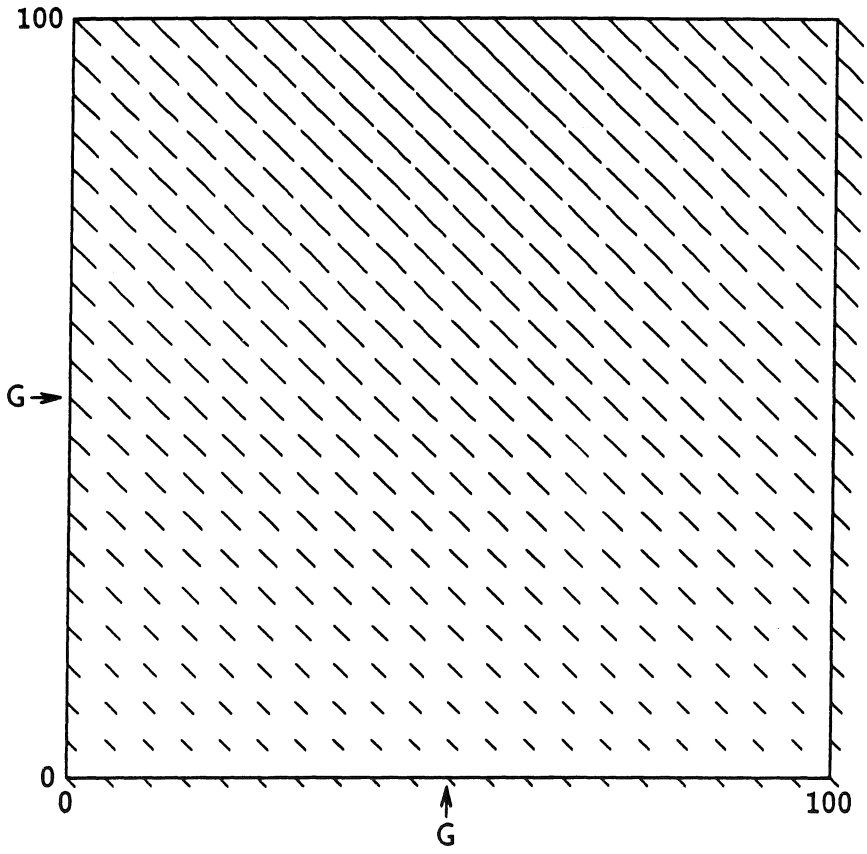

Figure 3.11: Operator expectation after 2 moves.

Table 3.9: Descriptions of operator preconditions after 2 moves.

Landmarks	Counts C_P	I_N	I_P	C_N	Weights LS	LN
Move Up						
north	1.00	1.92	1.00	1.08	0.78	1.39
south	1.00	1.37	1.00	1.63	1.09	0.92
east	1.00	1.55	1.00	1.45	0.97	1.03
west	1.00	1.56	1.00	1.44	0.96	1.04

Move Left						Move Right					
1.00	1.91	1.00	1.09	0.78	1.38	1.91	1.00	1.09	1.00	1.28	0.72
1.00	1.38	1.00	1.62	1.09	0.92	1.38	1.00	1.62	1.00	0.92	1.08
1.00	1.55	1.00	1.45	0.97	1.03	1.55	1.00	1.45	1.00	1.03	0.97
1.00	1.56	1.00	1.44	0.96	1.04	1.56	1.00	1.44	1.00	1.04	0.96

Move Down						
north	1.92	1.00	1.08	1.00	1.28	0.72
south	1.37	1.00	1.63	1.00	0.91	1.09
east	1.55	1.00	1.45	1.00	1.03	0.97
west	1.56	1.00	1.44	1.00	1.04	0.96

After a series of 100 move, evaluate, and update operations, the organism has followed the path depicted in Figure 3.12. Notice that the path overshoots the goal location of (50,50) somewhat. This is because STAGGER has only recently encountered a situation for which the downward operator is not appropriate.

At this point in time the resulting concept descriptions reflect the vertical path followed; STAGGER tends to direct the organism downward beyond the desired location. This is readily apparent in Figure 3.13 and reflects the current experience that STAGGER has had with this simple world: the organism began generally above the goal location and had to move downward to approach it.

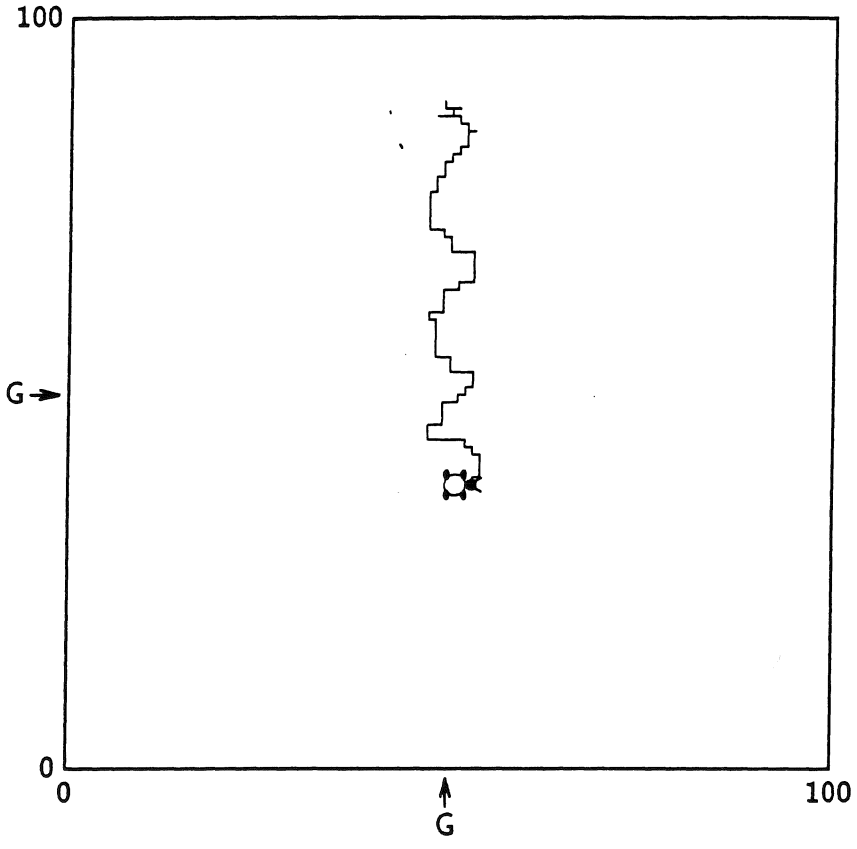

Figure 3.12: Path after 100 moves.

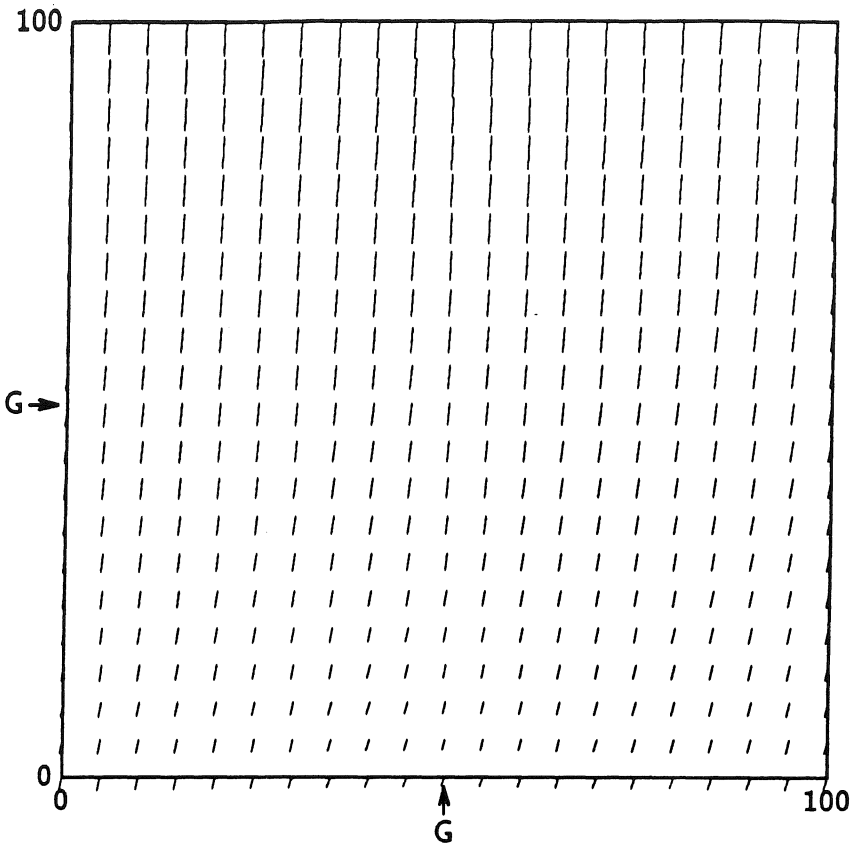

Figure 3.13: Operator expectation after 100 moves.

This set of operator preconditions is inadequate because it favors downward movement. Additional experience corrects this, and the next 100 moves give STAGGER ample opportunity to form suitable precondition concepts for the upward and downward operators. As might be expected, the additional steps carry the organism further downward and then back up to the general location of the goal. Figure 3.14 depicts the downward walk and its reversal back up toward the goal.

These additional moves result in accurate precondition descriptions for both the upward and downward operators. Figure 3.15 depicts cumulative operator expectation after 200 steps. The upward pointing lines near the bottom of the figure indicate that STAGGER will move the organism up if it is too low. Similarly, the downward pointing lines near the top show that the organism will move down if too high.

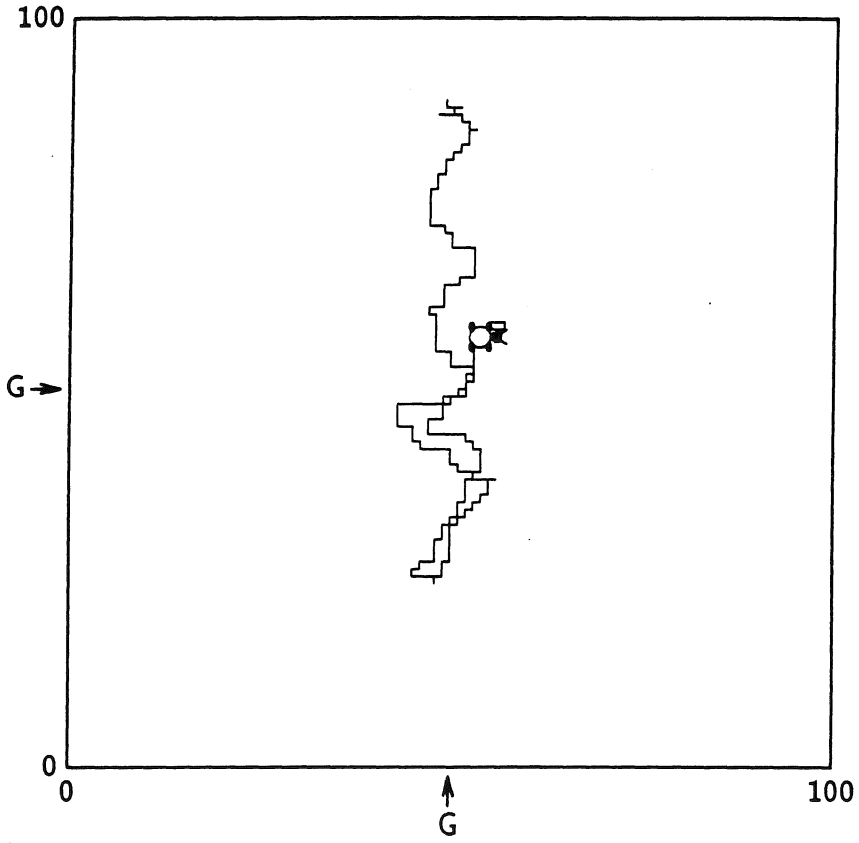

Figure 3.14: Path after 200 moves.

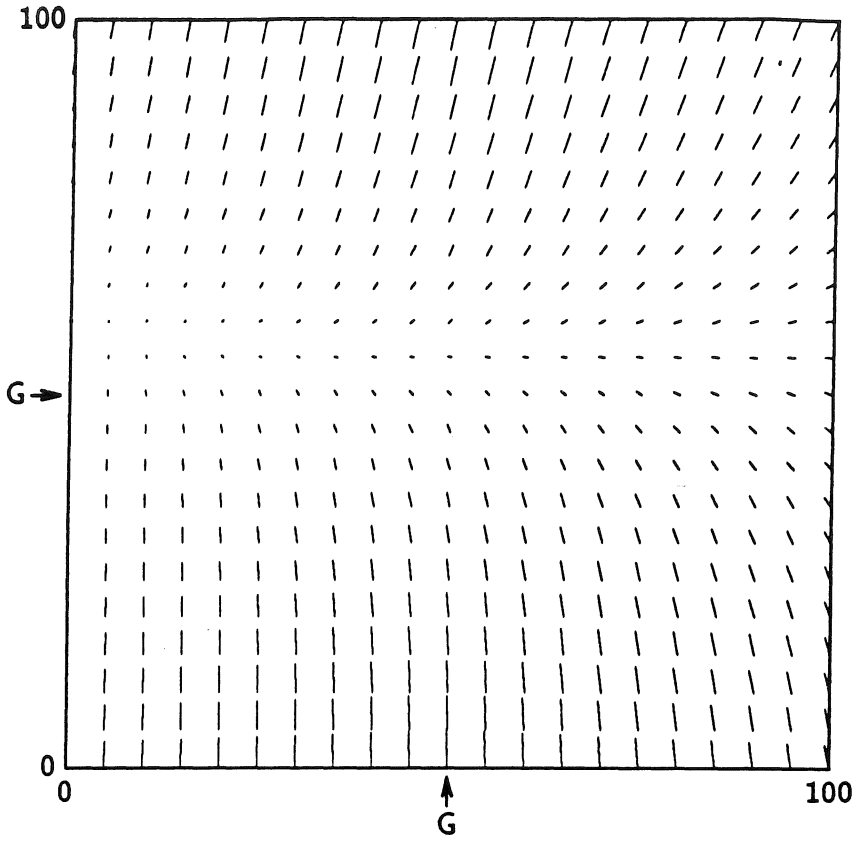

Figure 3.15: Operator expectation after 200 moves.

However, these operator preconditions are not completely correct, for the net operator expectation depicted in Figure 3.15 indicates that the organism will drift off to the left. At this point, STAGGER has had little effective experience with centering the organism horizontally in the space. Additional training rectifies this situation as well. After a total of 500 moves, STAGGER has directed the organism toward the goal location, wandered around it a bit, and has formed an effective precondition description for each of the four operators. Figure 3.16 depicts the last 100 moves. Note that most of the movement is confined to a small area close to the goal location; each step away serves as an effective training example. This implies that learning would be faster if the organism were originally started close to the goal, and informal studies indicate that this is the case. However, it is unrealistic to expect that the learner would be so fortunate, so in each of the experiments of this section, the organism was placed at a random location approximately 40 squares away from the goal.

After 500 moves, the weights are strong and stable. Figure 3.17 reflects the larger weights amassed by the descriptions for the upward and downward operators as compared to those for the leftward and rightward operators. Reading the description intuitively, the thickness

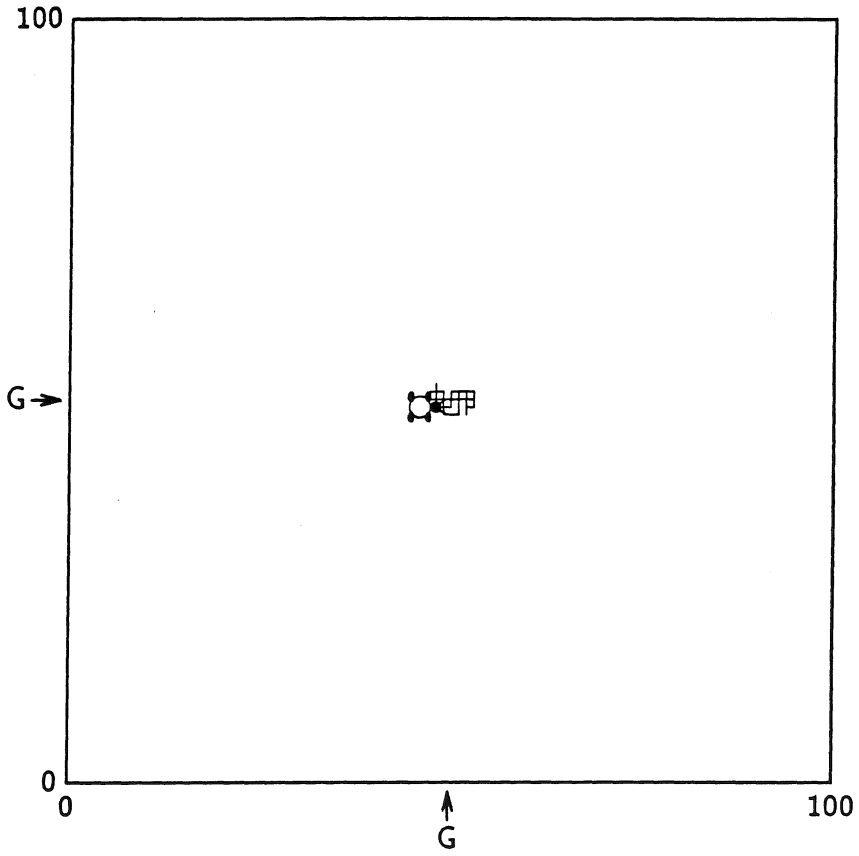

Figure 3.16: Path of the last 100 moves.

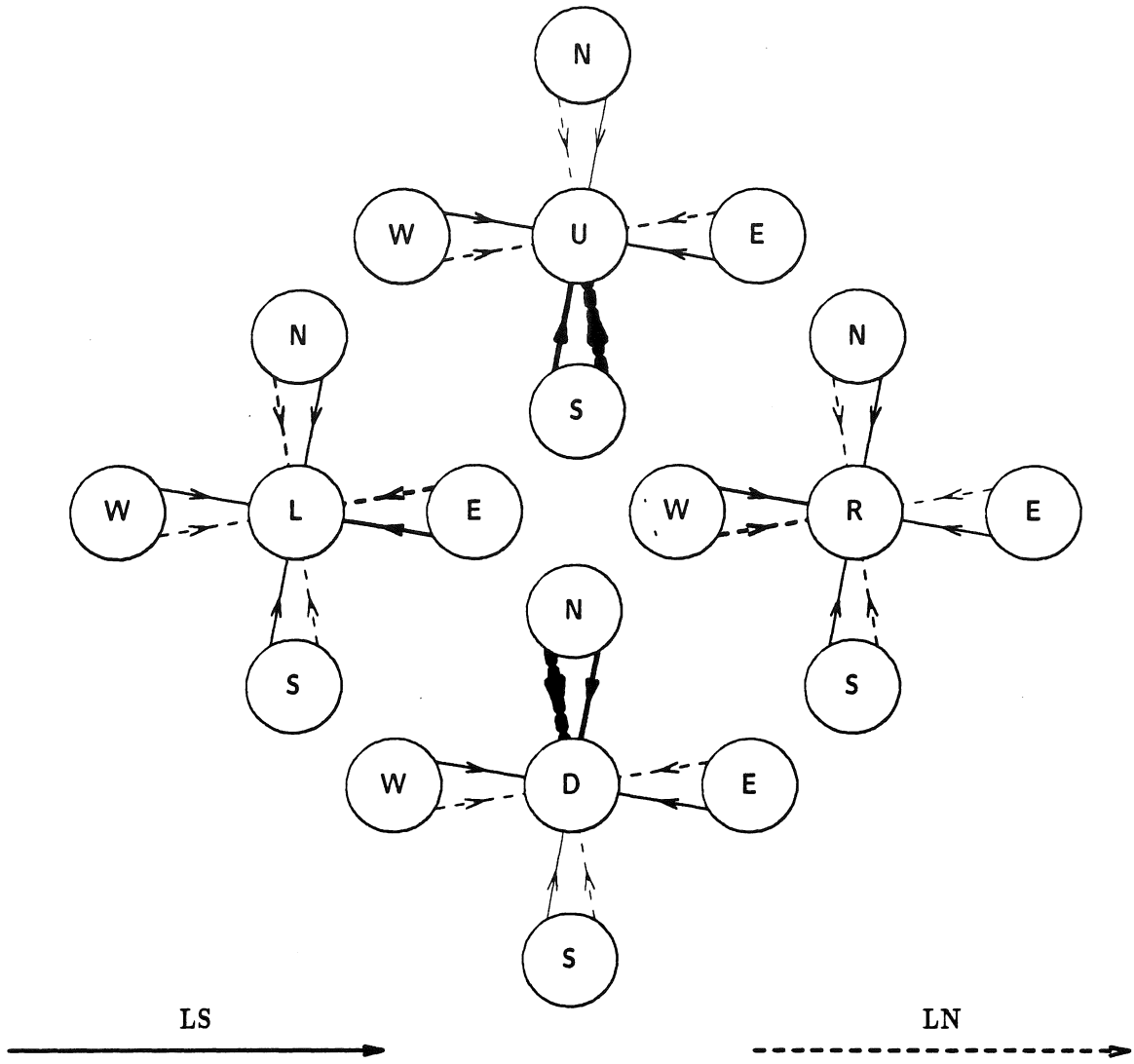

Figure 3.17: Concept descriptions after 500 moves.

of the dashed line (LN weight) between the element for the South landmark and the upward operator description indicates that it is necessary for the South landmark to be present to move up. This relationship also holds between the downward operator description and its element for the North landmark; the thick dashed line indicates that it is necessary to see the North landmark before moving down. The solid lines (LS weights) for these two elements are also significant, though less so, and they reflect the measure of sufficiency for their respective landmarks. The same opposing relationship holds for the leftward and rightward operators, though not as strongly.

Table 3.10 also indicates that the weights have been highly modified (compare to Table 3.9 after 2 moves). Note that the counts are rather large, reflecting an accumulation of experience. Furthermore, the differences between the weights for elements are slight compared to those

Table 3.10: Descriptions of operator preconditions after 500 moves.

Landmarks	Counts C_P	I_N	I_P	C_N	Weights LS	LN
Move Up						
north	81.25	92.15	53.03	41.09	**0.87**	**1.28**
south	91.60	79.38	42.68	53.87	**1.15**	**0.79**
east	85.48	84.61	48.81	48.63	1.00	1.00
west	85.98	84.38	48.30	48.86	1.01	0.98

Move Left						Move Right					
76.64	76.25	39.97	41.37	1.01	0.97	76.25	76.64	41.37	39.97	0.99	1.03
73.40	75.09	43.22	42.53	0.99	1.02	75.09	73.40	42.53	43.22	1.01	0.98
76.07	72.64	40.54	44.98	1.06	0.91	72.64	76.07	44.98	40.54	0.95	1.10
72.10	76.61	44.51	41.01	0.95	1.09	76.61	72.10	41.01	44.51	1.05	0.91

Move Down						
north	92.15	81.25	41.09	53.03	**1.14**	**0.78**
south	79.38	91.60	53.87	42.68	**0.87**	**1.27**
east	84.61	85.48	48.63	48.81	1.00	1.00
west	84.38	85.98	48.86	48.30	0.99	1.02

accrued in the examples of the previous section. The North landmark element is only slightly stronger than its counter-part the South landmark for either the upward or downward operator preconditions. This is in part due to the representation of attribute values as intensities and in part to the fact that there are no perfect predictors of when to apply a particular operator. There are only better and worse situations.

The corresponding set of operator expectations following these additional steps is depicted in Figure 3.18. Unlike Figures 3.13 and 3.15, this figure illustrates effective learning about each of the four operators, allowing STAGGER to move the organism toward the central goal location from any initial starting location.

Intuitive observations about the quality of the concept description listed in Table 3.10 and depicted in Figure 3.17 are easily demonstrated empirically by removing the external signal about the goal's location (i.e., removing any feedback), and noting the path the organism takes from a number of starting locations. Figure 3.19 depicts five such feedback-free walks, illustrating that STAGGER has formed a set of concept descriptions that accurately reflect the property of the operators and the location of the goal.

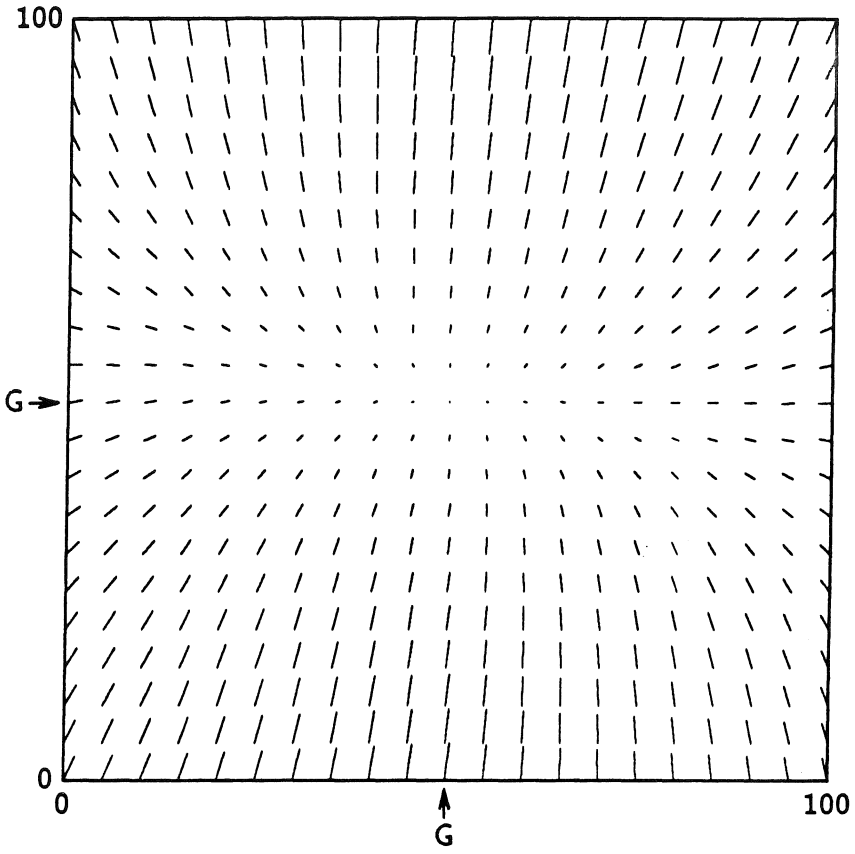

Figure 3.18: Operator expectation after 500 moves.

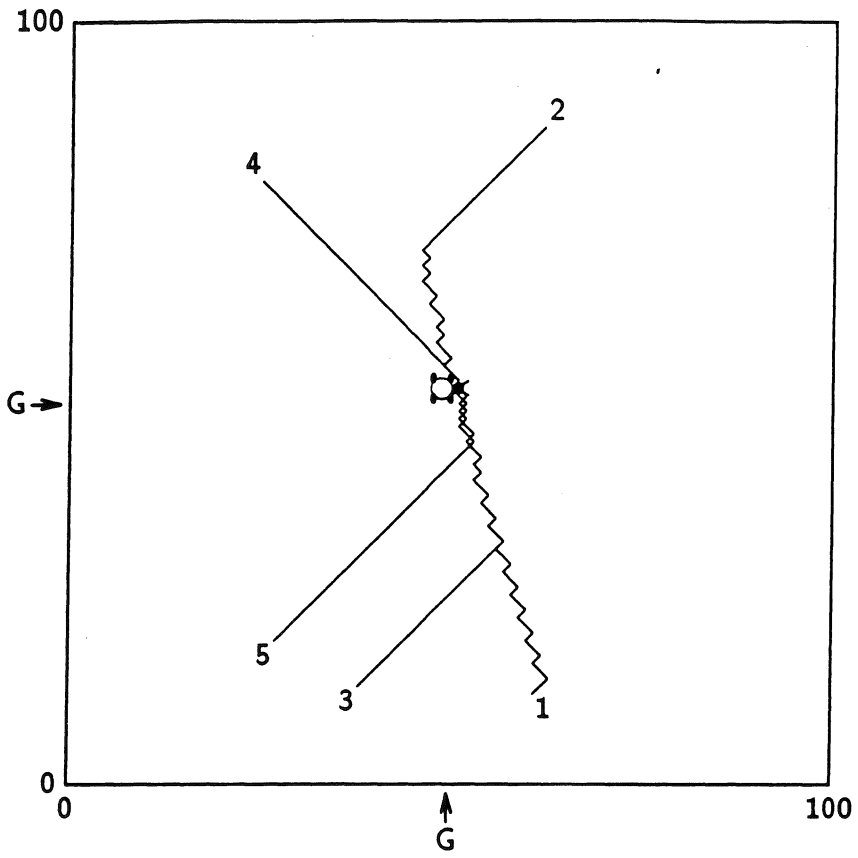

Figure 3.19: Finding the goal without feedback after 500 moves.

So far, this extended example has focused on a centrally located goal. It may be that this method is somehow utilizing the symmetry between the landmarks and the goal; perhaps it naturally balances the two pairs of opposing attribute-values. One way to test this assumption is to train STAGGER with an asymmetrical goal location. First, the goal was moved so that only one of the dimensions remained symmetrical; in this case the goal was moved upward so that it was relatively close to the North landmark. Figure 3.20 indicates that STAGGER is equally adept at acquiring asymmetrical precondition concepts that allow finding a northerly goal location.

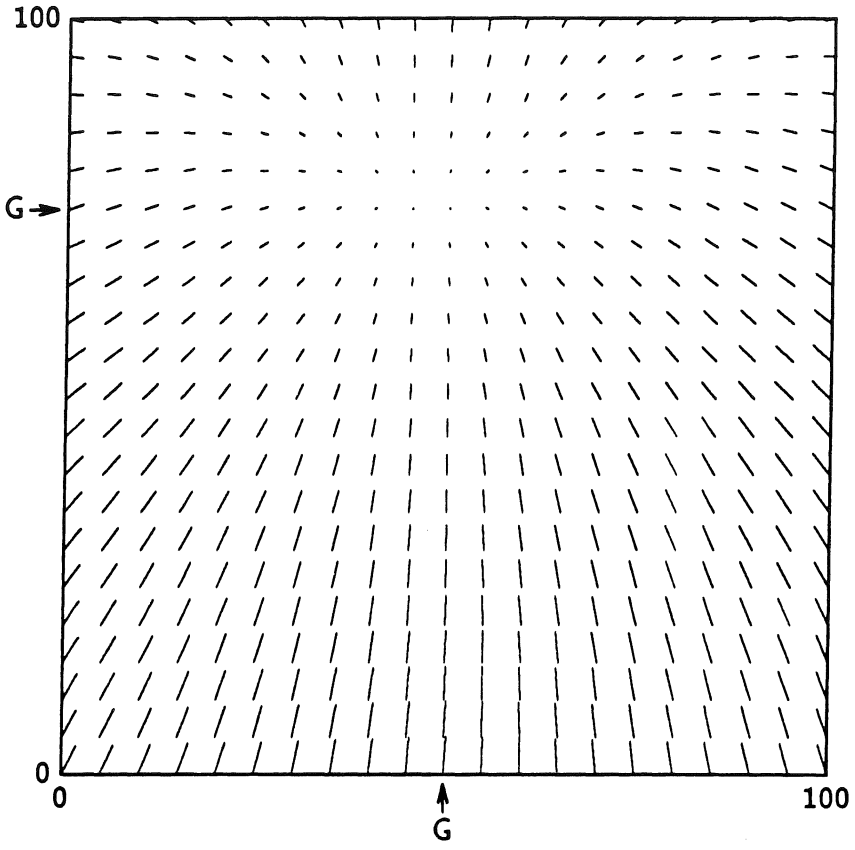

Figure 3.20: Operator expectation after 500 moves given a North goal.

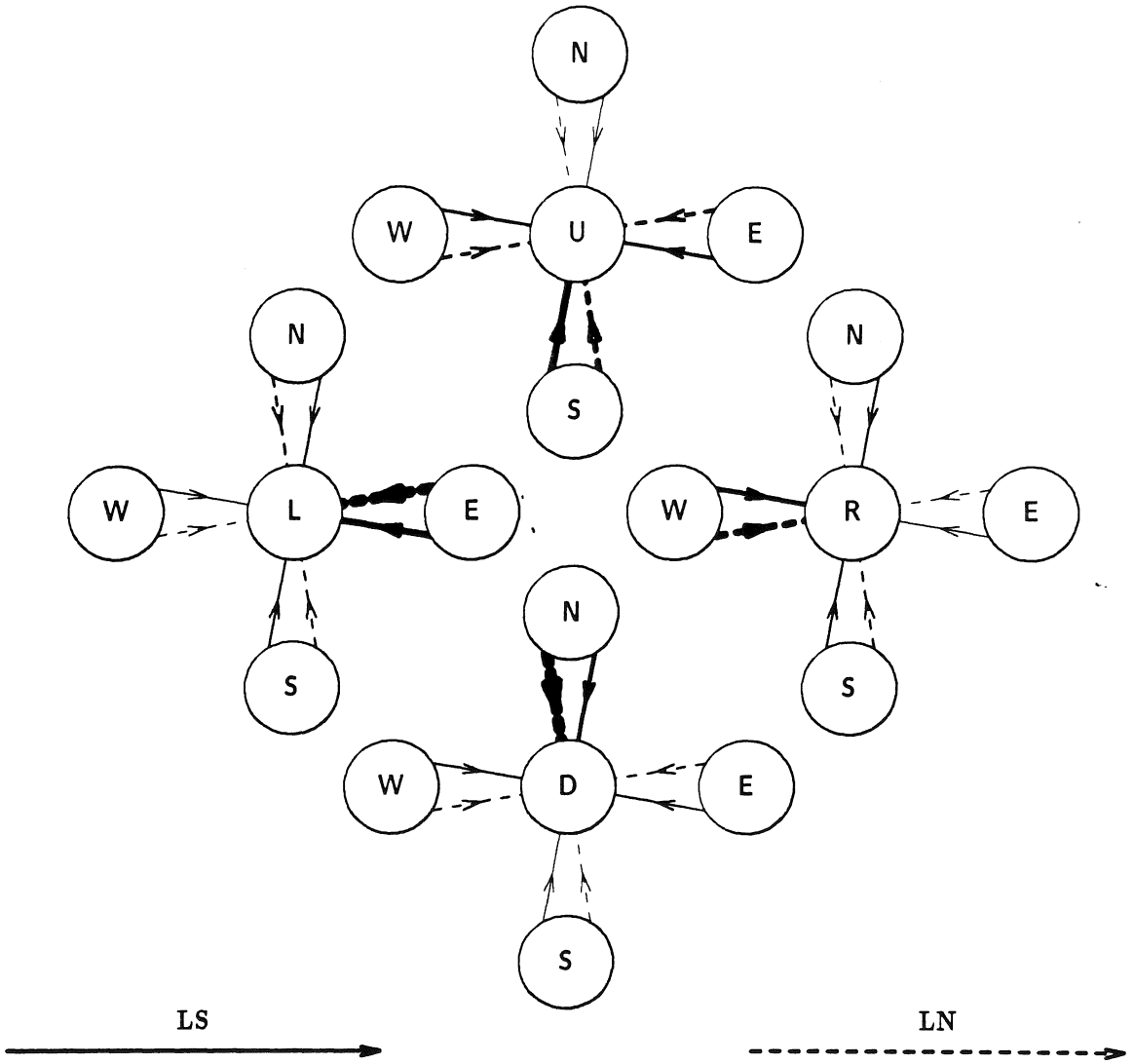

Figure 3.21: Concept descriptions after training for North goal.

Examining the individual concept descriptions reveals that the North landmark has become highly necessary for any application of the downward operator (Figure 3.21). In other words, compared to the central goal concepts of Figure 3.17, the organism will have to be closer to the North landmark before the downward operator is applied. There is some spurious asymmetry between the strengths of the East landmark for the leftward operator and the West landmark for the rightward operator. This effect disappears with further training.

This demonstration does not fully disprove the hypothesis that effective performance is due to the symmetry of the goal location, for the goal is still centered horizontally. Furthermore, it may be that the operator concept descriptions are able to home in on a particular landmark, and thus the Northerly goal location is a poor test. To test this the goal was moved to a location off-center in both the vertical and horizontal dimensions. Figure 3.22 depicts the net

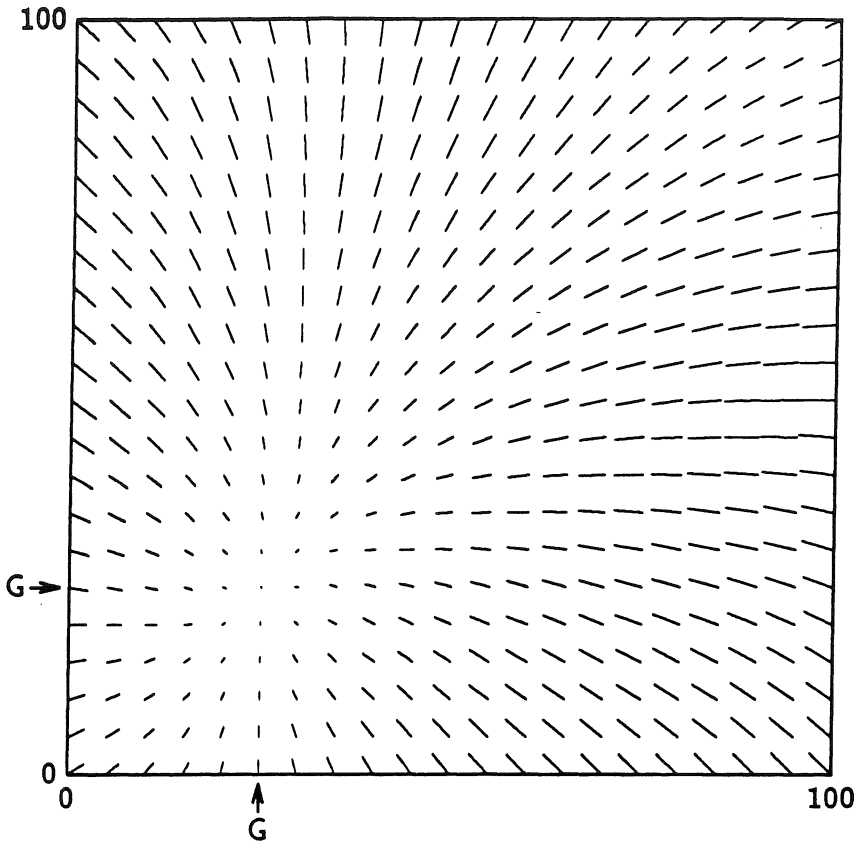

Figure 3.22: Operator expectation after 500 moves given a South-West goal.

operator expectation for a South-Westerly goal location after 500 moves.

The corresponding concept preconditions are depicted in Figure 3.23. Compared to Figure 3.17 (which shows the concepts for a central goal) note that the West landmark is also predictive of opportunities for the downward operator, and the South landmark predicts an occasion for applying the leftward operator. With these additional weights, the organism moves down as it nears the West and leftward as it moves South. These intuitions are borne out by Figure 3.22 above.

Thus, the acquisition of effective navigating behavior in STAGGER is not some fortuitous result arising from symmetry or landmark finding. Rather, the Bayesian weight method allows the formation of operator precondition descriptions that describe arbitrary goal locations in this simple two-dimensional world.

To examine the learning rate of STAGGER, we can plot the number of moves with feedback against the closeness of each attempt to find the goal without feedback. For this set of assessments, the organism was first allowed to take 100 moves. Then it was placed at 10

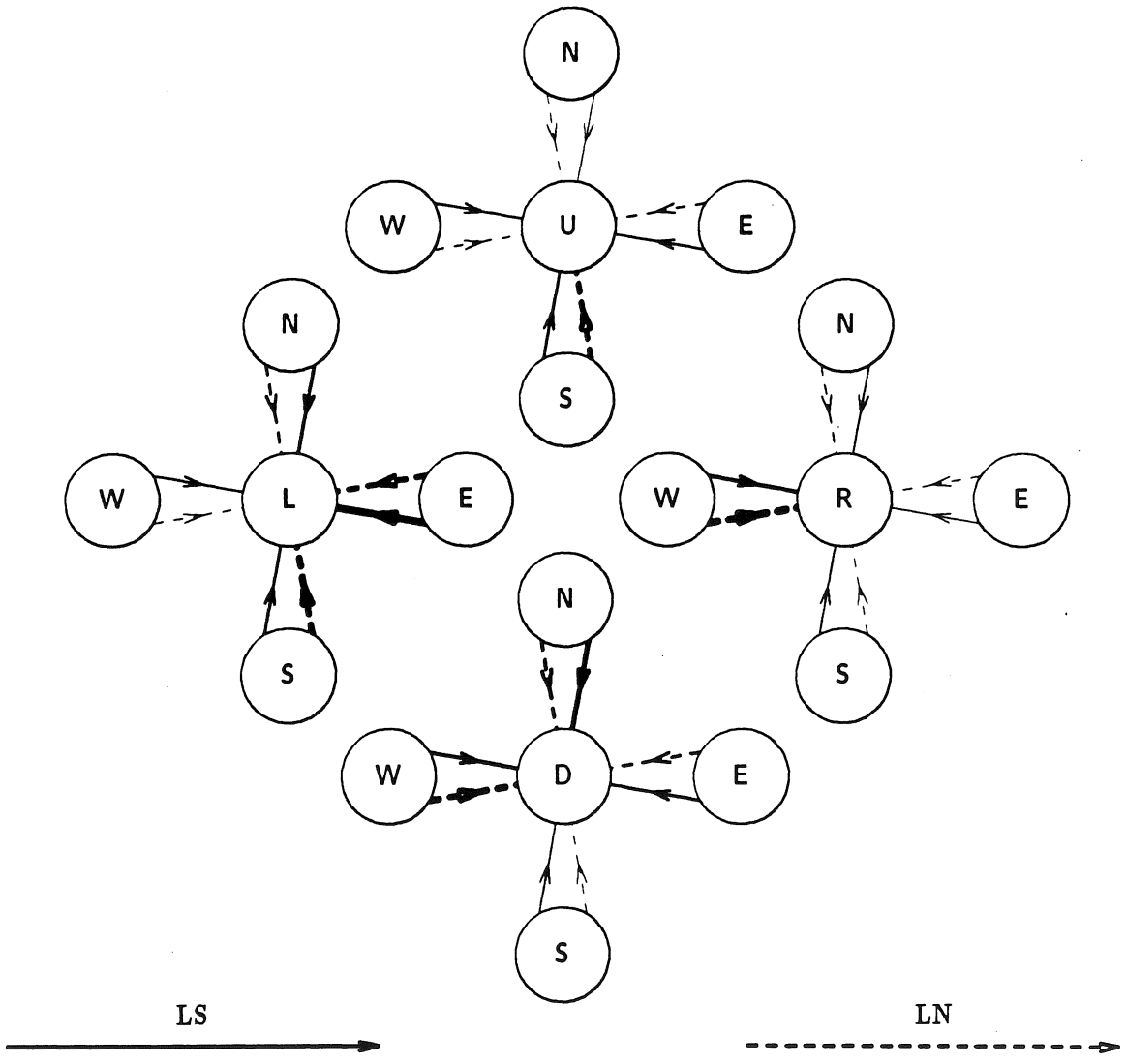

Figure 3.23: Concept descriptions after training for South-West goal.

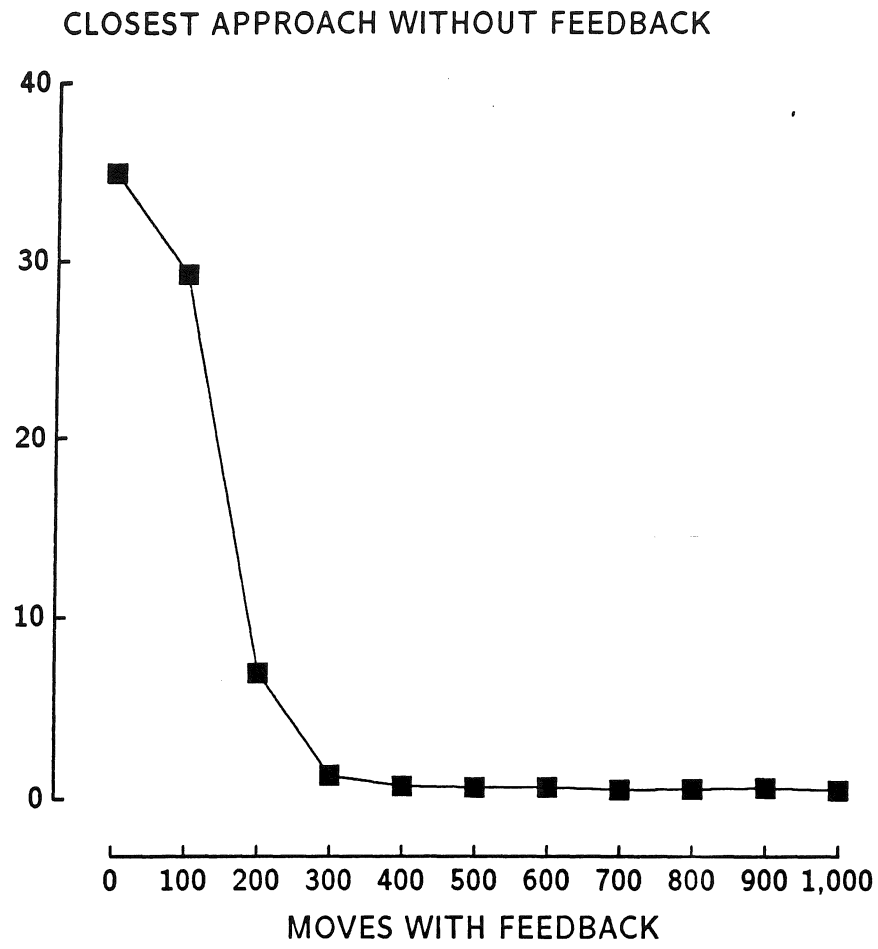

Figure 3.24: The learning rate of STAGGER in a simple spatial learning task.

random locations (one at a time) and allowed to make 100 moves without feedback. By averaging the closest the organism came to the goal after each attempt, we have a quantitative indication of learning quality to this point. Figure 3.24 depicts an average learning rate over 20 series of moves. Analyzing the individual data reveals that in each case, after 300 moves all attempts to find the goal location without feedback came within 3 units of the goal location.

In summary, through a few, simple modifications STAGGER is able to successfully direct an organism through a two-dimensional world toward a goal. The preconditions for each of the four directional operators are represented as concept descriptions composed of elements for each of the four landmarks. Sensory input from the four landmarks is matched against these concept descriptions, yielding an expected appropriateness for each operator. By applying an operator and noting the change in the goal's sensory value, STAGGER reinforces appropriate operator applications and weakens inappropriate ones. After a series of these moves, the operator preconditions guide the organism toward the goal from novel locations even in the absence of the goal landmark.

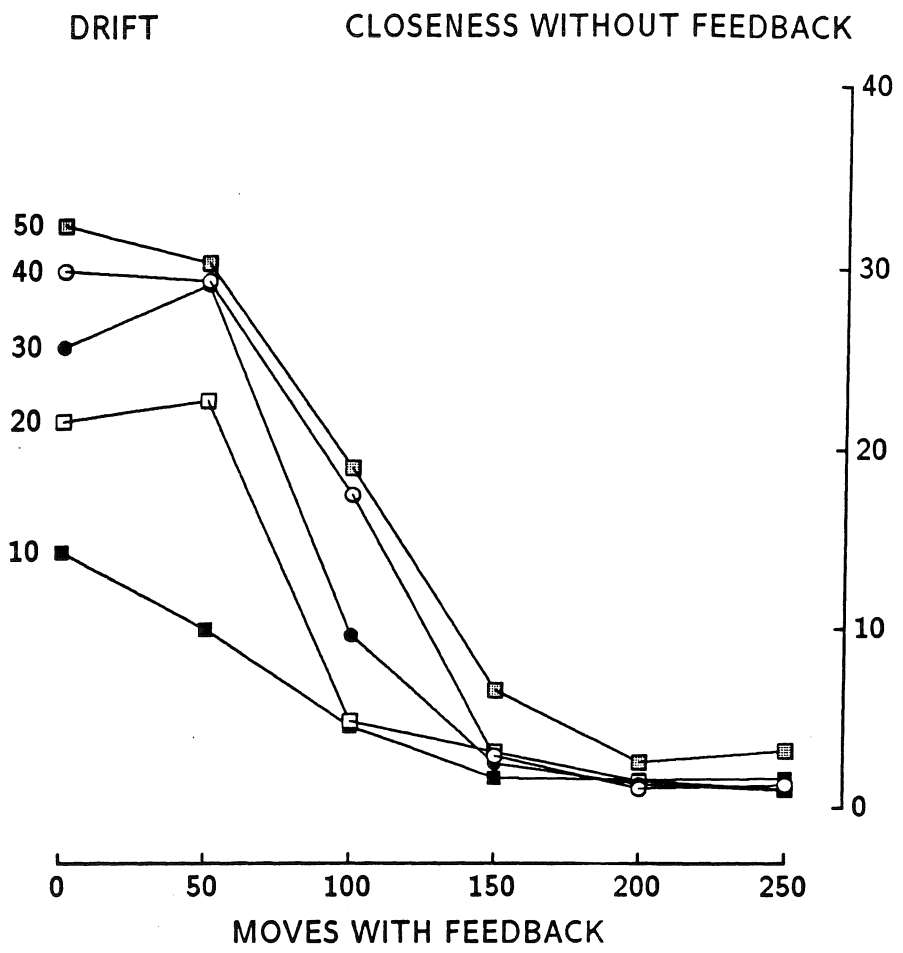

Figure 3.25: Moves to relearn new goal location as a function of goal drift.

This task domain affords a natural introduction to an additional property of the learning mechanisms employed in STAGGER: the ability to overcome previous learning if the environment changes and the concept drifts. In the next subsection I present two examples of STAGGER's tolerance to this type of instability.

3.3.2 STAGGER's ability to unlearn and relearn

In many learning situations the concepts to be characterized are not stable. For instance, the fox learns to adapt to the color of his prey as summer brings color out in the coat of the rabbit and winter removes it. STAGGER tolerates these types of changes with relative ease, for the numerical underpinnings of concept descriptions are easily changed to reflect current situations. In this subsection I present a pair of simple examples of concept drift and show how STAGGER tolerates them.

STAGGER is able to build operator preconditions that let it find the goal location from any starting location in a simple two-dimensional world. Consider the effect that moving this location will have on the ability of STAGGER to guide the organism. Since the operator preconditions are represented as a set of weighted landmarks, moving the goal location requires an adjustment of the LS and LN weights. Opportunities to adjust the weights arise whenever the new goal location requires a different operator application, i.e., whenever the organism is between the previous goal location and the new one. Intuitively, the further the goal location drifts from its original spot, the greater the change that will be required of the landmark element weights. If the goal has not drifted very far, then the landmark weights for each operator precondition description will not require much modification.

This intuition is borne out in practice; Figure 3.25 depicts a three-way comparison of the number of moves to find the goal without feedback as a function of the number of training moves with feedback. The additional independent variable, the amount of goal drift, is plotted by individual lines. Each line represents a movement of the goal location along the diagonal, away from the South-West location depicted in Figure 3.22. After 50, 100, 150, 200, and 250 moves, the organism was placed at 10 random locations and allowed to move toward the goal without any sense of the goal landmark. The closest approach of each of the 10 feedback-free walks was averaged together, and the process was repeated 10 times for each of the 5 drift amounts.

Note that in Figure 3.25, the greater the drift, the greater the number of moves required to readjust the landmark weights within the operator precondition descriptions. When the goal is moved only 10 units away from its former location, the operator concepts are effectively revised after 100 moves. When the goal is moved by 50 units, to the North-East corner, it is not until after 200 moves that the operator precondition descriptions accurately reflect the new location.

For completeness, consider a second, more drastic type of environmental change presented by Barto and Sutton (1981). Instead of moving the goal, in this experiment we move the landmarks. For instance, interchanging the East and West landmarks of Figure 3.18 makes the rightward and leftward operator preconditions incorrect. Figure 3.26 depicts the resulting operator expectations. As might be expected, the lines indicate that STAGGER will still direct the organism toward the vertical center but will sweep the organism away along one of the horizontal directions.

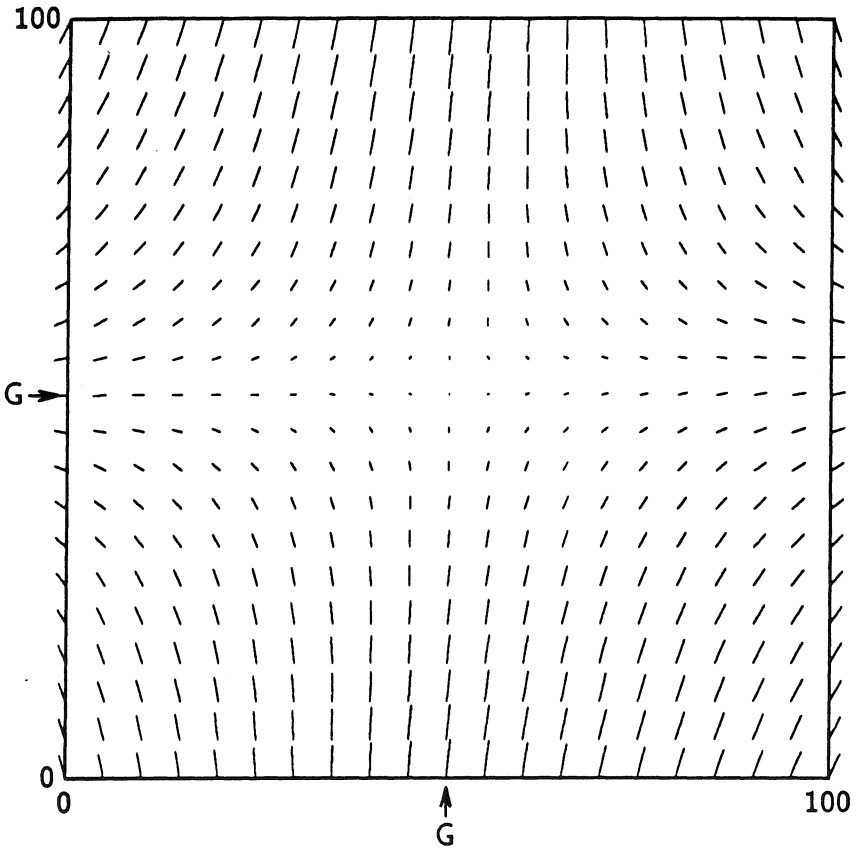

Figure 3.26: Operator expectation with East and West landmarks reversed.

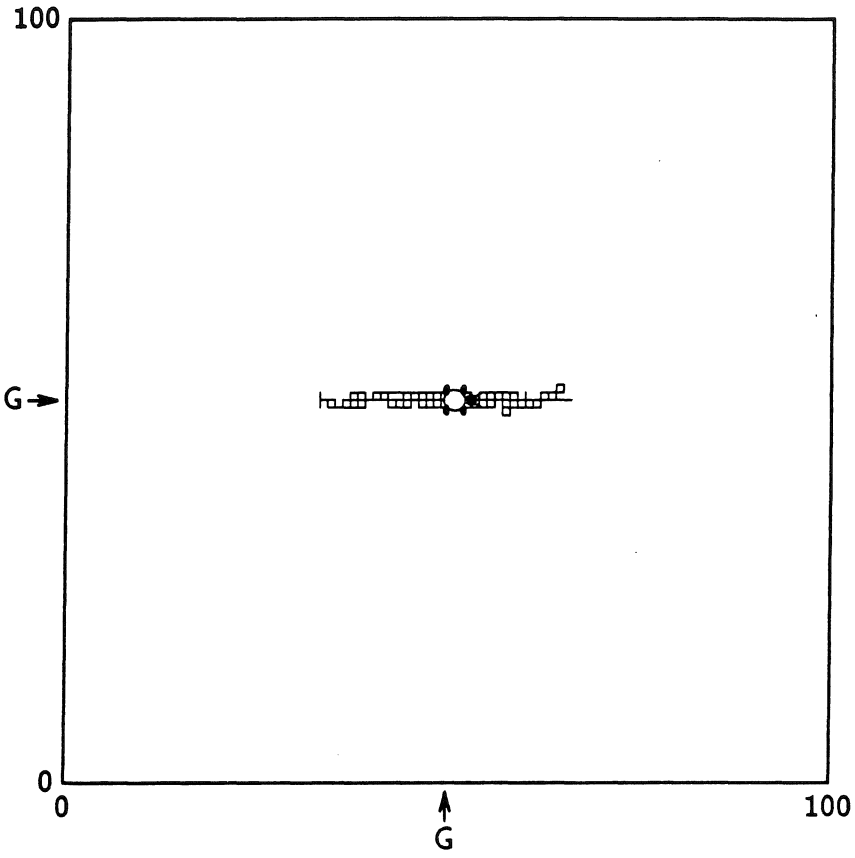

Figure 3.27: Path after 1000 moves with East and West landmarks reversed.

Given an additional 500 moves, STAGGER is able to correct for this change. It reverses the precondition definitions for the left and rightward operators by modifying the landmark weights within the rightward and leftward descriptions. Figure 3.27 depicts the 500 moves after the East and West landmarks have been interchanged. Note that STAGGER directs the organism out toward one of the wayward landmarks and then toward the other. This exploration is basically confined to the horizontal dimension, for the dynamics of the vertical dimension have remained unchanged.

Following this recovery period, STAGGER has reacquired an effective precondition for each of the operators. The resulting graphical depiction of operator expectation following this interchange of the West and East landmarks and subsequent retraining is similar to Figure 3.18. Also, Figure 3.28 reveals that the leftward and rightward operators have reversed their predictive relationships. As compared with Figure 3.17 (central goal location), the "West" landmark is now more predictive for the leftward operator, and the "East" wins over the "West" for the rightward operator.

In general, the representation of concepts as a set of weighted elements is a malleable format; the essence of the description is the magnitude of the weights, and they may be easily changed. Basing the LS and LN values on accumulated experience inhibits change, for if the environment has remained stable over a long period of time, the underlying counts will have grown large and will be resistant to alteration. This is a useful property, for it lends robustness to the learning method by allowing it to withstand small variations in the environment's regularities. If the method were extremely sensitive, it might never converge. Instead, the accrued counts underlying the weights slowly dampen variation in concept description, stabilizing them even though the data may be somewhat noisy. Moreover, as I explain in Section 7.3, this behavior is also exhibited by experimental subjects. Basically, a relatively naive learner is willing to change expectation while an experienced subject is reluctant.

3.3.3 Observations on a simple task

In this section I presented an application of STAGGER to the task of forming operator preconditions in a simple two-dimensional world. The task is not particularly demanding (e.g., it does not require a credit-assignment method), but it serves as a demonstration of the flexibility inherent in STAGGER's mechanisms. Besides being able to form concept descriptions for simple symbolic objects, in this section I have shown that STAGGER is also capable of capturing numerical associations. Additionally, the propensity of concepts to change over time is easily accommodated using these methods. For STAGGER, slight concept drift requires less retraining than drastic drift.

In the following section I describe another task for which STAGGER is reasonably well suited. Instead of dealing with object descriptions or olfactory landmarks, this task focuses

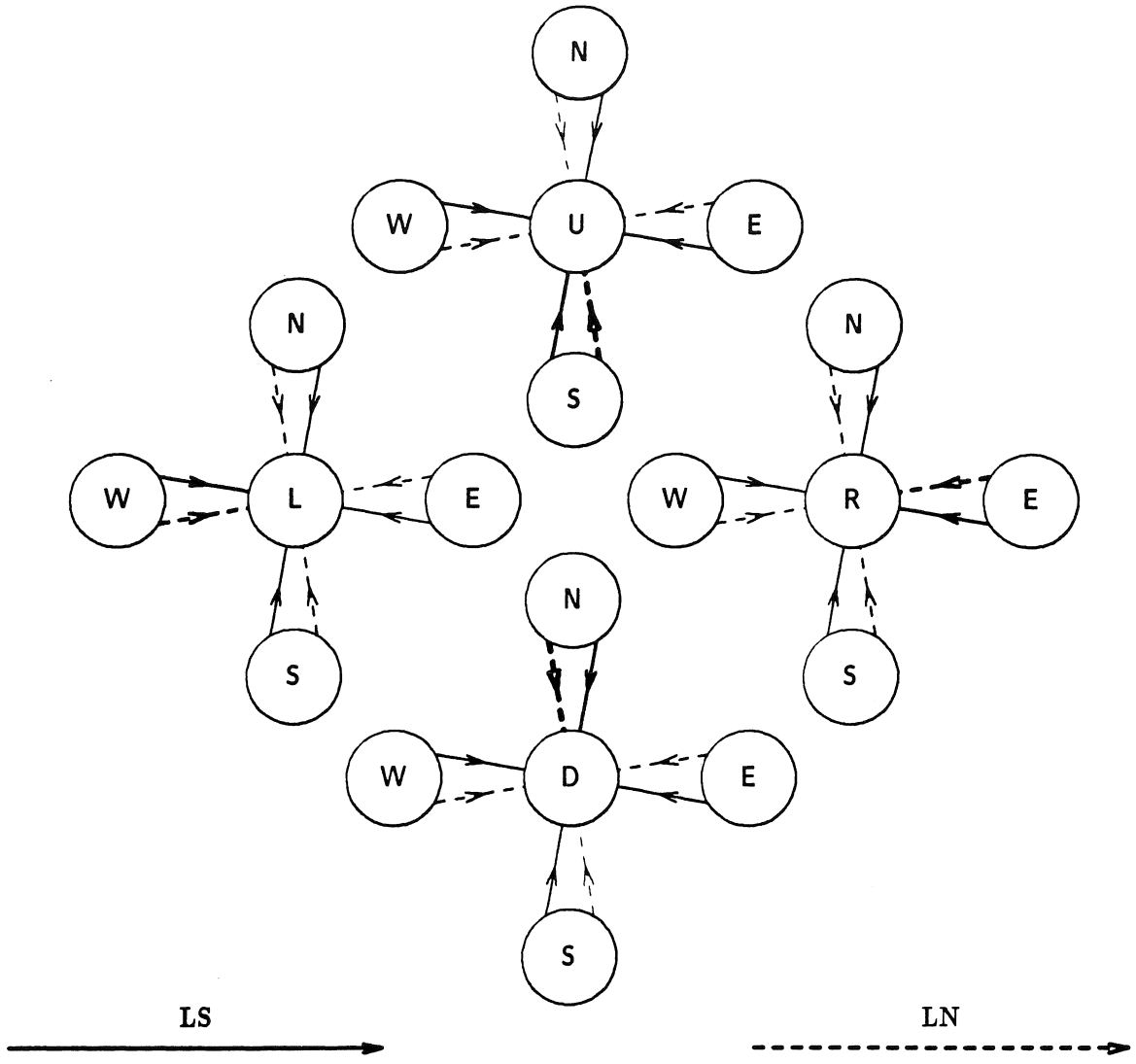

Figure 3.28: Concept descriptions with East and West landmarks reversed.

on the retrieval of relevant documents given a list of keywords. STAGGER can serve to automate portions of this task that have traditionally been performed manually.

3.4 Adaptively retrieving information

In the previous sections I described a pair of simple adaptation tasks and showed how STAGGER addressed each. In this section I turn to a discussion of another task: retrieving relevant documents given a set of keywords. Unlike the earlier two examples, this formulation relies on a large number of concepts – more than the number of featural descriptors for any one concept. In brief, the Bayesian weight method is able to effectively retrieve relevant documents and recover from poor initial document descriptions. The section begins with a definition of the task, continues with a detailed description of the formulation and operation of STAGGER within it, and concludes with some quantitative indications of the applicability of this method.

3.4.1 Information retrieval

As the amount of available information grows, so do the problems associated with its effective retrieval. To cope with these difficulties, various systems have been developed to provide easy and efficient access to information libraries. These systems may be roughly divided into two classes based on the types of information they provide. Data retrieval systems are typically used to find the answer to a specific question, such as 'What is the melting point of Aluminum?' In contrast, the task of information retrieval usually aims at finding a number of documents or sources that are relevant to a particular topic, such as 'welding methods for Aluminum' (Lancaster, 1979). The effectiveness of systems in the former class is easily assessed by measuring how quickly the correct answer is found. However, in the latter class the notion of an effective retrieval is somewhat fuzzy since it involves a tradeoff between recalling many of the relevant documents and few of the irrelevant ones. In this section, I focus on this latter task.

Figure 3.29 outlines the major components of an information retrieval system as identified by Lancaster (1979). Starting at the top of the figure, documents to be stored are first subjected to a conceptual analysis. Usually this is done by an individual with expertise in the subject area. A conceptual representation of the document is translated into a set of retrieval keys that are drawn from the system's vocabulary. This vocabulary is constrained in order to control the potential ill-effects of synonyms, antonyms, and homographs (same spelling but different meaning). The document and its assigned keywords are then indexed into some internal representation.

Moving to the bottom of Figure 3.29, users form queries in normal English or a specialized

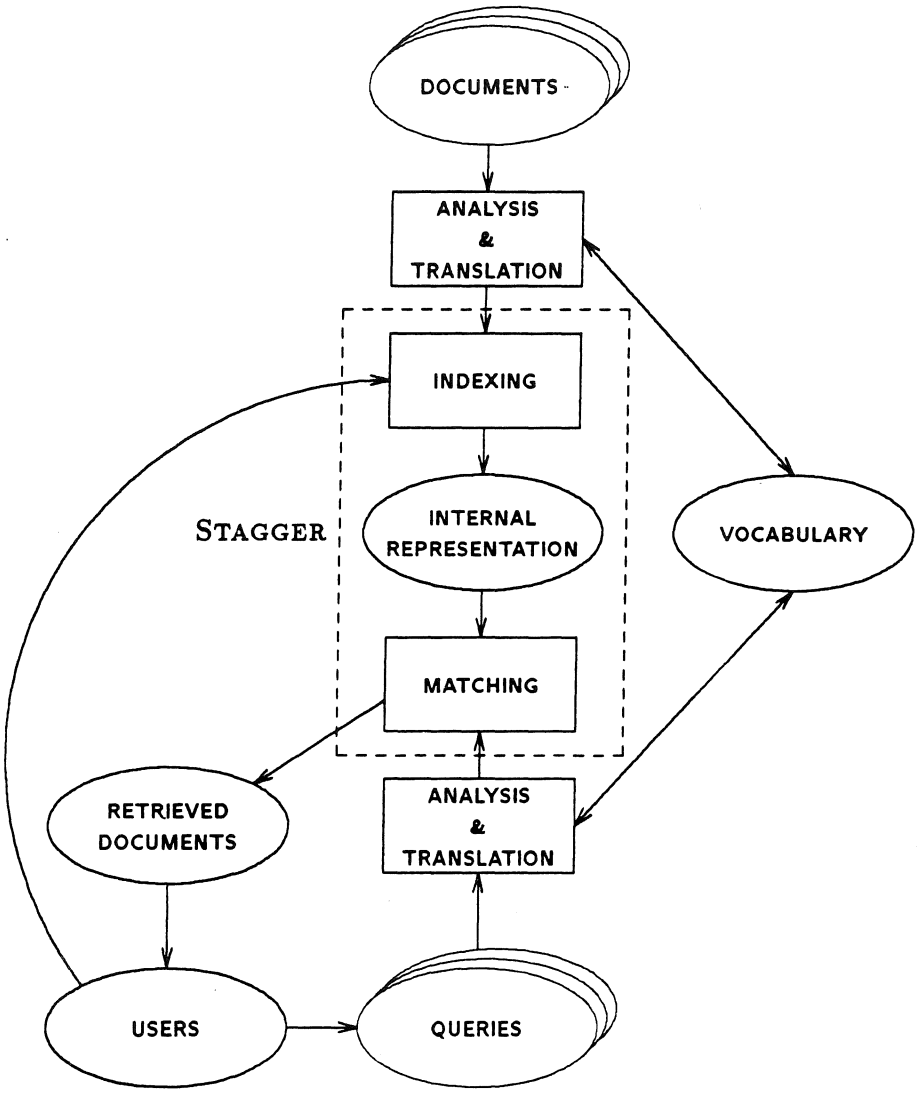

Figure 3.29: The functions performed in many information processing centers.

form of keyword selection. In any case, a type of conceptual analysis and translation is performed on each query. The resulting query may be a complicated structure or a simple list of keywords.

To merge the queries and documents, some type of matching process searches through the internal representation of the documents, seeking to satisfy the user's request. The retrieved set of potentially relevant documents is returned to the user. A form of feedback is possible if the user is able to identify either relevant documents that were not found, or irrelevant documents that were retrieved. The cycle of request, search, and retrieve may be iterated until the user's information need is satisfied.

The dashed box outlines the functionality implemented for this example with STAGGER. Referring to the top of Figure 3.29, I assume that the documents are already analyzed for content and have been assigned a set of reasonably descriptive keywords. Similarly, referring to the bottom of the figure, I assume that the queries are translated into a simple list of keywords within the system's vocabulary. The heart of the demonstration of this section lies in the processes of indexing documents and matching queries against them. The former process is implemented in STAGGER as a process of modifying retrieval weights for each document's individual keywords and adding new keywords if necessary; user evaluation is an essential element for indexing improvement. The second process of matching follows the framework established in the two earlier examples: query keywords are matched against document keywords to derive an expectation of the relevance of each document.

A set of 77 documents for this example were collected from learning papers published in the proceedings of the national conferences on artificial intelligence. Prior to beginning programming for this section, I assigned between 3 and 13 keywords to each document, drawing the keywords from the title, abstract, section headings, and the name of system described. The authors' names were not included because I wanted to be able to informally evaluate the query generation method described below. The total vocabulary numbers 281 and includes some terms that occur in 36% of the documents and some that only occur in one document.[2]

As an alternative to the expensive process of collecting actual queries, a simple procedure generates queries randomly by imitating a user with a subject-oriented request. First two or three documents are selected randomly until a group is found which shares one or more keywords. Then each of the documents that contain this cluster of keywords are collected together to form the relevant documents for the query. Finally, the keywords for the query are generated by selecting one or two keywords at random from each of the relevant documents. The net effect of generating queries in this manner has been quite satisfactory. For instance, although the authors' names are not part of the keywords describing documents, frequently a query will contain all of the papers by a given author. Table 3.11 lists an example query that

[2]These documents and their assigned keywords are listed in Appendix C.

Table 3.11: Sample query.

RELEVANT DOCUMENTS	QUERY KEYWORDS
(O'Rorke, 1984) (Doyle, 1986) (Hall, 1986) (Lebowitz, 1986) (Mooney & Bennett, 1986) (Rosenbloom & Laird, 1986)	causal chunking equalities explanation explanation-structure generalization knowledge-based learning-from-examples precedent schema similarity-based

was generated around the subject keywords **explanation-based** and **generalization.**

The information retrieval task, therefore, involves translating documents and queries into a representation suitable for matching. In the next section, I present a formulation of this task in terms of learning from examples and show how STAGGER improves its performance on the indexing and matching subtasks.

3.4.2 STAGGER's role as an automatic indexer and matcher

Briefly stated, the goal of information retrieval is to locate relevant documents given a user's query. By viewing each document as a concept to be described, and each query as an example description of the relevant documents, this task may be mapped onto learning a concept from examples. The task is to improve information retrieval as successive queries are processed. There are many concepts (a relevancy description for each document), and the examples are drawn from queries. Feedback essential to the learning task is drawn from user evaluation of the relevance of each of the retrieved documents – an indication of whether or not a query is an example description of a document. With this mapping in mind, let us consider STAGGER's approach to the indexing and matching processes of information retrieval.

STAGGER's internal representation for documents and their keywords is similar to Mooers' original formulation.[3] In Mooers' system, a document is represented by a punched index card with coded holes around its edge. Each of the holes corresponds to a keyword or class, and by cutting out specific holes, it is possible to encode the appropriate retrieval keys for a particular document. Figure 3.30 illustrates a typical Mooers punched-hole card.

[3]See (Lancaster, 1979).

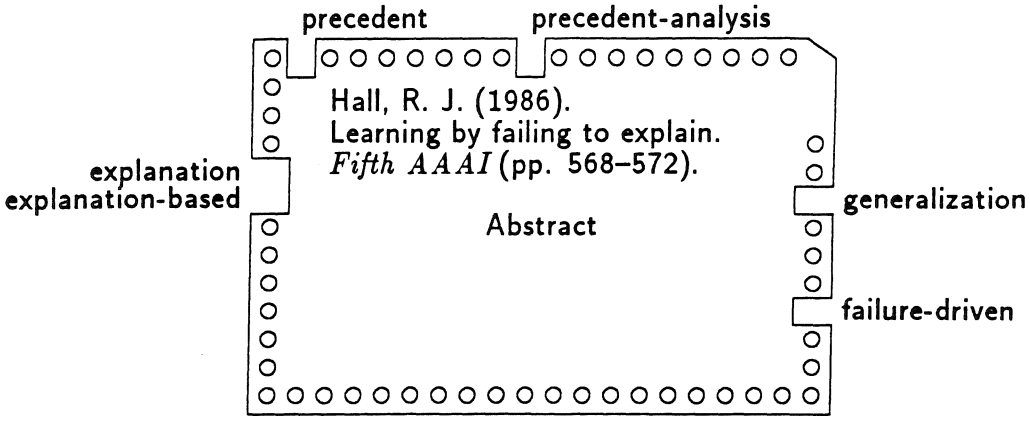

Figure 3.30: Sample Mooers type punched hole index card.

Retrieving relevant documents with Mooers' system is simple. To find documents corresponding to a particular keyword, a rod is inserted through the appropriate hole in the deck of document cards. By lifting the cards with the rod, those cards with a notched hole in the key's location fall out. These documents can then be restacked and the process repeated if additional selection is required.

STAGGER forms a concept to describe the relevancy of each document. The cut holes in the card depicted in Figure 3.30 are added as elements to the document's relevancy description. A keyword query corresponding to a rod placed through the deck is a simple check for the presence of a keyword element. Unlike Mooers' system, in STAGGER the keyword holes are continuously sized, and each keyword element is dually weighted to represent a degree of relevancy. As queries are processed and documents are identified as relevant or irrelevant by the user, STAGGER adjusts these LS and LN weights, thereby improving the quality of future retrievals.

To correct for the possibility of initially poor indexing, if a relevant document fails to be retrieved on a given query (an error of omission), new elements are added to that document's concept description for each of the query keywords. Adding these new predictors would result in over-retrieval if they were not subject to subsequent refinement. However, their weights are adjusted in the same way as the other elements, and if they are not predictive of this document's relevancy, then they become weakly weighted. Furthermore, poor keywords are removed from the document's concept description if, after some experience, they prove to be irrelevant. If the sum of the counts for an element is more than twice the initial amount, and the LS value is not greater than one (or the LN value is not smaller than one), then the keyword element is removed from the document's relevancy description.

Having described the general formulation of the information retrieval task, in the succeeding paragraphs I describe a detailed example following the indexing and matching of Hall's 1986

Table 3.12: Initial concept description for (Hall, 1986).

KEYWORDS	COUNTS				WEIGHTS	
	C_P	I_N	I_P	C_N	LS	LN
precedent	1	1	1	1	1	1
explanation-based	1	1	1	1	1	1

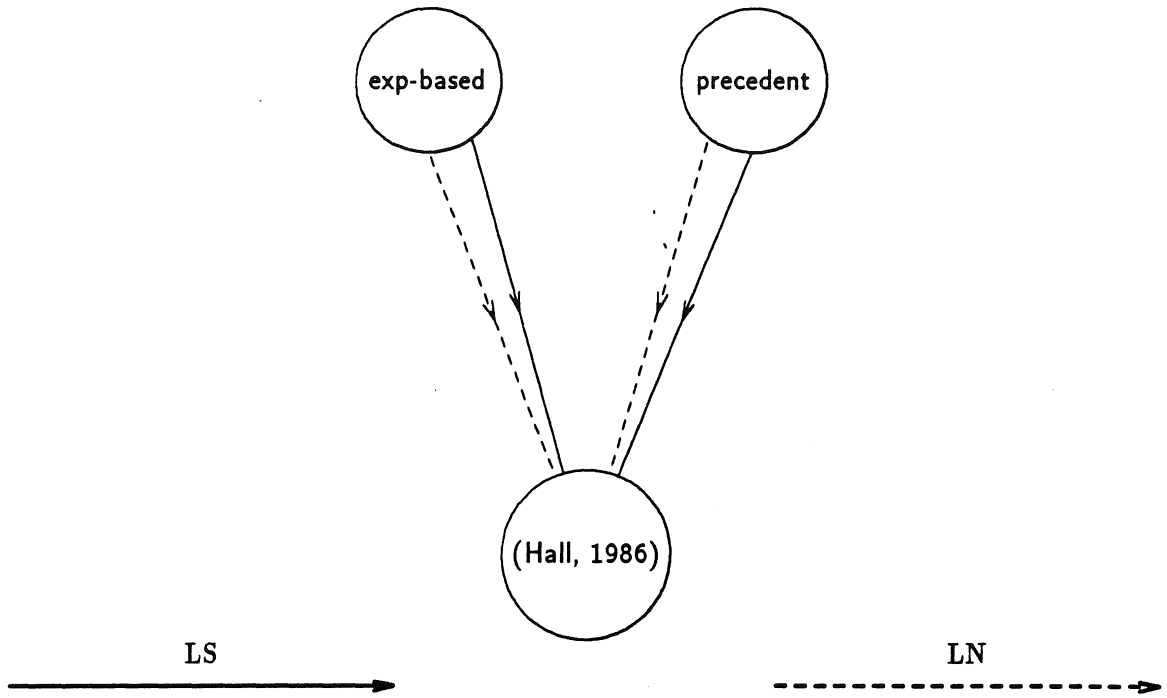

Figure 3.31: Initial keywords for (Hall, 1986).

AAAI paper depicted in Figure 3.30. To simulate a potentially incomplete keyword-assigning process, only one fourth of the keywords are included in the initial concept description. Though the resulting elements could be meaningfully weighted, they are assigned unbiased weights. Table 3.12 lists the initial concept description elements for (Hall, 1986).

Figure 3.31 displays this initial concept description using the notation developed in the previous two sections. Mirroring Table 3.12, each of the lines (weights) connecting the outer circles (elements) are of equal weight, and thus the concept description is naive.

Table 3.13 lists the first query which was generated by finding the set of four documents which share the keywords **explanation** and **generalization**. The set of query keywords was collected by choosing up to two keywords randomly from each of the relevant documents. As the table indicates, (Hall, 1986) is relevant and should be retrieved.

Matching the document in Figure 3.30 against this query is straightforward. Using the

Table 3.13: First query.

RELEVANT DOCUMENTS	QUERY KEYWORDS
(DeJong, 1982) (Doyle, 1986) (Hall, 1986) (Mooney & Bennett, 1986)	domain-theory explanation explanation-based explanation-structure generalization precedent-analysis

simple matching equation introduced in the previous chapter (Equation 2.3), the LS weights of all keywords present in the query are multiplied together with the LN weights of all missing keywords.

$$\begin{aligned} Odds(Relevant) &= (C_P + I_P)/(I_N + C_N) \\ &= (1+1)/(1+1) \\ &= 1 \end{aligned}$$

$$\begin{aligned} Odds(Relevant|Query) &= Odds(Relevant) \times \prod_{\forall Matched} \text{LS} \times \prod_{\forall Unmatched} \text{LN} \\ &= 1 \times 1 \times 1 \\ &= 1 \end{aligned}$$

The result of *odds* = 1 corresponds to the unbiased property of the initial concept description elements. In this case, assume that STAGGER makes the random choice that (Hall, 1986) is not relevant to this query. This is a recall failure, or an error of omission, and in order to correct possible underindexing of this document, a new element is added to the document's concept description for each of the query keywords. Next, each of the elements' counts are updated. This is a positive instance; thus the positive confirming (C_P) counts are incremented for keywords which were in the query, and the positive infirming (I_P) counts are incremented for those which were absent. Table 3.14 lists the concept description for (Hall, 1986) after decaying the counts and computing the LS and LN weights according to Equation 2.4.1.

Figure 3.32 displays the current keyword description of (Hall, 1986). Newly added elements corresponding to keywords have their weights updated, as do previously existing ones. Looking ahead a bit, not all of these keywords will survive; for instance, **generalization** will eventually prove ineffective and be dropped.

The next query is generated around the subject keyword **problem-solving**. This relatively general subject heading covers 17 documents, and 22 query keywords are collected from these. Table 3.15 lists this query and its relevant documents. However, in this case (Hall, 1986) is not relevant. It does provide some empirical support for the quality of the query generation

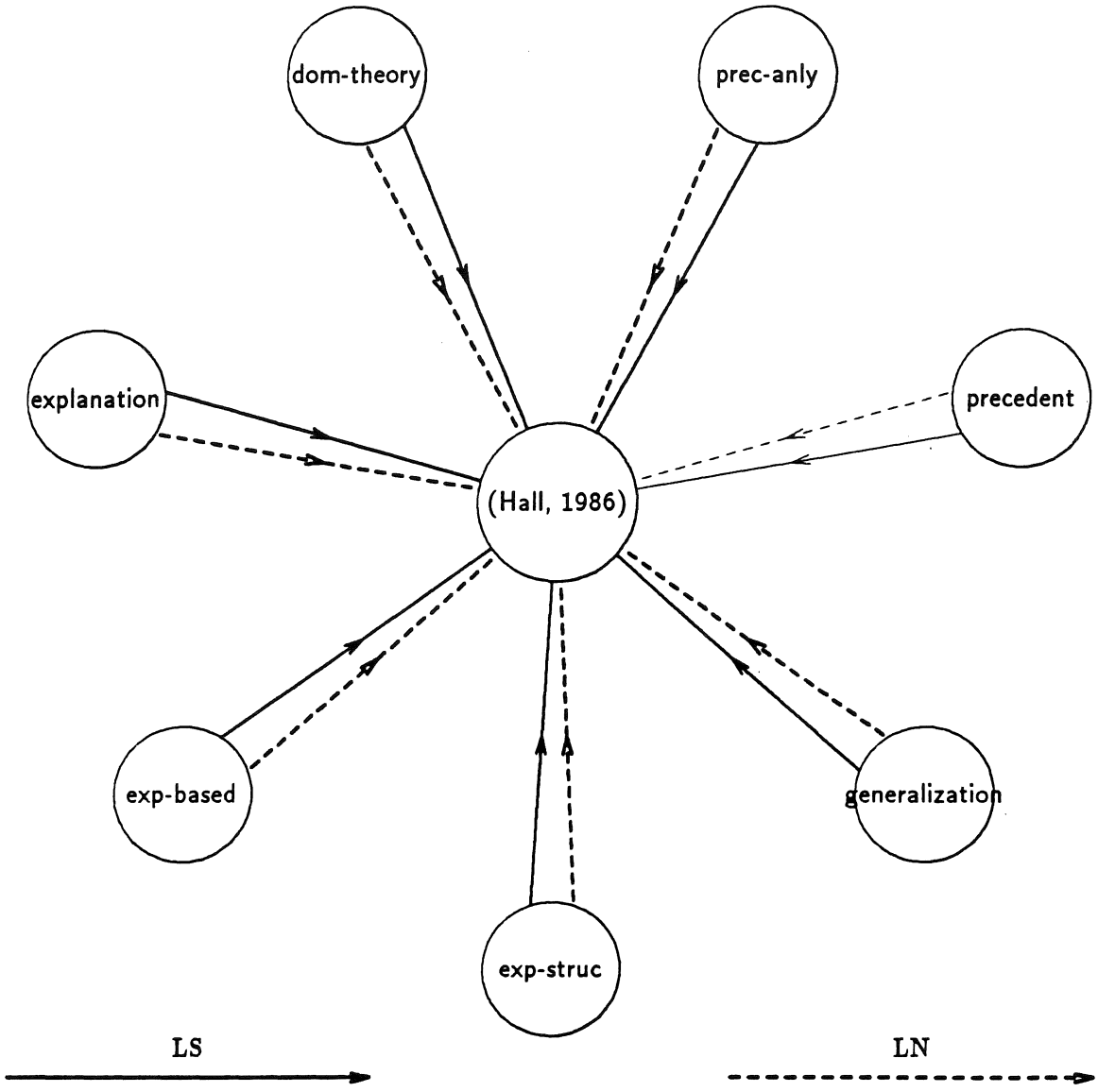

Figure 3.32: Keywords for (Hall, 1986) after processing one query.

Table 3.14: Augmented concept description for (Hall, 1986).

KEYWORDS	COUNTS				WEIGHTS	
	C_P	I_N	I_P	C_N	LS	LN
precedent	0.99	0.99	1.99	0.99	0.66	1.34
explanation-based	1.99	0.99	0.99	0.99	1.34	0.66
precedent-analysis	1.99	0.99	0.99	0.99	1.34	0.66
generalization	1.99	0.99	0.99	0.99	1.34	0.66
explanation-structure	1.99	0.99	0.99	0.99	1.34	0.66
explanation	1.99	0.99	0.99	0.99	1.34	0.66
domain-theory	1.99	0.99	0.99	0.99	1.34	0.66

technique. Both of the papers by Kibler and Porter, three papers with Korf as an author, and three papers by Carbonell are included, even though the authors' names were not included as part of the keyword-document data set.

Referring back to Equation 2.3, the match between the concept for (Hall, 1986) and this query yields:

$$\begin{aligned} Odds(Relevant) &= (C_P + I_P)/(I_N + C_N) \\ &= (1.99 + 0.99)/(0.99 + 0.99) \\ &= 1.51 \end{aligned}$$

$$\begin{aligned} Odds(Relevant|Query) &= Odds(Relevant) \times \prod_{\forall Matched} \text{LS} \times \prod_{\forall Unmatched} \text{LN} \\ &= 1.51 \times 2.41 \times 0.39 \\ &= 1.40 \end{aligned}$$

These odds are greater than one ($p = 0.58$), and thus STAGGER incorrectly retrieves this document. Hall's (1986) paper does not concern problem solving directly and is irrelevant, thus the negative instance counts are updated: negative infirming (N_I) for present keywords and negative confirming (N_C) for missing ones. The elements' counts are then decayed, and their weights are updated. Figure 3.33 shows the concept description for (Hall, 1986) after processing these first two queries. Note that no new keywords have been added because (Hall, 1986) was not relevant.

This set of keywords is both incomplete and poorly weighted, but subsequent training improves the situation. After processing 500 queries, (Hall, 1986) has been either retrieved or relevant 29 times. Figure 3.34 displays the resulting keyword relevancy description.

These final elements point out a shortcoming in the original keyword set. This is reflected by the absence of one of the original keywords shown in Figure 3.30, **generalization**. Another

Table 3.15: Second query.

Relevant Documents	Query Keywords
(Smith, 1980) (Carbonell, 1982) (DeJong, 1982) (Korf, 1982) (Carbonell, 1983) (Keller, 1983) (Kibler & Porter, 1983) (Korf, 1983) (Mostow, 1983) (Araya, 1984) (Laird, Rosenbloom, & Newell, 1984) (O'Rorke, 1984) (Porter & Kibler, 1984) (Cheng & Carbonell, 1986) (Christensen & Korf, 1986) (Eshelman & McDermott, 1986) (Shimura & Sakurai, 1986)	application-heuristic breadth-first-search episodic explanation explanation-based fermi generalization heuristic-classification inference-engine instructional-strategy learning-from-examples memory-organization operator plan-learning problem-solving production-rule puzzle recognition relational-learning search serial-decomposibility transfer

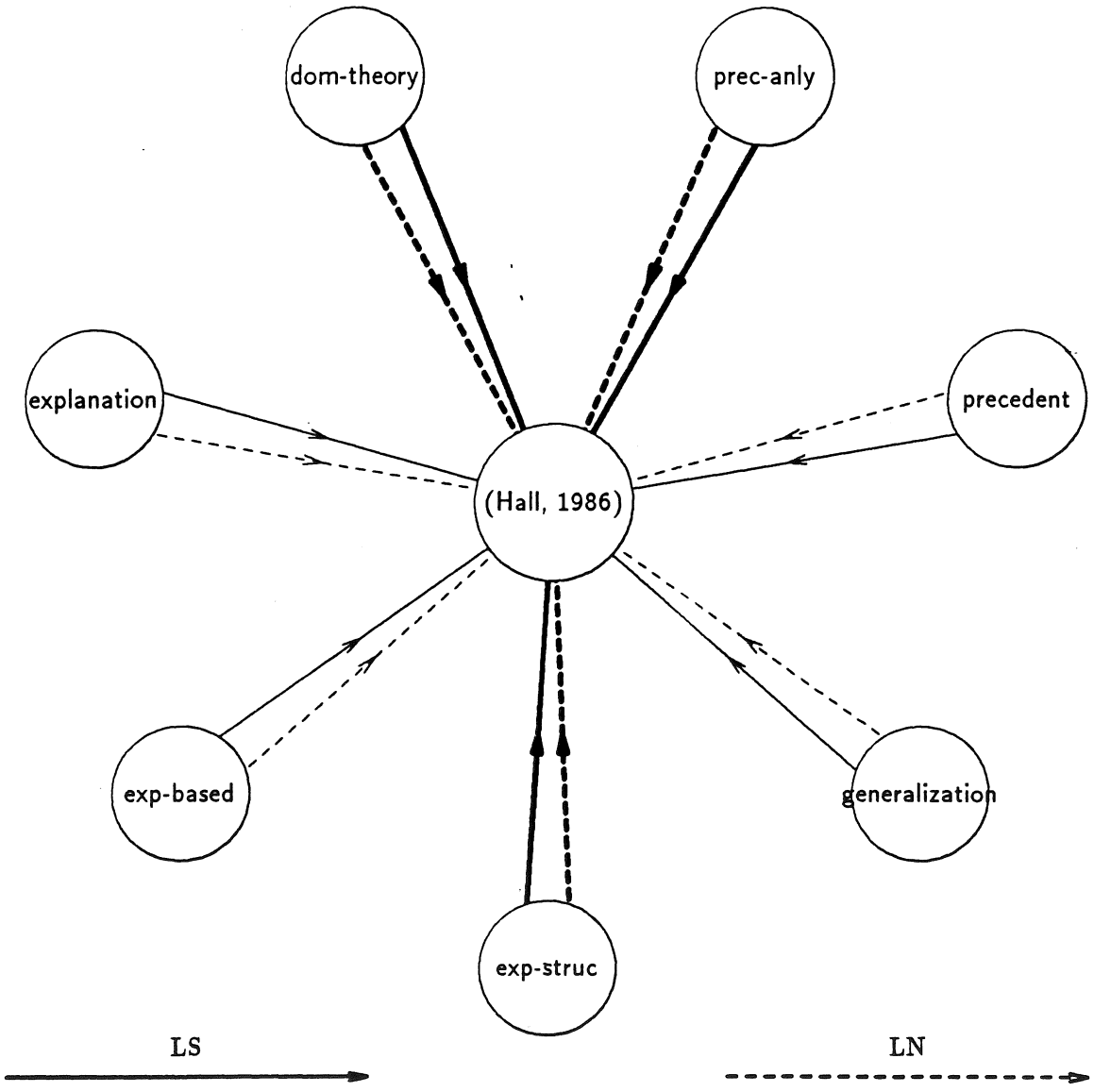

Figure 3.33: Keywords for (Hall, 1986) after processing two queries.

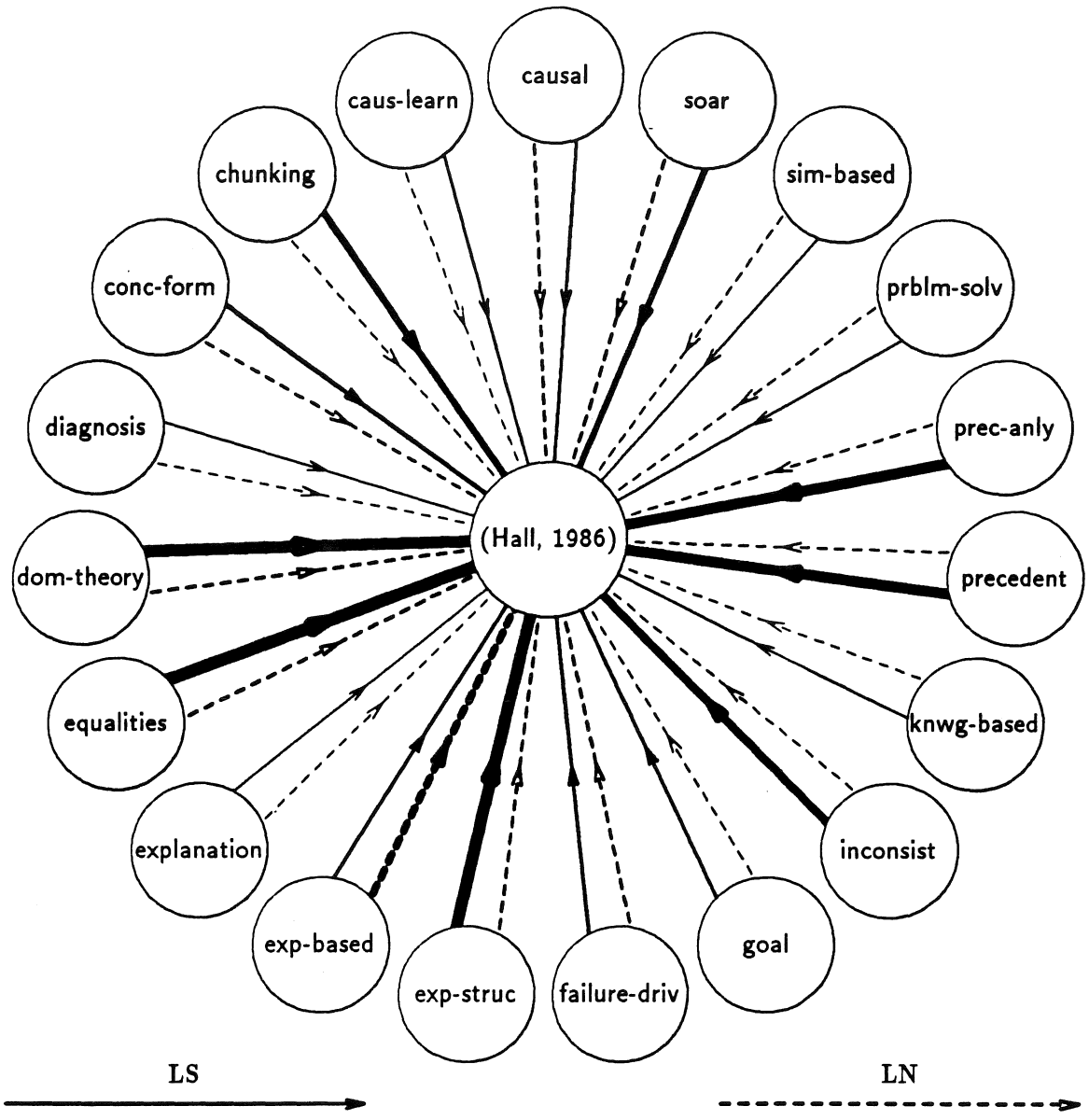

Figure 3.34: Keywords for (Hall, 1986) after processing 29 queries.

term, **explanation**, is so poorly predictive that it has little effect on the document's expectation. Collectively, these keywords occur in 39% of the documents and have little predictive power.

The inclusion of keywords not originally specified in Figure 3.30 illustrates that STAGGER has implicitly grouped documents by cross-indexing them with keywords from other related documents. Thus, in Figure 3.34, the inclusion of the terms **chunking** and **soar** operationally reflects that Rosenbloom and Laird's (1986) paper is about explanation-based learning within the Soar framework. Given the subject-oriented nature of the queries, this would be a relevant document for a query aimed at retrieving (Hall, 1986).

Carefully examining the magnitude of an individual element's counts in Table 3.16 indicates that the prior odds of relevance for this document are greatly overestimated; using the counts of the last element, $odds(relevant)$ is equal to $(13.97+8.98)/(5.98+3.99) = 2.29$ or $p(relevant) = 0.70$. This arises because the element counts are updated only when the document is either retrieved or relevant. Of the previous 500 queries, in this example, only 29 of the queries resulted in any change to this concept description, yielding a more accurate estimate on the probability of relevance of $p(relevant) = 29/500 = 0.06$. In order to maintain an effective estimate, the counts of each document would have to be updated after each query. With a reasonably large collection of documents, the inefficiencies of such a strategy dwarf any current inefficiencies in STAGGER's approach. In the matching process, this overestimation results in inappropriately high expectations and may cause the retrieval of irrelevant documents. However, the quantitative results in the next subsection indicate that this is not the case; this overexpectation is compensated by the keyword relevancy weights.

In this subsection I described the processing associated with the relevancy concept for a single document. To provide a broader picture of the system's performance, in the following final subsection I review two common quantitative indicators of information retrieval performance and then evaluate STAGGER's performance.

3.4.3 The effectiveness of automatic indexing and retrieval

In general, an information retrieval system is effective if it retrieves all of the relevant documents and none of the irrelevant ones. These intuitive notions of effectiveness have been quantified by the two measures of *recall* and *precision*. Recall is defined as the ratio of relevant documents retrieved to the total number of relevant documents. A recall value close to one indicates that most of the relevant documents have been retrieved, while a low recall (close to zero) suggests that many have been overlooked. Recall is defined as:

$$recall = \frac{|retrieved \wedge relevant|}{|relevant|} \tag{3.1}$$

Precision also has a simple definition; it is the ratio of the relevant, retrieved documents to

Table 3.16: Concept description for (Hall, 1986) after processing 29 queries.

KEYWORDS	COUNTS				WEIGHTS	
	C_P	I_N	I_P	C_N	LS	LN
explanation-structure	8.98	0.99	13.97	8.98	3.95	0.68
equalities	7.99	1.00	10.99	7.99	3.79	0.65
domain-theory	7.98	0.99	14.97	8.98	3.51	0.72
precedent	5.98	0.99	16.97	8.98	2.63	0.82
precedent-analysis	5.98	0.99	16.97	8.98	2.63	0.82
inconsistent	5.99	1.00	15.98	7.99	2.46	0.82
chunking	4.99	1.00	15.98	7.99	2.14	0.86
soar	3.00	1.00	3.00	3.00	2.00	0.67
concept-formation	7.99	2.00	12.99	6.99	1.71	0.80
explanation-based	17.97	4.98	4.98	4.98	1.57	0.43
failure-driven	10.99	3.00	9.99	5.99	1.57	0.71
goal	6.99	2.00	14.98	6.99	1.43	0.88
causal	12.98	4.00	7.99	4.99	1.39	0.69
causal-learning	2.99	1.00	17.98	7.99	1.29	0.96
problem-solving	7.99	3.00	11.99	5.99	1.20	0.90
similarity-based	10.99	4.00	9.99	4.99	1.18	0.86
knowledge-based	4.99	2.00	15.98	6.99	1.07	0.98
diagnosis	2.00	1.00	14.99	7.99	1.06	0.99
explanation	13.97	5.98	8.98	3.99	1.01	0.98

the total number of retrieved documents and ranges in value from zero to one. A low precision indicates that most of the retrieved documents are irrelevant; a high precision denotes that most are relevant. For systems that divide documents into two groups, those that are retrieved and those that are not, this measure is appropriate. However, STAGGER answers queries with an ordered list of documents. A generalized form of precision that includes the effects of ordering is called *log-precision.* Williamson, Williamson, and Lesk (1971) define it as:

$$\textit{log-precision} = \frac{\sum_{i=1}^{n} \log i}{\sum_{i=1}^{n} \log r_i} \tag{3.2}$$

where n is the number of relevant documents, r_i is the rank of the ith relevant document, and the resulting value $\in [0, 1]$. Log-precision is highest if all of the relevant documents are retrieved prior to any irrelevant ones; it is lowest if none are found. Intuitively, high log-precision indicates that the relevant documents appear early in the retrieval. When low, it suggests that if the documents are examined in order, many irrelevant ones will have to be examined before most of the relevant documents are seen.[4]

Using these two measures, we may quantitatively examine the quality of STAGGER's retrieval and learning mechanisms. Figure 3.35 depicts the changes in recall and log-precision as queries are processed. Each point represents the average of 10 random queries for each of 20 executions.

Even though STAGGER begins with relatively poor knowledge about appropriate retrieval keys, after processing 50 queries, both recall and log-precision have risen dramatically. By 150 queries, STAGGER is reliably retrieving over 80% of the relevant documents and is ordering them correctly nearly all the time.

These results are somewhat high; it is unreasonable to expect that nearly all queries will be answered with complete recall and precision (Lancaster, 1979). In large part this results from using a small document collection, two or three orders of magnitude smaller than many systems. The data are also skewed by the methodology I chose to generate queries. After a few hundred queries, STAGGER begins to review queries it has seen before. In informal experiments with query generation methods that yield greater diversity, I found that the quality of recall and precision varied somewhat. For instance, with a high variance in the generation of query keywords, STAGGER is unable to retrieve documents with the exceptionally high precision seen in Figure 3.35.

[4]Dennis Kibler pointed out that perhaps $\sum i/r_i$ would be better than Equation 3.2 since it is also sensitive to retrieval order within the relevant documents. For instance, Equation 3.2 is high even if the relevant documents are retrieved in inverse order, so long as they all precede the retrieval of any irrelevant documents.

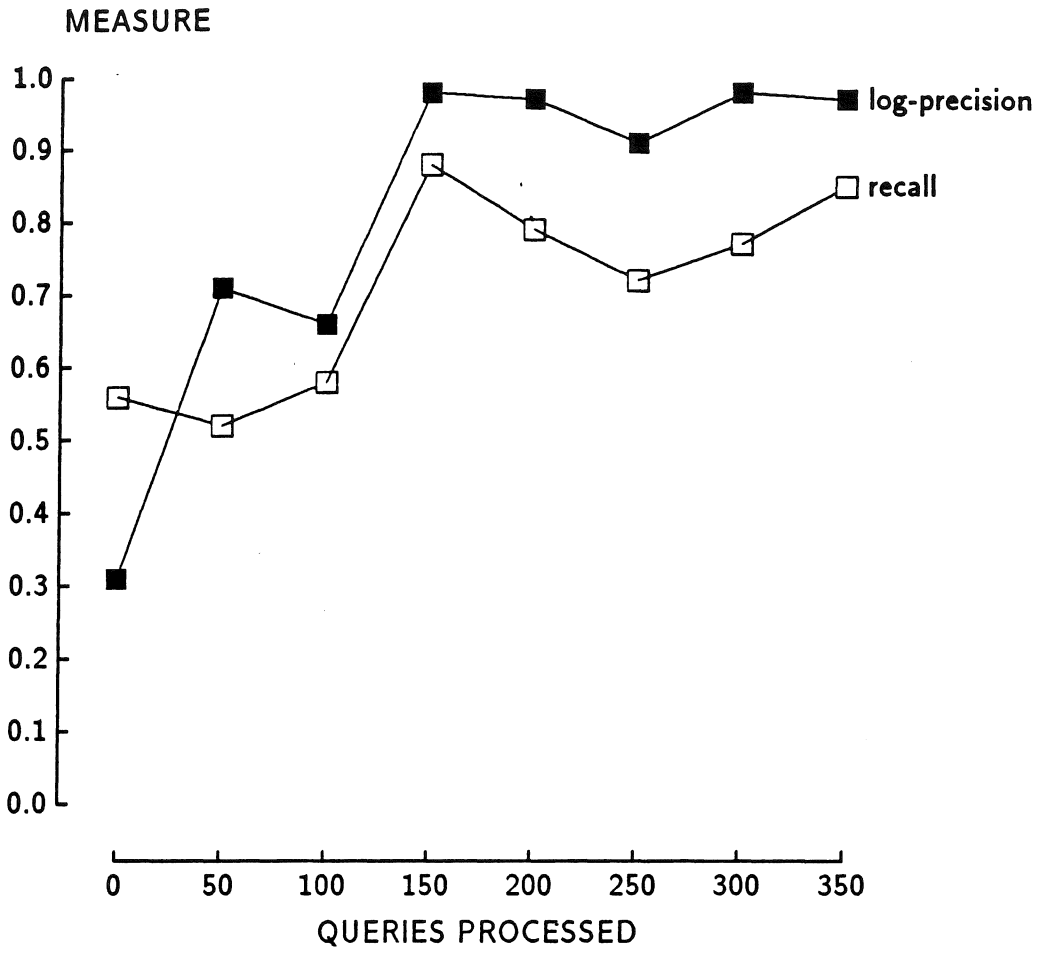

Figure 3.35: Recall and log-precision as a function of queries processed.

Though STAGGER reaches a high level of recall and precision, the approach described here falls into a subtle and infrequent form of retrieval ambiguity called *false associations* (Lancaster, 1979). For instance, given a query about welding copper, STAGGER would retrieve a document with the keywords **welding, aluminum, cleaning,** and **copper** even if the document were only about aluminum welding and copper cleaning. This issue can arise whenever there are interelations between a document's indexing keywords. The current data set of keywords and documents probably does not exhibit this ambiguity; the papers are all rather short, and this brevity can force consistency. For STAGGER's part, its current representation language of weighted keywords is not sufficient to represent all types of interrelation. If STAGGER could index the document with (welding $\wedge$ aluminum) $\vee$ (cleaning $\wedge$ copper) then it would be able to avoid this ambiguity. Along with the arguments presented in the following section, this is a motivating reason for developing the Boolean learning method described in Chapters 4 and 5.

On the positive side, the ability of STAGGER to add and test new keywords relaxes an uncomfortably tight constraint on many information-retrieval systems. Lancaster (1979) notes that perhaps the single most important subtask in developing a system is assigning appropriate keywords to documents. Although STAGGER's formulation assumes a solution to this subtask, it reduces the importance of performing it perfectly, because inappropriate keywords are eliminated and omitted terms are included in a natural manner. Note that the performance depicted in Figure 3.35 began with impoverished document representations. Only 1/4 of the keywords were used to simulate indexing and matching performance with a poor keyword extraction process.

In summary, this subsection has shown that STAGGER effectively adapts document relevancy descriptions to match the usage of their keywords. The two measures of recall and log-precision demonstrated a reduction in retrieval errors of omission and commission, respectively. Furthermore, the high value of these two measures indicated that a user's queries are answered by retrievals that are complete and consistent. Additionally, an adaptive solution like STAGGER reduces the importance of a perfect keyword extraction technique, allowing the system to bolster the indexes for individual documents as needed.

3.4.4 Conclusion

In this section I mapped the task of retrieving relevant documents given a set of keywords into the task of learning from examples. Within information retrieval, STAGGER focuses on the processes of indexing and retrieving documents, assuming solutions to the process of assigning keywords to documents and interpreting user's queries. The basis for the experiment is a set of 77 learning papers drawn from the AAAI conferences. Each document is represented as a relevancy concept; the elements of the concept description are drawn from potentially

predictive keywords. In order to simulate use of the system in a dynamic manner, queries are generated following a subject orientation. Errors in the process of assigning keywords to documents are overcome by dynamically adding omitted keywords and eliminating poor ones. STAGGER is able to significantly improve its performance over time, resulting in queries which are completely and consistently answered.

As with any illustration, the centerpiece of this section has the problem that it provides a biased indication of STAGGER's adaptive capability. In the next section I describe an objective set of tasks that better illustrate both the advantages and disadvantages of the methodology herein.

3.5 A comprehensive learning task

In the previous sections, I described STAGGER's performance in three domains: simple object concepts, two-dimensional spatial concepts, and document relevancy concepts. These example tasks served to indicate the method's generality. What they fail to do is convey completeness. A proof of correctness provides a convincing argument for a method's effectiveness, but a proof can be difficult to formulate. As a substitute, in this section I provide some comprehensive evidence that indicates the capabilities and limitations of the weight adjusting method used in STAGGER. First I define a simple, abstract domain and then examine the possible concepts. By eliminating redundancies, the large number of concepts is reduced to a smaller, tractable number. By examining learning performance over these concept types, we can determine for which class of concepts the method is applicable. For each of these concept types, I describe STAGGER's ability to capture the implied concept. The results indicate that the Bayesian weighting method is limited; disallowing weighted *combinations* of attribute-values inhibits successful learning of many concept types.

Consider a simple domain of three attributes, each with one of two values: $A_1 \in \{V_{1,1}, V_{1,2}\}$, $A_2 \in \{V_{2,1}, V_{2,2}\}$, $A_3 \in \{V_{3,1}, V_{3,2}\}$. By multiplying the number of values for each attribute together we see that there are 8 theoretically possible objects ($2 \times 2 \times 2 = 8$). They are listed in Table 3.17.

Each discrete concept in this world may be represented as a set of objects. Therefore, the number of possible concepts is the number of object subsets, or $2^8 = 256$. However, given that the attributes and their values were arbitrarily named several of these possibilities may be translated into one another through name substitution. Specifically, two concepts are equivalent if the objects in one can be transformed into the objects in the other by uniformly replacing attribute-value names. For instance, the subset of objects {1, 4} (object 1 with $V_{1,1}$, $V_{2,1}$, $V_{3,1}$, and object 4 with $V_{1,1}$, $V_{2,2}$, $V_{3,2}$) have the same value for the first attribute and different values for each of the remaining attributes. Similarly, the subset of objects {3, 8}

Table 3.17: Eight possible objects with three binary-valued attributes.

	$A_1 =$	$A_2 =$	$A_3 =$
1.	$V_{1,1}$	$V_{2,1}$	$V_{3,1}$
2.	$V_{1,1}$	$V_{2,1}$	$V_{3,2}$
3.	$V_{1,1}$	$V_{2,2}$	$V_{3,1}$
4.	$V_{1,1}$	$V_{2,2}$	$V_{3,2}$
5.	$V_{1,2}$	$V_{2,1}$	$V_{3,1}$
6.	$V_{1,2}$	$V_{2,1}$	$V_{3,2}$
7.	$V_{1,2}$	$V_{2,2}$	$V_{3,1}$
8.	$V_{1,2}$	$V_{2,2}$	$V_{3,2}$

(object 3 with $V_{1,1}$, $V_{2,2}$, $V_{3,1}$, and object 8 with $V_{1,2}$, $V_{2,2}$, $V_{3,2}$) also share a value for only one attribute. By allowing substitution of attribute and value names, these two subsets are identical. Repeating this type of analysis yields a group of 12 subsets of two objects, all which have only one attribute with a common value. Of the remaining two object subsets, 4 have completely disjoint attribute-value pairs, and 12 more have two attributes with a common value. Reducing the 256 concepts by this method results in 19 concept types. Table 3.18 lists these concept types, the subset sizes, a description of each concept type, and an example of the type.

Each of these types is defined in terms of the commonality of values among variables. For subsets containing three objects, all of the attributes must share at least two values, because the number of objects outnumbers the number of values. Some subsets of three objects show additional regularity and have one attribute with three common values. Because all of the objects within a subset of size three are distinct, it is not possible for two or three attributes to have three common values.

Several of the larger concept types are defined in terms of the objects that are missing from the subset. Thus, 5a is defined as any subset of five objects that excludes two objects with completely disjoint attribute values.

Each of these concept types may be mapped onto Boolean descriptions. The easiest two are the empty set (0) and the complete set (8) which map onto **false** and **true**, respectively. Type 4e is also simple, for it maps into $A_1 = V_{1,1}$. A simple conjunction is represented among the subset classes; 2c requires that $A_1 = V_{1,1} \wedge A_2 = V_{2,1}$. More complex descriptions are also represented, including exclusive-or (4a): $A_1 = V_{1,1} \oplus A_2 = V_{2,1}$. In fact, all possible Boolean descriptions over objects with three binary-value attributes are represented by one of the 19 concept types listed in Table 3.18. Thus, we may comprehensively examine the learning behavior of a procedure by assessing its performance over these 19 types.

To test the ability of the weight-altering method of Chapter 2 on these 19 cases, a single

Table 3.18: Nineteen possible concept types.

	Num. of Objects	Description	Example Subset
0.	0	No objects.	{}
1.	1	One object.	{1}
2a.	2	No attribute has a common value.	{1, 8}
2b.	2	One attribute has one common value.	{1, 4}
2c.	2	Two attributes have one common value.	{1, 2}
3a.	3	All attributes have two common values.	{1, 2, 8}
3b.	3	One attribute has three common values.	{1, 2, 3}
4a.	4	All attributes have two common values.	{3, 4, 5, 6}
4b.	4	One attribute has three common values.	{1, 2, 3, 8}
4c.	4	Two attributes have three common values.	{1, 3, 4, 8}
4d.	4	All attributes have three common values.	{1, 2, 3, 5}
4e.	4	One attribute has four common values.	{1, 2, 3, 4}
5a.	5	Inversion of 3a.	{3, 4, 5, 6, 7}
5b.	5	Inversion of 3b.	{4, 5, 6, 7, 8}
6a.	6	Inversion of 2a.	{2, 3, 4, 5, 6, 7}
6b.	6	Inversion of 2b.	{2, 3, 5, 6, 7, 8}
6c.	6	Inversion of 2c.	{3, 4, 5, 6, 7, 8}
7.	7	Inversion of 1.	{2, 3, 4, 5, 6, 7, 8}
8.	8	Inversion of 0.	{1, 2, 3, 4, 5, 6, 7, 8}

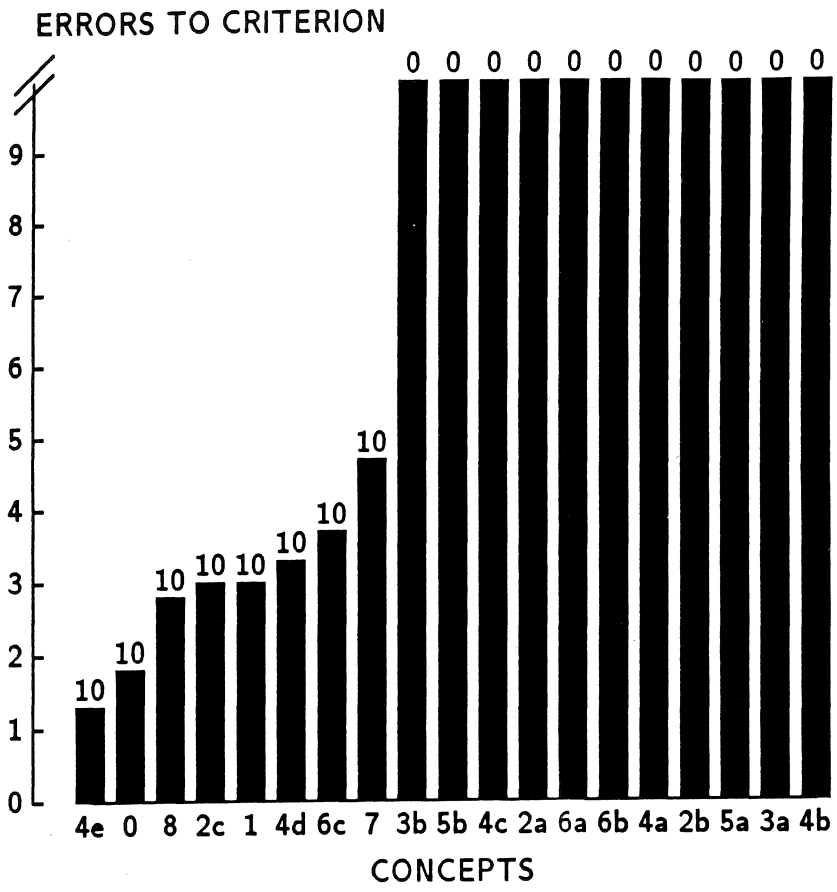

Figure 3.36: Learnable concepts with weight adjustment in STAGGER.

concept description was formed with a representational element for each attribute-value pair. As before, these elements are initially assigned unbiased matching counts, and as new objects are classified as either in the subset concept or not, these counts and their weights were updated. Figure 3.36 displays the resulting performance for each of the 19 types listed in Table 3.18. Each column indicates an average over 10 executions of up to 4,000 examples, and the number at the top indicates the number of executions that converged on a correct concept description.

As a mapping of the subset classes into Boolean descriptions would imply, some of the concepts represented in Table 3.18 are easily learned by the weight modification method in STAGGER (to the left in Figure 3.5), while others are impossible (those to the right). The method seems quite capable of acquiring the concepts **false** (type 0), **true** (8), and $A_1 = V_{1,1} \wedge A_2 = V_{2,1}$ (2c). In each of these cases, the numeral 10 topping each dark column indicates that the Bayesian weighting method converged to a correct description. However, the exclusive-or concept of subset 4a is not learned; in none of the ten cases did the weighting-adjusting method derive a correct concept.

The inability of the Bayesian weight-learning method to acquire the exclusive-or concept is a representational deficiency, not a learning deficiency. In order to represent $A_1 = V_{1,1} \oplus A_2 = V_{2,2}$, matching expectation must be high when $A_1 = V_{1,1}$ is true and high when $A_2 = V_{2,2}$ is true. Contrariwise, matching expectation must be low when they are both true, and this is impossible given that the magnitude of expectation is metered by weights that ignore the presence or absence of *combinations* of attribute-values. Again, each of the weights must be strong to increase expectation when they are singly matched, yet these weights in combination must decrease expectation when they are conjointly satisfied. I conjecture that the weight method in STAGGER is only capable of acquiring linearly-separable concepts (Duda & Hart, 1973).

One remedy is to assign weights to elements that represent combinations of attribute-values. For instance, in the exclusive-or case, if an element representing $A_1 = V_{1,1} \wedge A_2 = V_{2,2}$ had an inhibitive matching weight, then the exclusive-or concept could be adequately represented. In the following chapter, I introduce a method embodying this solution; it constructs concept description elements representing Boolean combinations of attribute-values. In Chapter 5 I then show that the Boolean chunk learning method allows the weight modification method to acquire and represent the 19 concepts enumerated above.

One unrealistic assumption I have been making with this domain is that all of the possible objects are equally likely. This is not typically the case with real-world classification tasks. For instance, though it is possible to describe an animal covered with feathers that nurses its young, no such creatures exist.

Second, this domain has so few objects in it, and STAGGER learns relatively slowly, that it stretches the definition of induction when each test object has already been observed. A simple method that remembered each object it had seen would be able to perform this task without having to process each object more than once. However, the generality exhibited by STAGGER in each of the previous examples of this chapter illustrate that it is able to induce a description given only some of the examples. In the first example, STAGGER correctly categorized all of the objects after seeing 7/9 of them. Similarly, after visiting about 300 locations in the two-dimensional world, STAGGER successfully navigates through all 10,000 of them.

In summary, any set of demonstrations suffers from the problem that they provide a biased interpretation of the system's abilities. The most objective assessment is a proof of correctness, but one of those would be difficult to construct given the complexity of the methods I describe here. Instead, I presented a simple domain of three binary-valued attributes and collapsed all of the possible concepts into 19 interesting cases. The weight modification method of Chapter 2 is representationally insufficient for a number of these concepts, and in the next chapter I describe a Boolean learning method which remedies this limitation.

3.6 Overview

The basis of the learning methods employed in STAGGER is the representation of a concept description as a set of independently weighted elements. Each element has two weights: one that increases expectation of a positive example when it matches the example and one that decreases expectation when it does not match. The underpinning of these weights is a set of four counts that reflect the frequencies of each of the possible matching events between an element and example. These counts are subjected to a small decay, and thus they are able to gracefully discard previous experience as the environment changes. In this chapter I have demonstrated each of these qualities through a series of four diverse and detailed examples.

The first example served as a precise introduction to the operation of the weight adjustment method. By building a prototypical concept description, STAGGER is able to correctly classify novel objects. However, the weighted attribute-value representation has the drawback in that it is unclear; it is difficult to tell with a casual inspection when the attribute-value weights will cooperatively indicate a positive or negative example. Adding a more explicit representation through Boolean combinations of attribute-values may improve the comprehensibility of the concept descriptions.

The second example demonstrated learning about a goal location in a two-dimensional world. Unlike the simple object concept learning task, this requires multiple concepts and involves real-valued inputs. Though the task encompasses 10,000 separate locations, STAGGER forms an effective description of the goal after visiting as few as 500 and moves toward it from novel locations without any feedback. Moreover, if this world changes, e.g., the goal or landmarks move, the weight modification method readily adapts, for the numerical underpinnings of the LS and LN weights are naturally adjusted to reflect current experience.

In the third example, STAGGER improved its retrieval of documents given a set of query keywords. The method begins with an impoverished description of each paper, adding valuable keywords that are omitted and deleting useless ones. The two quantitative measures of recall and log-precision both indicate an improvement in the quality of retrievals. The need for Boolean combinations of keywords arises in this example, for there is a subtle type of retrieval ambiguity error wherein the interrelations between document keywords may not be adequately represented.

The final example tested the behavior of the weight adjustment method on all possible concepts over objects with three binary-valued attributes. This artificial task provides a means of performance analysis that, although less satisfying than a proof of correctness, gives some objective evidence of the method's strengths and weaknesses. Those concepts that cannot be acquired indicate a representational deficiency, which may be overcome by the addition of Boolean combinations of attribute-values.

Hopefully, these four examples have established three major points in the reader's mind.

First, the weight adjustment method is general and flexible. It is relatively domain independent and can be easily extended to novel problem types. Second, this learning method alone is insufficient for many types of concepts since it lacks a mechanism for representing associations between combinations of attribute-values and the identification of examples. Third, the method's limitations, capabilities, and applications are clearly illustrated through the use of both artificial and naturally constrained learning tasks.

Having discussed these four examples in light of the weight-learning method, in the next chapter I present a method which addresses the need seen consistently throughout these examples: the ability to learn in terms of combinations of attribute-values.

Chapter 4

Boolean Chunk Learning

4.1 Introduction

In Chapter 2, I discussed the representation of learned concepts, their use in matching, and adjustment of the weights attached to each of the concept description elements. Several demonstrations of this capability in STAGGER were given in Chapter 3, yet some limitations were also observed. As Section 3.5 describes, there are some concepts the Bayesian weight method cannot acquire – those that cannot be described as linearly separable concepts (Duda & Hart, 1973). To tackle more complex concept learning tasks, some mechanism must be formulated that allows one to express and discover non-linearly separable concepts. In this chapter, I describe an approach that formulates explicit Boolean combinations (called *chunks*) from pre-existing elements, enabling the Bayesian weight learning method to represent a concept that is any Boolean function of its inputs. In subsequent chapters, I present empirical evidence that combining the resulting method with the Bayesian weight learning method alleviates many shortcomings of the latter.

4.1.1 The introduction, testing, and refinement of new chunks

STAGGER's Boolean chunking incrementally adds new elements to the concept description by conjoining, disjoining, or negating existing elements. For instance, if **small** and **red** were two elements in the concept description, a new chunk representing **small** $\wedge$ **red** might be added. The choices of a Boolean function, and arguments for it, are based on heuristics that examine matching errors and concept element weights.

Figure 4.1 outlines several processes that operate on the newly introduced chunks. The new chunk may be pruned from (Box 2) or inaugurated into (Box 3) the concept description. Following the latter case, the chunk may trigger backtracking (Box 4), which may in turn lead to further pruning (Box 5). Since this process iterates, chunks may serve as compo-

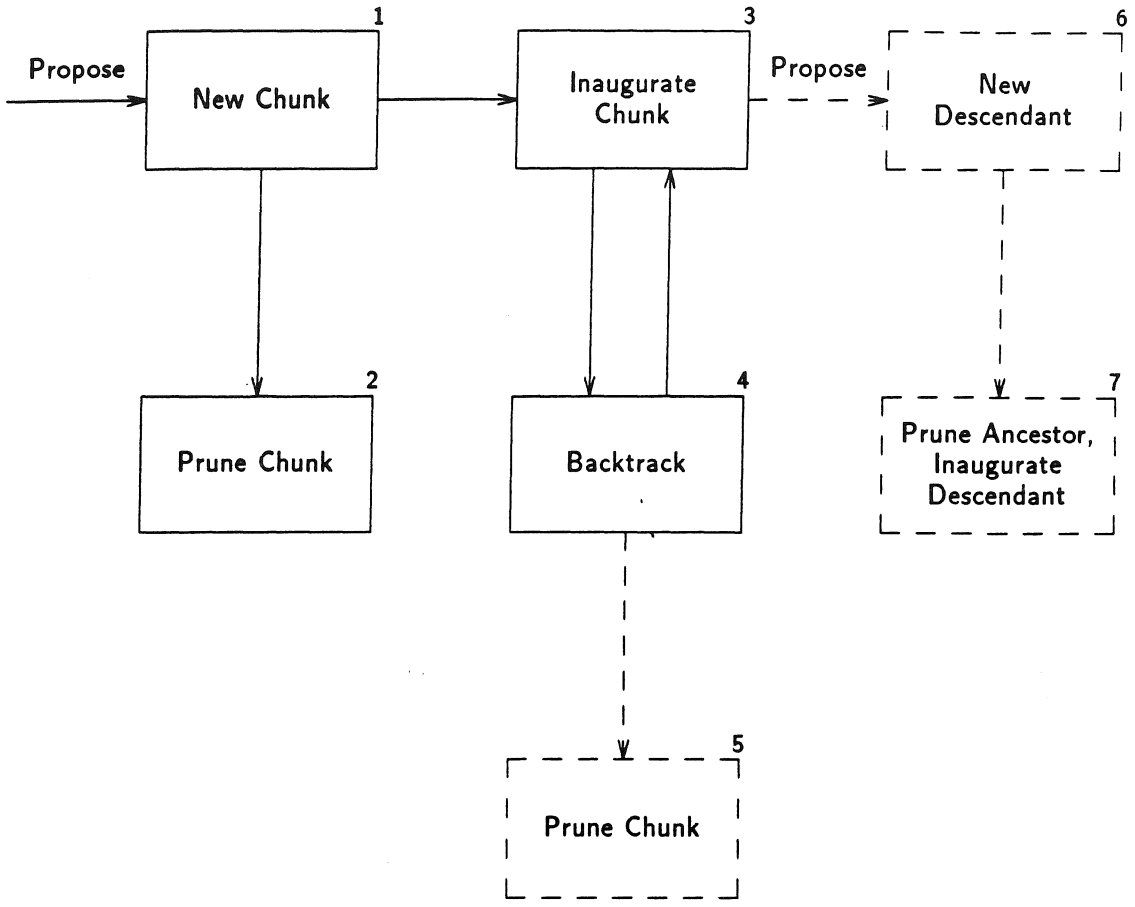

Figure 4.1: Simple schematic of a chunk's lifecycle.

nents for larger chunks. Box 6 represents an instantiation of Box 1 when a large chunk is proposed. Similarly, if the larger chunk proves effective, it is inaugurated (Box 7) as with Box 3. Additionally, the intermediate component chunks are pruned, leaving room in the concept description for future expansion. As this chapter describes, most of the transitions in Figure 4.1 are driven by changes in the chunks' weights.

In general, pruning serves to clean up the concept description and limit computation. Some existing elements are removed along with other newly added elements. Only a frontier of effective chunks is maintained and, over time, the mechanisms outlined in Figure 4.1 condense the distributed concept description into a unified one. This process diagram serves as a rough outline for the remaining sections of the chapter.

4.1.2 A search metaphor for Boolean chunking

The process of adding new Boolean chunks may be described in terms of search. States in the search space consist of all possible chunks. The operators that traverse this space are

Boolean conjunction, disjunction, and negation. Figure 4.2 shows a small portion of the search space for a simple domain of three attributes (size, color, and shape). As the figure indicates, the possible Boolean functions are partially ordered along a dimension of generality (Mitchell, 1982); nodes higher are more *specific* while more *general* nodes are below. Drawing on this observation, the Boolean operators of conjunction and disjunction correspond to search operators of generalization and specialization, respectively. Negation has a different flavor; it does not have a consistent effect on generality. Potential applications of the negation operator are shown by dashed lines in Figure 4.2. STAGGER's initial search frontier (its initial concept description) is the set of single attribute-value pairs like those highlighted in Figure 4.2.

This size of this space is exponentially larger than the space typically searched by a conjunction-only method like version spaces (Mitchell, 1982). Another difference is that the version space method searches from both sides toward the middle; STAGGER uses the Boolean operators of conjunction, disjunction, and negation (described more fully below) to search from the middle outward toward both edges.

4.1.3 Combining features as Boolean chunking

Before proceeding further with the description of the processes that form, test, and refine the Boolean feature combinations, let us consider the use of the term *chunk* to refer to a Boolean combination of simple attribute-value patterns. An early use of the term is due to Miller (1956), who recounts several experiments that illustrate a remarkable memory limitation phenomena: a dramatic increase in immediate memory capacity as the items to be recalled take on semantic content. For instance, one study showed that subjects could recall only about seven randomly presented letters of the alphabet. However, if the letters were instead grouped according to familiar acronyms (e.g., TV, FBI, YMCA), then subjects could accurately recall more of the letters; the subjects appeared to be limited to about seven acronyms. The primary hypothesis advanced to account for these results was that subjects in the second setting were combining the simple alphabetic input into more meaningful semantic units called *chunks*. This hypothesis has received considerable attention and has proved theoretically fruitful (e.g., Bower, 1972; Chase & Simon, 1973; Chase & Ericsson, 1981; Laird, Rosenbloom, & Newell, 1986).

Combining existing elements into Boolean conjunctions, disjunctions, and negations is also a type of chunk formation. These new combinations recode the input to the Bayesian weight method, allowing it to form weighted concept descriptions in terms of these higher level "features" rather than the low-level features in the input.[1] The net result is that the combined

[1]Throughout this chapter the term *chunk* refers to a combination of features, while the more general term *element* is used to refer to any weighted pattern of the attribute-values.

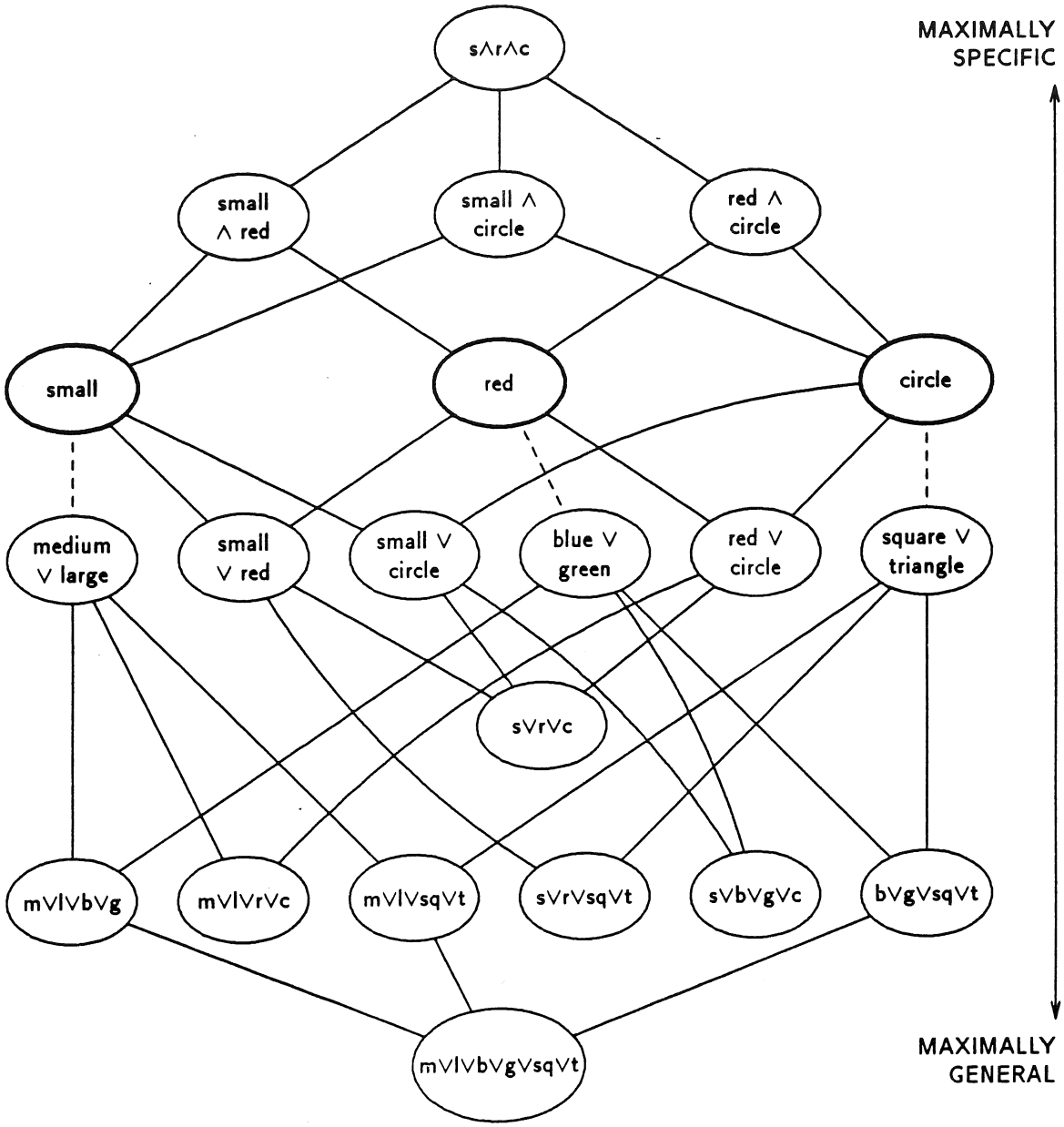

Figure 4.2: Part of the space of Boolean chunks.

methods can learn a larger class of concept types than the Bayesian weighting method alone. Put another way, the chunking method is rewriting the input to the Bayesian weight learning method, allowing the latter to search a larger space of possible concepts. The increase in learning capability is demonstrated clearly by the contrast between the results of Section 3.5 and Section 5.5 of this chapter.

4.2 Boolean search operators

STAGGER's three search operators introduce new chunks into the concept description by specializing, generalizing, or inverting existing elements after expectation errors. Referring to Figure 4.1, this corresponds to the paths into Box 1 (for combinations of simple elements) and into Box 6 (for more complex chunks). To make an element more specific, the Boolean operator of conjunction adds a new AND of two good elements currently in the search frontier. For instance, this operator may lead from the nodes labeled **small** and **red** in Figure 4.2 to the node labeled **small** $\wedge$ **red**. Conversely, the generalizing operator uses the Boolean disjunction operator to form a new OR from a pair of existing elements (e.g., **small** and **red** disjoined to **small** $\vee$ **red**). The inversion operator, which may raise or lower generality, uses the Boolean negation operator to form new NOT chunks for the concept description (e.g., **small** is negated to form **medium** $\vee$ **large**).

4.2.1 Heuristic operator selection

The Boolean conjunction, disjunction, and negation operators are not applied exhaustively; expanding search is limited by adding new chunks to the concept description only when STAGGER makes a matching error. These errors indicate that the Bayesian weighing method could use some help in describing the concept. The type of matching error also serves as a heuristic for identifying an appropriate operator. For instance, when a nonexample is mistakenly predicted to be a positive example (an error of predictive commission), the concept representation is too general; it is including too many examples in the positive class. A new concept description element is needed: one that is less admitting, more discriminating, and would not have allowed the matching process to falsely expect that this nonexample was positive. In this case specialization proceeds by applying the Boolean conjunction operator. It forms a more restrictive concept element, since the conjunction is true of fewer examples than the component elements. Adding a new conjunctive chunk to the concept description search frontier effectively stretches it upward (see Figure 4.2).

On the other hand, an erroneous guess that a positive example is negative (an error of predictive omission) is overly specific; an example that should been included in the positive class was not. To correct the overly exclusive concept description, search expands toward a

Table 4.1: Operator preconditions.

EXPECTED	ACTUAL	MATCHING ERROR	OPERATORS	
			SEARCH	BOOLEAN
Positive	Negative	Commission	Specialize	Conjunction
Negative	Positive	Omission	Generalize	Disjunction
—	—	Either	Invert	Negation

new, more inclusive chunk. In this case, disjunction is applied as a generalization operator since the OR of a pair of elements results in more general one (i.e., it is true more often and is satisfied by more examples). Applying this operator corresponds to moving downward in Figure 4.2.

A weaker heuristic condition specifies that after either type of matching error, STAGGER should expand the frontier of the concept description by applying negation. In general, negating an element does not change its generality. Table 4.1 summarizes this heuristic for operator preconditions.

4.2.2 Heuristic argument selection

Half of the task of adding a potentially useful element to the concept description is identifying the appropriate Boolean operator to apply. The second half is selecting existing elements to serve as arguments for that operator: choosing a site in the search space for expansion. A pair of heuristics for this latter selection cooperatively act as a two-pass process. One heuristic identifies coarse groupings of concept elements to be combined; another narrows the groupings down to the best possibilities.

Grouping arguments

Specialization The first pass criterion for selecting the required elements is aligned with the preconditions for the operators themselves. Consider the case of an error of predictive commission. In this case STAGGER has mistakenly predicted that a nonexample is positive and is therefore behaving too generally. The operator preconditions discussed above prescribe the formation of a conjunction. Since a conjunction is only true when all of its contributing elements are true, only elements of high necessity should be included, i.e., those that are required of positive examples. Conjoining an element that is not necessary would be overly restrictive, leading to the faulty exclusion of positive examples. However, in the task of learning a class description from examples, the environment does not identify necessary elements. STAGGER uses heuristics for this purpose.

Heuristically identifying necessary elements is based on the observation that some necessary elements may have been matched, but at least one necessary element was not. If this latter statement were not true (i.e., all necessary elements were matched), then this nonexample would have been positive. A straightforward strategy would be to combine two unmatched elements. However, including a matched element has an additional recommendation. Since STAGGER was misled to expect a positive example, some element it considered important was matched. Therefore, STAGGER selects one matched and one unmatched concept description element as arguments for the Boolean conjunction operator.

Negation (i) Following an error of predictive commission, the operator preconditions suggest the formation of a new NOT. Negation is used to invert concept description elements that are negatively predictive: those which match or predict nonexamples. Inverting makes them positively predictive, causing them to match positive examples. In some sense, these elements do not need to be inverted, for their element weights effectively encode negative prediction. An element with very low LS or very high LN weights effectively inhibits positive expectation when it is matched (the low LS reduces expectation) and encourages expectation when it is unmatched (high LN increases expectation). However, it is desirable to apply a consistent interpretation to the element weights, and therefore the search is biased toward positively predictive elements. The unitary argument of the Boolean negation operator is selected from the concept description elements which matched this nonexample, i.e., those that are negatively predictive.

Generalization In the case of an error of predictive omission, where a positive example was incorrectly expected to be negative, STAGGER is behaving too specifically. As described above, an application of the Boolean disjunction operator is prescribed. Since the disjunction operator results in a new chunk that is true when any of the elements are matched, only sufficient elements should be included. If non-sufficient elements were added to a disjunction, the resulting chunk would incorrectly include nonexamples. Following the reasoning outlined in discussing the arguments for the conjunction operator, in this positive example at least one sufficient element must have been matched, but some may have been unmatched. If none of the sufficient concept description elements were matched, then this example would not have been positive. The disjunction could be formed from two matched elements, but including a matched and a matched element has the advantage: it may cover the weighty element that kept STAGGER from an accurate expectation.

Negation (ii) Following an error of predictive omission, the Boolean negation operator is also applied. This operator inverts negatively predictive elements (i.e., those which fail to

Table 4.2: Nomination heuristics.

MATCHING ERROR	OPERATORS		ARGUMENTS
	SEARCH	BOOLEAN	
Commission	Specialization	Conjunction	Matched, Unmatched
	Inversion	Negation	Matched
Omission	Generalization	Disjunction	Matched, Unmatched
	Inversion	Negation	Unmatched

match in examples). Given that this positive example was incorrectly identified as negative, an unmatching concept description element is selected as the source for the new negation.

These argument heuristics may be likened to a portion of the political process: potential candidates are nominated to run for one of several offices. Similarly, concept description elements are selected to fill one of the argument slots of a Boolean operator. For convenience, the first pass selection heuristics are termed *nomination* heuristics. Table 4.2 summarizes these.

Selecting arguments

The nomination heuristics serve to separate necessary, sufficient, and negatively predictive elements into coarse groupings of Boolean arguments. For instance, it would be possible to form a conjunction consisting of all matched elements and all unmatched elements after an error of commission. However, this would be useless since the resulting conjunction would never be true. An additional set of heuristics must be consulted to determine the best of the possible arguments. Drawing freely on the political simile, I will refer to this second layer of selection as the *election* heuristics. Like the process by which the electorate chooses their favorite candidates, the element weights will indicate the best elements.

Unlike the nomination heuristics, electing the best concept description elements is not linked to the type of matching error. Instead, the heuristics for the final pass of argument selection are based solely on the operator itself. For instance, conjunction is most effective if used to combine necessary elements, or those elements that must be present for the example to be positive. By examining the counts of matching events accrued by necessary elements, it is possible to make qualitative assertions about the values of their LS and LN weights. Specifically, a necessary element never fails to match a positive example. Referring to Table 2.4, we see that this amounts to an absence of positive infirming (I_P) evidence for necessary elements. The other type of infirming evidence, I_N, may either be small or large, for the necessary element may or may not appear frequently in nonexamples.

Consider the implied values of the two weighting measures LS and LN for a necessary concept description element. Equation 2.6 for these weights is reproduced below for convenience.

$$LS = \frac{C_P(I_N + C_N)}{I_N(C_P + I_P)} \qquad LN = \frac{I_P(I_N + C_N)}{C_N(C_P + I_P)} \tag{4.1}$$

For a necessary element, nothing general can be said about the value of the LS weight; it is inversely proportional the amount of negative infirming evidence, and in general no assertion can be made regarding the value of that amount. However, the LN weight will be significant. If there is little positive infirming evidence (and I_P is close to zero), LN will be significant ($\ll 1$). LN is termed logical necessity after this property of signaling necessary elements.[2] Therefore, following the prescription of conjunction and the nomination of matched and unmatched concept description elements, the two elements with the smallest LN values are conjoined to form a new chunk.

The heuristics follow a similar line of reasoning for electing the best elements nominated as disjunction arguments. Disjunction combines sufficient elements. These are never present in a nonexample, but may be missing from a positive example. In terms of the infirming evidence accruing to each sufficient element, they will have little or no I_N, but may have any amount of I_P. The LS measure is inversely proportional to the sum of negative infirming evidence – when this value is small, LS is large. LS is thus named logical sufficiency as a result of its ability to identify sufficient elements. The dual measure, LN, is not constrained, for it varies in proportion with the unconstrained sum of positive infirming evidence. The election of nominated elements as arguments to disjunction is based on the LS weights of those elements: the highest LS weighted elements are elected as the disjuncts for a new OR.

When electing an argument for the Boolean negation operator, the election heuristics do not specify a preference of one weighting measure over the other. Although each negatively predictive element has a large amount of each type of infirming evidence, in general there is no significant difference between the positive and negative infirming sums. With the weights on equal footing, the element with either the lowest LS or the highest LN value is elected. A direct comparison can be made between LS and LN for this purpose by inverting the LN measure and selecting the element with the smaller of the two. Table 4.3 summarizes the second step candidate election heuristics.

In summary, STAGGER searches from simple toward complicated descriptions, stopping when the elements accurately describe the concept. The concept description elements are as simple as possible, and this preference for parsimony is a major source of bias. STAGGER employs three search operators, generalization, specialization, and inversion. These are trig-

[2]Because of the nature of correlational-based empirical learning, it may also be the case that coincidental elements appear to be necessary. However, the LN weighting measure will always identify genuinely necessary elements as such.

Table 4.3: Election heuristics.

BOOLEAN OPERATOR	ARGUMENTS
Conjunction	LN $\ll 1$
Disjunction	LS $\gg 1$
Negation	LS $\ll 1$ or LN $\gg 1$

gered by matching errors and use a pair of heuristics to select arguments to be applied by the Boolean operators of conjunction, disjunction, and negation.

4.3 Memory limitations

The chunk formation processes provide for growth in the search frontier, but without pruning, the concept description can become too large. To limit this growth, STAGGER forces each new chunk to compete with its formative elements. The basic idea is to retain only those chunks that the Bayesian weight method finds useful for its concept description task. In this section I describe the competition between component elements and the new chunk, how poor chunks are eliminated, and how good chunks are retained. As part of the overall lifecycle of a chunk, Figure 4.1 depicts this potential transition from Box 1 to either Box 2 or Box 3. In general, the goal of the competition is to limit the distributed concept description size, both in growth and decay, without relying on domain specific assumptions.

4.3.1 Competition between new chunks and their constituents

When a new chunk is proposed by the Boolean operators, it is not immediately added into the concept description. Instead it progresses through two phases: testing and incorporation. The first phase is based on a simple competition between the new chunk and the elements that served as arguments for the Boolean operator that formed it. This competition takes the form of a threshold assessed against the weights of the new chunk. During this phase the new chunk's weights are adjusted as described in Section 2.4, and it enters into the matching process. If the Bayesian weight method finds this chunk effective and the new chunk is successful at winning the competition, it is fully incorporated into the concept description, or *inaugurated.* Typically, this involves removing the contributing elements from the concept description and placing the new chunk in their place.

The basis for the competition between the newly proposed element and its constituents are the weights of the latter. Since the weights are a representation of the importance of each element's participation in describing the concept, they are a natural basis for assessing the

Table 4.4: Thresholds for competition between new and component elements.

BOOLEAN OPERATOR	THRESHOLD	INAUGURATE	PRUNE
Conjunction	$T = \min\{LN_i, LN_j\}$	$< T$	$\not< T$
Disjunction	$T = \max\{LS_i, LS_j\}$	$> T$	$\not> T$
Negation	$T = 1/LN_i$	$< T$	$\not< T$
	$T = 1/LS_i$	$> T$	$\not> T$

relevance of new chunks. As I described in Section 4.2.2, the weights indicate different types of relevance: The LS weight represents sufficiency while the LN weight represents necessity. Selecting an appropriate weight is based on these considerations. For instance, given a new conjunctive element, it is guaranteed to be false at least as often as its components; the Boolean conjunction operator makes the new chunk more specific. However, the new chunk may be false too often, and the operator may have over-specialized. In this case, the new chunk will fail to correctly expect some positive examples, and it will not be a necessary chunk for the concept. Thus the necessity weight LN serves as the source of a threshold for the new chunk. When a new conjunction is proposed, a threshold based on the minimum of the LN weights of each of the component elements is assigned. If the new chunk exceeds this threshold, it will be inaugurated.

By similar reasoning, for a new disjunctive element the LS weights of the component elements are used as a threshold. This guards against the introduction of overly general chunks. An overly general chunk will not be sufficient and will have a poor LS weight. The threshold for a new disjunctive chunk is the maximum of the LS weights of its two components.

The Boolean negation operator may add a new chunk that is either more or less general than its predecessor. Therefore, it is not possible to assess whether applying the operator has resulted in an overly specific or overly general element. The threshold assigned to a new negated element is simply the mathematical inverse of its component element's electing weight. For instance, if an element was elected as an argument for negation by its high LN weight, the resulting negated chunk would have to exceed 1/LN to be inaugurated as part of the concept description. Table 4.4 summarizes the selection of competition weights.

4.3.2 Pruning poor chunks

Knowing when to inaugurate a new chunk is clear: when it has surpassed its threshold. Knowing how long to wait for a chunk to surpass its threshold is not as clear. How long will it take for the Bayesian weigh learning method to determine if this chunk is useful? This potential for discarding a newly formed chunk is diagramed in Figure 4.1 by Box 2.

One approach might be to wait until a prespecified number of examples have been processed. If the new chunk was still not past its threshold, it could be confidently pruned. This simple strategy has a fundamental drawback: The appropriate number of examples is a domain-dependent quantity and is not directly available from the environment.

A second approach might be to examine the derivative of the new chunk's threshold weight. A small derivative would indicate that the element's weight was approaching an asymptote, and it could be confidently pruned. One property of this 'patience' criterion is that a new chunk with a strong weight but making poor progress would be pruned sooner than one with a weak weight that was making good progress. An earlier version of STAGGER used this technique to restrict the expansion of the search frontier (Schlimmer & Granger, 1986a).

This approach does not require the domain-dependent 'patience' parameter the first approach did. However, it does rely on a parameter to determine the minimum allowable slope. In a complex domain with a low probability of encountering examples of the concept, the threshold for a new chunk might be quite high, and yet the chunk would be unable to progress quickly. In this situation, new chunks with small slopes should be allowed to continue on toward their threshold. Conversely, in a domain with a low complexity and an even mixture of positive and negative examples, even a spurious new chunk would be able to maintain a small degree of weight change. The allowable slope value should be high in this case.

The final approach I discuss here attempts to tie both the 'patience' and slope values to functions of the input. Instead of waiting a prespecified number of trials before pruning, the new chunk is allowed to continue as long as its weights are changing, i.e., as long as it appears that the new chunk is coming up in the estimation of the Bayesian weight method. Instead of pruning an element with a weight slope below a certain value, the difference between the maximum and minimum of previous weights is examined. There is still a cutoff value (if the difference between the maximum and the minimum values is less than 0.1, for example), but the number of previous values over which this test is made varies as a function of the domain.

Consider the effect on pruning if only the most recent LS and LN weights are examined. The new chunk would have to change by the cutoff value after each example in order to avoid being pruned. For instance, if the window of previous weights to examine is lengthened to 10, the new chunk must exhibit a change equal to the cutoff value within the last 10 examples. In general, shortening the number of previous values examined enacts a more strict pruning mechanism, while lengthening this amount instantiates more patient pruning. The specific number of previous weight values that are examined by the pruning process is a function of the probability of a positive example. If this probability is low, new chunks may not be able to progress as quickly to their threshold, so the number of examined values is lengthened. Conversely, if examples and nonexamples are evenly mixed, new chunks are able to progress quickly, so STAGGER shortens the number of saved values. Figure 4.3 portrays the number of

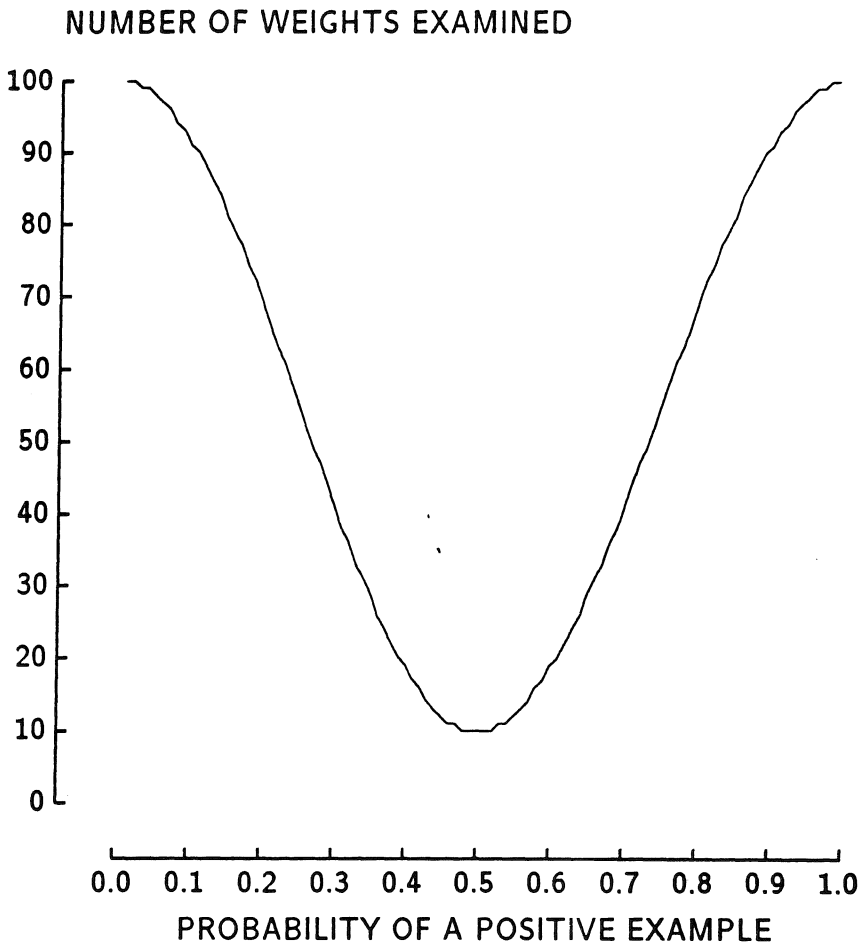

Figure 4.3: Number of weights examined to estimate slope for new chunks.

previous weight values examined as a function of the probability of a positive example.

The 'U' shape of this curve indicates that a great deal of patience is extended to chunks of concepts that have either a very low or very high probability. In domains where examples are evenly mixed, the probability of a positive example is 1/2 and considerably less patience is demonstrated. Intuitively, adjusting the number of previous weights examined in this manner allows the pruning mechanism to consider the performance of the new chunks in both positive and negative examples.

This competition mechanism limits the concept description size by pruning ineffective advances of the search frontier. Those chunks that are not useful to the Bayesian weight learning method are discarded. In the next section, I describe another way to limit the size of the concept description: pruning redundant or sloppy elements as the search frontier advances past them.

4.3.3 Inaugurating good new chunks

Given that a new concept description chunk has not yet been pruned, and its weight now exceeds the threshold assigned, it is inaugurated into the concept description. Figure 4.1 outlines this as the transition from Box 1 to Box 3 for simple chunks and from Box 6 to Box 7 for complex ones. Not much changes for the new chunk; its weights continue to be adjusted, and they in turn continue to influence matching. However, in order to limit the size of the concept description, complex chunk's constituents are deleted (Box 7). This pruning cleans up the receding edge of the search frontier in a manner complementary to the clean up of the advancing edge.

A simple heuristic is layered on top of this strategy: Do not delete elements from the concept description if STAGGER is effectively classifying new examples. So, instead of always deleting the receding frontier, previously effective elements are more or less likely to be pruned when their proposal is inaugurated: more likely if STAGGER is doing a poor job of classification and less likely if STAGGER is doing well. Figure 4.4 displays the specific probabilistic function used. The important qualities of the function shown in Figure 4.4 are that it is inversely proportional to the level of classification success and that it is a smooth function.

There is one added proviso to the cleanup of the backside of the search frontier: the first elements, corresponding to single attribute-value pairs, are not pruned. This is necessary property of the system rather than a heuristic one, for there are some relatively simple concepts that may not be learned if the first elements are pruned. For instance, a concept like (**small** $\wedge$ **red**) $\vee$ (**small** $\vee$ **circle**) cannot be learned if the element **small** is pruned after the first disjunct is inaugurated because it would be unavailable for formation of the second one.

Given that the search frontier is tentatively advanced in an ambitious manner, carefully established, and cleaned up behind, there is little opportunity for mistakes. However, it may still be the case that noisy examples or an initially skewed sequence of examples can lead search down the garden path. In the following section, I describe a mechanism that allows STAGGER to recover from inappropriate search.

4.4 Backtracking

The search process I describe above is a beam search. The best nodes in the search frontier are opportunistically expanded following matching errors and a number of highly-ranked nodes are retained for possible further expansion. The type of error indicates an appropriate search operator and thus an appropriate Boolean operator to apply. Error and operator type collectively suggest potential nodes for expansion. A few of these sites are chosen for advancement by the Bayesian element weights serving as an evaluation function. If the search frontier advances too far, either as a result of an unusual distribution of the examples or because

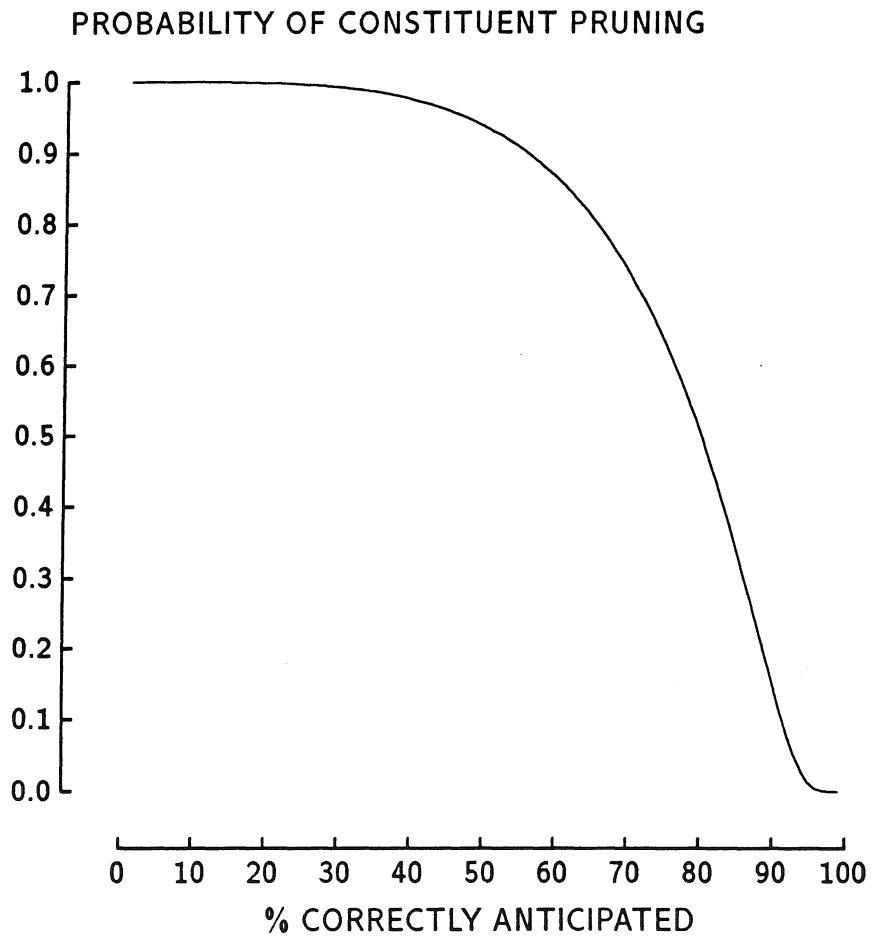

Figure 4.4: Probabilistic component pruning following inauguration.

of an error on STAGGER's part, backtracking may be useful to retreat from changes in the concept description. Figure 4.1 diagrammatically depicts this possibility with the transition from Box 3 to Box 4.

In this section, I describe a backtracking mechanism specifically designed for STAGGER's search through the space of Boolean chunks. First, I argue that though backtracking is unnecessary, it is useful and inexpensive. Then I describe a form of chronological backtracking employed in previous versions of STAGGER. In the third subsection, I describe a heuristic backtracking mechanism which results in more efficient retraction of poor chunks and reflects the search heuristics presented in Section 4.2.

4.4.1 The value of backtracking

Depth-first search illustrates one type of need for backtracking. At each choice point in the search, this method makes a single choice and moves ahead; each path is completely explored before starting on the next one. When dead-end is reached, search must be restarted because one of the choices was incorrectly made. This is commonly done by reseting search at the most recent choice point and following a new choice. This is called backtracking, for a mechanism other than the existing search operators is invoked to undo the search. This is like what people do when they reach dead-ends in paper mazes; they back up to the most recent intersection in the maze and try another branch.

Breadth-first search illustrates a second need for backtracking. This technique explores each path with equal speed; none of the paths are completed before the others. While it does not need to backtrack in order to explore alternative choices, if the optimal point in the search space lies in the middle of a path rather than at one end, this method may overshoot and have to recover by backtracking. One way to backtrack would be to revert search one step back on all paths.

The Boolean search in STAGGER does not need to backtrack for either of these reasons. First, because this search explores a number of paths simultaneously, it does not need to backtrack to consider alternate choices. Second, though the optimal chunks may lie in the middle of the search space, and STAGGER may overshoot them by forming overly specific or overly general chunks, the search operators provide the ability to reverse the direction of search. The operators of generalization and specialization are inverses of each other, and thus there is no need for an additional mechanism to undo the effects of search. For instance, if a concept description element has become overgeneralized, applying the specialization operator may correct the situation. These properties of the search in STAGGER eliminate the need for a backtracking mechanism.

While backtracking is not necessary, it may be a useful mechanism. Specifically, the notion of reversing the direction of search does not work as smoothly with the Boolean conjunction

and disjunction operators as might be expected. In order to undo the effects of conjoining **small** $\wedge$ **red** and **red** $\wedge$ **circle**, the Boolean disjunction operator must be able to recombine one of them with the resulting chunk **small** $\wedge$ **red** $\wedge$ **circle**. But, as I described in Section 4.3, those chunks may have been pruned in order to conserve computational resources. Without backtracking, a series of disjunctions may be required in order to recover the pruned chunks, yet the chunks that allow smooth reversal of search may not be available. Thus the search has the undesirable property that it is easier to advance the search frontier from a point than it is to revert the search back to that point.

Backtracking may be done inexpensively and within the search framework already established. The additional elements introduced by backtracking are easily added to the concept description. Similarly, the pruning mechanisms described in Section 4.3 are easily extended to control the load induced by backtracking as I describe below.

4.4.2 Chronological backtracking

The term chronological backtracking conveys the notion that search is rewound to an earlier temporal state. Previous backtracking mechanisms employed in STAGGER did just this: an ineffective chunk was replaced with its formative elements (Schlimmer & Granger, 1986a). For a simple chunk there would be no effect because its components are two unpruned initial elements. However, larger, ailing chunks are based on two intermediate, stepping-stone chunks, and these latter were pruned during inauguration.

Specifically, when a new chunk was introduced into the concept description, it was given a threshold based on the weight which elected it. For instance, if LN was used to select the component elements for a new conjunction, then the new chunk was given a threshold based on the most significant LN value of either of the component elements: the lowest LN value of the two. If the new chunk survived the pruning process (described in Section 4.3) long enough to achieve a LN weight in excess of this threshold, it became an established part of the concept description. Following the political simile, the element became inaugurated as part of the concept description: The component elements were removed from the concept description and placed on a backtracking list attached to the new chunk. The weights of the component elements were no longer updated, they were not part of the matching process, and they could not become arguments for the Boolean operators. Eliminating these elements improves both the efficiency of the concept description (by reducing the complexity of matching and chunking) and its clarity (by unifying the concept description).

Later, if the added chunk's LN weight became worse than its threshold, the component elements were retrieved from the backtracking list. Completing the simile with the political process, the ailing chunk was *impeached* by its weight electorate. The original components were reinstantiated as tentative members of the concept description much like the ailing chunk

Table 4.5: Chronological backtracking.

BOOLEAN OPERATOR	THRESHOLD	INAUGURATE	IMPEACH
Conjunction	$T = \min\{LN_i, LN_j\}$	$< T$	$> T$
Disjunction	$T = \max\{LS_i, LS_j\}$	$> T$	$< T$
Negation	$T = 1/LN_i$	$< T$	$> T$
	$T = 1/LS_i$	$> T$	$< T$

originally was. Table 4.5 summarizes this simple method of unwinding search paths. The first three columns are reproduced here from Table 4.4 for clarity.

Backtracking from an ailing element to its immediate predecessors adds modest space requirements. For each newly proposed chunk, its two component elements must be retained. This adds at most $2n$ to the space (where n is the number of concept description elements). Furthermore, the actuation of backtracking can add at most these $2n$ elements to the concept description; each of the two components can be reintroduced in the worst case. These additional space costs are linearly bounded in the size of the existing concept description.

This method had the advantage that it brought symmetry to the search process: Getting back from a point in the search space was now just as easy as getting to it. However, this strategy is relatively information poor; it only examines the value of the search evaluation function at a single point in the distributed concept description. Following the symmetry suggested by the use of the election heuristic in proposing new chunks, backtracking could utilize information in the weights of a large number of concept description elements to recover from poor ones. In the next subsection, I describe that method.

4.4.3 Heuristic backtracking

There at least three shortcomings to chronological backtracking, each of which results in unnecessary work (de Kleer, 1984). First, search is reverted to the most recent choice point, and this may not be far enough to get past the dead-end. This leads to the second problem: search is rewound past choice points according to a temporal ordering rather than according to a heuristic one; valid search choice points may be undone in order to undo invalid ones. Third, because of this temporal ordering of choice points, the forward search process may be forced to reexplore contradictory choices repeatedly until the backtracking resets search at a deep enough choice point. STAGGER uses a simple form of belief revision and an associated dependency-directed backtracking mechanism to overcome the first two shortcomings. Although it would be possible to overcome the third one as well, I will argue in the following paragraphs that it is undesirable to do so in a general learning system.

Belief revision

Belief revision systems can serve a supervisory role for search-based deductive systems. The underlying deduction mechanism operates uninterrupted until a contradiction in the database is discovered. At this point the belief revision system is invoked, and it takes items out of the database in an attempt to resolve the contradiction. This process of removal is sometimes simply referred to as making items *out*.

A typical example of such a system is one that schedules an executive meeting. First, a date for the meeting is assumed, say Wednesday. Then a time is selected. Assumptions like these in turn lead to deductions regarding the selection of a room, for example. Unless the system has been exceptionally fortunate, one of the assumptions will lead to a contradiction. For instance, the meeting is on Wednesday at 2 PM (assumption), yet the meeting cannot be on Wednesday because one of the attendees is out of town on this day. The belief revision mechanism examines each of the assumptions and their effect on the state of the meeting's plan. Usually some sort of backtracking is used to rewind search to a point beyond one of the assumptions, so that the forward reasoning process of the meeting scheduler can proceed.

If used, chronological backtracking would return to the most recent assumption. In the meeting scheduler example, even though if it was determined that one of the attendees could not attend on Wednesday, search would first rewind back to the assumption about the time of day, go forward to discover the conflict again, and rewind back to the time of day assumption until each possible time was exhausted. By contrast, a mechanism that examines the dependencies inherent in the conflict could avoid thrashing between search and backtracking by resetting search at the assumption that led to the contradiction in the first place. This is what *dependency-directed* backtracking does. In the meeting example, search would be restarted with a new day for the meeting. Decisions about time and location would be preserved.

Dependency-directed backtracking

It is just this sort of mechanism that STAGGER uses to overcome two of the limitations of chronological backtracking listed above: failure to reset search far enough back and unnecessarily repeating search through valid, worthwhile paths. With the chronological backtracking mechanism, STAGGER would have placed the component elements on a backup list attached to a newly inaugurated chunk. If the chunk ever violated its inaugural threshold, it would have been impeached, and search would have rewound to include the ancestors. Instead of this information-poor strategy, STAGGER uses a heuristic backtracking mechanism.

As before, when a new chunk is proposed, it is given a threshold based on the electing weight. (E.g., a new conjunction has a LN threshold equal to the minimum of the LN weights of the two combined elements.) In belief revision terms this is a *hypothesis*: the LN weight of the new chunk will exceed the threshold. If the new chunk surpasses its threshold, it is

inaugurated, and the hypothesis becomes a *belief*: the LN weight of the new chunk exceeds the threshold. The component elements are pruned and forgotten; they are not placed on a backtracking list. If the new chunk's electing weight falls below the inauguration threshold, it is impeached (transitions from Box 3 to Box 4 in Figure 4.1). In belief revision terms, this is a *contradiction*: the LN weight of the new chunk does not exceed the threshold. To recover from this inconsistency, a dependency-directed backtracking mechanism is invoked.

The nature of dependency-directed backtracking requires that the premises upon which each belief is based can be easily examined. This is because when backtracking is invoked, it will examine the validity of each premise and return search to a consistent state, one in which all of the premises are valid. Therefore, when a new chunk is inaugurated, its premises are collected into a list and attached to it. If the two arguments to the chunking operator are chunks they may be pruned (see Section 4.3.3), but if they are initial, single attribute-value elements, they are retained to allow later comparison. The premise list consists of the initial elements present in the new chunk and their weight at the time of inauguration. For instance, the premises for the chunk small $\vee$ red $\vee$ circle depicted in Figure 4.2 are:

1. The LS value of small is greater than T_{small},
2. The LS value of red is greater than T_{red}, and
3. The LS value of circle is greater than T_{circle}.

Heuristic backtracking examines the truth of each of the premises upon which the contradictory chunk is based. Each premise is easily assessed by comparing the current weights of each premise element with its corresponding saved value. If below the recorded threshold, the premise is a contradiction too, and the pattern for each of these invalid premises is syntactically removed from the ailing chunk. The result is proposed as a new element and competes as its predecessor did before. The transition from Box 4 to Box 5 in Figure 4.1 denotes the possibility that the resulting chunk may supplant the contradictory chunk. The net effect of backtracking is to reset search at a state in which each of the premises are valid, i.e., to a new chunk which omits the invalid ancestor premises and includes the valid ones. This state may not have been previously visited, but it is guaranteed to be closer to initial states since ailing chunks are simplified.

Comparison to chronological backtracking

This dependency-directed method of backtracking is less space efficient than the chronological backtracking method discussed in Section 4.4.2. For each newly proposed chunk, a pointer and a threshold for each of the included primary elements must be retained for at most n^2 pointers (where n is the size of the concept description). Yet dependency-directed backtracking

introduces fewer new chunks in the worst case: Each backtrack results in only one additional new chunk for a total of n.

There are two additional functional differences between the heuristic and chronological backtracking methods. First, the dependency-directed backtracking process does not necessarily reset search at a previously explored state. In the chronological backtracking system, this was always the case, for search was rewound to the most recent state. However, removing the faulty premises from a contradictory chunk may result in a completely new (and as yet unexplored) element. This is a reflection of the role of backtracking as a general inverse operator. In the chronological version, backtracking serves as an inverse for each of the Boolean operator applications by explicitly storing the previous state. This is backtracking in the sense that it moves search to a previous temporal state. Since the search operators move forward from simple states toward more complex states, the dependency-directed mechanism performs backtracking in the sense that it always moves search to a simpler state.

Second, the heuristic backtracking mechanism is an information-rich strategy as compared with the chronological version. Instead of relying solely on the competition mechanism and a receding series of inauguration and impeachment in order to rewind search, the heuristic version examines the weights of each of the included primary elements and may reset search directly at a better location.

However, both of the backtracking mechanisms share the same triggering mechanism: impeachment. Similarly, they both utilize the same type of competition to test the results of backtracking. In chronological backtracking, the ailing chunk's ancestors are tentatively reintroduced into the concept description; with dependency-directed backtracking, a refined chunk is introduced. In both cases the new chunks must surpass a weight threshold based on the ailing chunk's performance in order to supplant their predecessor.

General dependency-directed backtracking

Several of the general benefits of using dependency-directed backtracking are realized in STAGGER. For instance, Stallman and Sussman's (1977) EL system exhibited the property that small changes in the modeled system resulted in only a small number of changes in the database. This property is also true of STAGGER as I will describe in Section 6.3, since it is able to maximally reuse previously learned concepts if the environment changes.

Another property of general belief revision systems is that they can recover from the effects of pruning the database. Items placed *out* of the database (to resolve a contradiction) may represent the results of substantial problem solving. Therefore, if they are now consistent with the database, they can be reinstantiated (or *unouted*), saving the work they represent. In EL, if a set of beliefs are contradicted and outed from the database, subsequent contradictions may lead to the reinstallment of their premises. If this happens, the beliefs are restored wholesale

to the database without repeating the search used to derive them.

Unouting occurs in STAGGER, though in a diminished form and as an emergent phenomenon. If an ailing chunk is the subject of backtracking, it may not be pruned. Even if the new belief was confirmed (inaugurated), the ailing predecessor might not be pruned if performance is sufficiently high (see Section 4.3.3). Any change in the situation that favors the previously contradictory element could be accommodated, since the chunk would still be in the concept description, ready for the Bayesian weighting method.

One particular benefit is not realized in STAGGER, nor is it appropriate that it should be. This is the third shortcoming of chronological backtracking, reexploring contradictory choices, which EL overcomes by maintaining explicit records of inconsistencies. Following each contradiction resolution, a *nogood* structure is created that records the cause of the conflict and ensures that the same combination of choices is never tried again. Adding nogoods assumes that contradictory information is persistent. This is an unwelcome assumption to make when designing a general learning mechanism, since a highly skewed set of examples or a noisy domain could both lead to apparent contradictions.

4.5 Overview

The Bayesian weight learning method is not sufficient for all concept learning tasks, so in this chapter I formulate an additional method to assisting it. This new method primarily constructs representational chunks of existing concept description elements and is called the Boolean chunk learning method because the new elements are composed using conjunction, disjunction, and negation. Figure 4.5 is a completed schematic of a chunk's lifecycle, and it summarizes the overall operation of the chunking method.

Three layers of heuristics are used to identify potentially effective new chunks (transition into Box 1). First, chunks are added after STAGGER makes an expectation error. If the error indicates that the overall description is too general, a more specific, conjunctive chunk is added. Conversely, if the description appears to be too specific, a more admitting, disjunctive chunk is called for. The components of these chunks are selected by identifying necessary and sufficient concept description elements.

The resulting chunks are not immediately introduced into the concept description. Rather, they are subjected to a period of testing after which they may either be inaugurated as full members of the description or they may be eliminated altogether. In the former case, the chunk must garner a weight in excess of its assigned threshold. If chunking is successful in this respect, it is inaugurated (right into Box 3). However, some chunks do not meet their threshold, and these are simply discarded (down into Box 2). This process of proposing new chunks, testing them, and inaugurating them, is iterated until the chunks are sufficiently

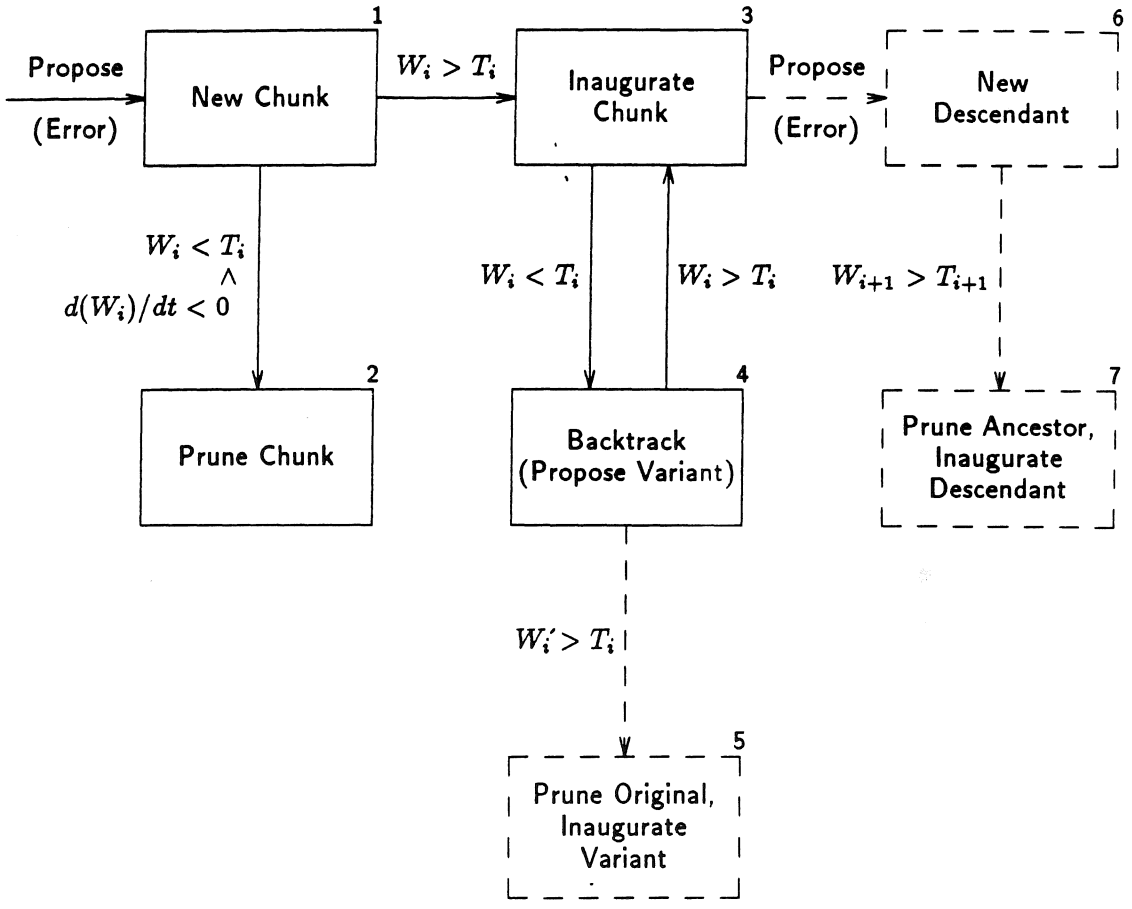

Figure 4.5: Full schematic of a chunk's lifecycle.

complex for effective concept identification (into Boxes 6 and 7). Recall that when a complex chunk is inaugurated, its two constituent elements may be discarded (down into Box 7) unlike simple chunks whose two components are initial elements that are not pruned (Box 3).

If a once-successful chunk falls below its inaugurating threshold, it is impeached and dependency-directed backtracking is triggered (down into Box 4). This process generates a consistent variant of the ailing chunk by examining the predictive contribution of each of the embedded sub-elements. The resulting element is tentative placed into the concept description, and it competes as the faltering chunk did previously. If the variant surpasses the threshold originally used to inaugurate the faltering chunk, then the latter is discarded in favor of the newer element (down into Box 5). Alternatively, if the chunk regains strong weights and resurpasses its threshold, backtracking is canceled (back to Box 3) though any proposed variants are allowed to die-out or succeed as they may.

These processes may also be viewed as beam search through the space of possible chunks. In this framework, the concept description is a set of states in which the proposal of new chunks advances the search frontier, inauguration establishes it, pruning of ineffective chunks retracts hasty advancement, and pruning of component elements following inauguration cleans up the rear of the frontier.

Returning to the original motivation for the Boolean chunk learning method, the Bayesian weight method is incapable of representing and acquiring some concepts. The Boolean chunking method serves to restructure the representation language for this simple method by forming higher level features or chunks. In assisting the weight method, the chunk learning method is effectively changing the substrate from which Bayesian weight method proceeds, allowing it to effectively search a larger space of possible concepts.

The next chapter illustrates these processes and their implications for the Bayesian weight learning method through a series of examples after the explanatory manner of Chapter 3 for Chapter 2.

Chapter 5

Examples of Boolean Chunk Learning

5.1 Introduction

In the previous chapter, I introduced the Boolean chunking method that allows STAGGER to assign predictive values to combinations of features. These processes of proposing new combinations, testing them, pruning ineffective ones, and backtracking from ailing to predictive elements are somewhat complicated. The motivation for marrying them with the weight learning method is simple: the combination of the two methods is capable of acquiring a larger class of concept types than either alone. The framework for demonstrating this will be a revisitation of the set of 19 concept types first introduced in Chapter 3. Regrettably, the necessity for the complexity of the methods is not as easily motivated. Therefore, in this chapter I first illustrate the functioning of each of the processes introduced in Chapter 4 through a series of examples, following the organization of Chapter 3. The detail of each demonstration sets the stage for discussing the purpose of the processes developed in the previous chapter.

To provide a clearer understanding of the functioning of Boolean chunk learning, reconsider the lifecycle of a typical chunk schematically depicted in Figure 5.1 (a duplication of Figure 4.5). Beginning near the upper left of the figure (Box 1), a chunk is initially formed following an expectation error. A performance threshold T is associated with the chunk's weight W at this time; in order to avoid pruning, the chunk must garner a weight stronger than T. Moving downward to Box 2, if the weight remains low and appears to be leveling off, the chunk is pruned. Alternatively, moving right to Box 3, if the chunk meets this threshold, it is inaugurated into the concept description.

Moving down to Box 4, if at some later time the chunk's weight W falls below its inaugurating threshold T, backtracking is invoked to propose a variant of the ailing chunk. Further down at Box 5, if the variant chunk surpasses the threshold T, then it replaces the original, ailing chunk. On the other hand, if an inaugurated chunk regarners a strong weight $W > T$, then backtracking is canceled (back to Box 3).

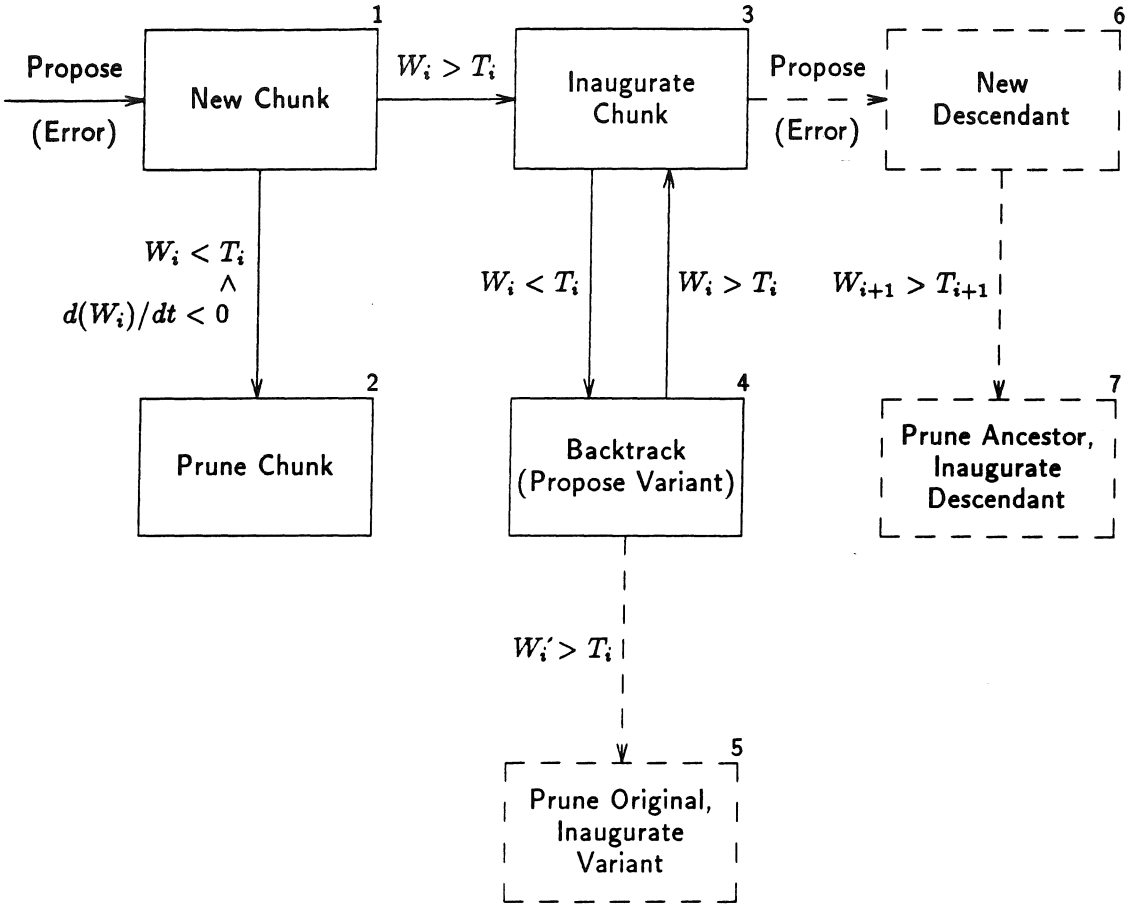

Figure 5.1: Schematic of a chunk's lifecycle.

At the rightmost of the figure (Box 6), the cycle of proposing new chunks is completed, for an inaugurated chunk, if effective, may serve as a component for a new super-chunk following another expectation error. Down at Box 7, if the descendant, super-chunk passes its threshold T_i, then the ancestral, component chunk is pruned.

The first illustration of these processes in action is the simple one seen first in Chapter 3, and the point of repeating it is to clarify the generation, testing, pruning, and recovery of new Boolean chunks.

Turning to a pair of naturally constrained tasks, I apply the combined methods to the task of acquiring an accurate description of edible mushrooms. Though this task involves some subjective assessment of attribute-values, and many of the characteristics of the examples are subject to variability, STAGGER is able to form an effective concept description.

In the second naturally constrained task, STAGGER constructs a description of a Democratic in terms of individual voting records from the House of Representatives. Two interesting characteristics of this task are: (1) nearly 1/2 of the examples contain one or more unknown values as many Congressmen do not vote on all legislation, and (2) many of the examples are contradictory since Congressmen frequently vote against their party line. Despite these characteristics, STAGGER is able to construct an effective concept description.

Lastly, as in Chapter 3, I examine performance in light of a comprehensive set of tasks. In this case the Bayesian weighting method is assisted by the Boolean chunking, and the results are improved. Some of the representational shortcomings of the weight learning method seen clearly at the end Chapter 3 are shown to be overcome through the addition of Boolean chunking.

5.2 A simple classification task revisited

Let us again consider a simple domain of objects describable in terms of their size, color, and shape. As before, STAGGER's initial concept description consists of an element for each attribute-value pair, each assigned unbiased LS and LN weights as displayed in Figure 5.2 (A duplication of Figure 3.1).

5.2.1 Adding new chunks

Assuming the same sequence of examples, the Boolean chunking method does not come into play until the second object: a medium, red triangle. As the matching equation instantiated in Section 3.2 indicates, STAGGER incorrectly expects that this object is a positive example. This amounts to a predictive error of commission and is an indication that the current concept description is too general. To remedy the situation, a new, more restrictive chunk is added.

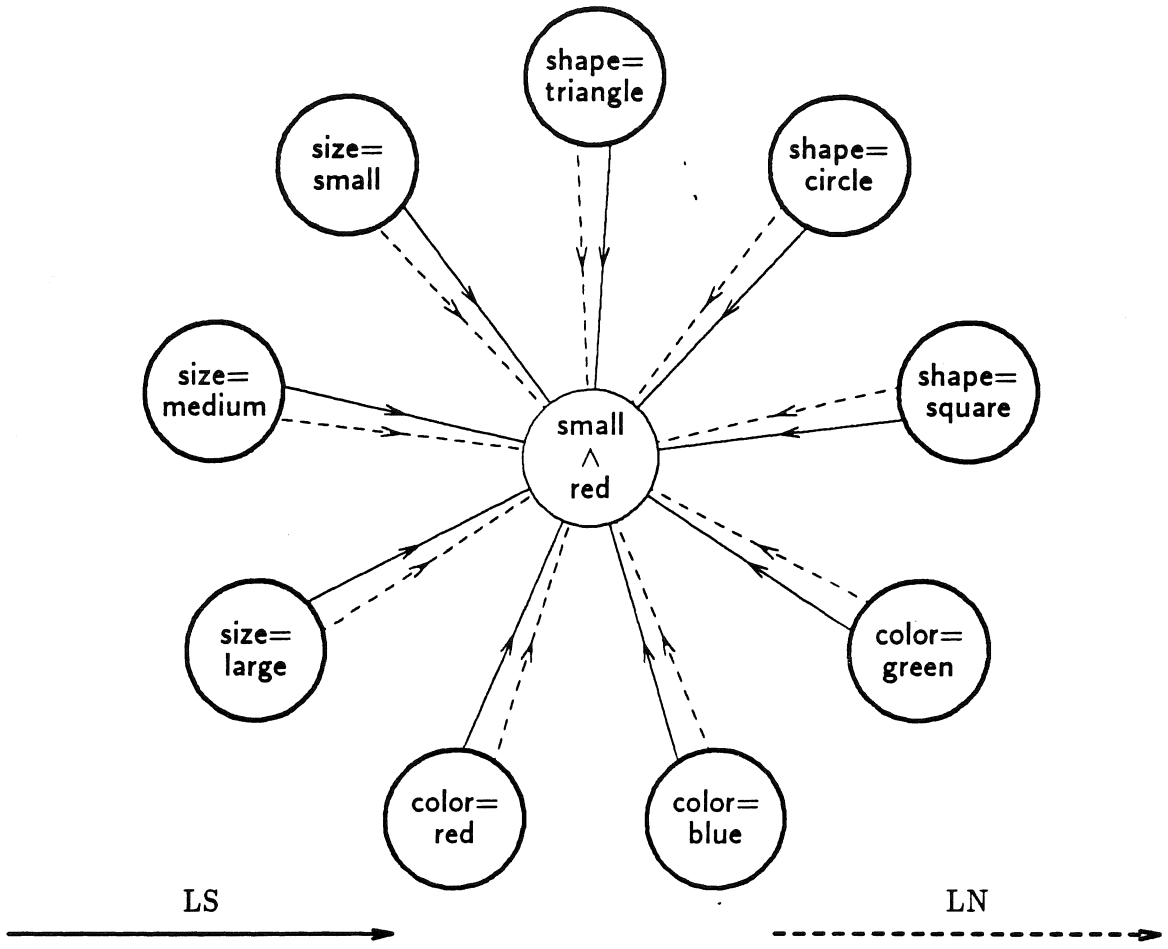

Figure 5.2: Initial concept elements.

Table 5.1: Matches between medium, red triangle and concept description.

MATCHED			UNMATCHED		
PATTERNS	LS	LN	PATTERNS	LS	LN
color = red	1.00	1.00	size = small	2.11	0.47
			shape = circle	2.11	0.47
size = medium	0.47	2.11	size = large	1.00	1.00
			color = green	1.00	1.00
shape = triangle	0.47	2.11	color = blue	1.00	1.00
			shape = square	1.00	1.00

Following the heuristics listed in Tables 4.1, 4.2, and 4.3, first the elements are separated into two groups via a nomination process: those elements that match this object and those that do not. Table 5.1 lists the results of this partitioning using the elements as listed in Table 3.3.

The nomination heuristics specify that a new conjunction is to be formed from a matched element (one that misled matching to succeed) and an unmatched element (an important one that was missing). These heuristics do not limit the possibilities much, as is clearly indicated by Table 5.1, for there are $3 \times 6 = 18$ possible conjunctions between the matched and unmatched elements. Thus the second heuristics (Table 4.3) elect the best elements to serve as arguments for the specializing (AND) operator. Simply, the most predictive elements are combined, or those with the strongest weight. Because they are to be formed into a new AND, the elements with the most significant (much less than one) necessity weights are elected; this orders the 18 possible new conjunctions. Table 5.1 also reflects these heuristics, for within the columns, the elements are sorted according to decreasing LN significance. STAGGER proceeds to examine each of the possibilities until it finds a conjunction that is not already in the concept description. In this case the new chunk small $\wedge$ red is added since it has the best LN weights and expands the representation.[1] Figure 5.3 shows the resulting concept description.

The nomination and election heuristics also indicate that this error of predictive commission is an opportunity to invert a poorly weighted element. As Table 4.2 indicates, a matched element is inverted (it should not match a nonexample), and Table 4.3 specifies, an element with poor LS or LN weights is chosen. The shortage of experience results in two equivalent options for negation: size $\neq$ medium or shape $\neq$ triangle. Assume that the latter is added. As with the conjunction operator, if this chunk were already in the concept description, the next worst element would be inverted.

[1]Indecision between adding small $\wedge$ red and red $\wedge$ circle arises because the weights are based on a few examples. Further experience imposes a more precise ordering.

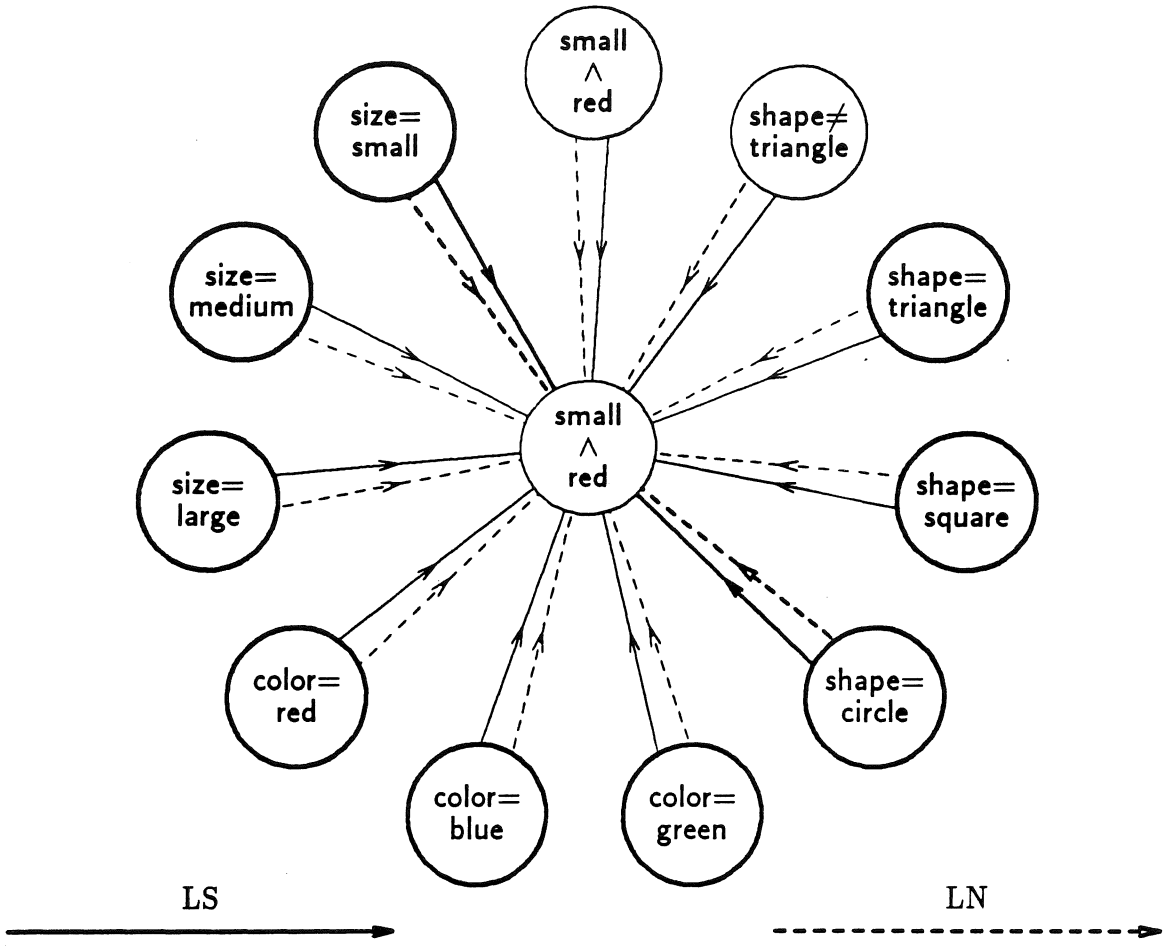

Figure 5.3: Extended concept description.

Table 5.2: Concept description after processing 9 examples.

PATTERNS	COUNTS				WEIGHTS	
	C_P	I_N	I_P	C_N	LS	LN
color $\neq$ green	1.00	1.00	1.00	1.00	1.00	1.00
blue $\vee$ (small $\wedge$ red)	1.00	1.00	1.00	1.00	1.00	1.00
shape $\neq$ triangle	1.99	6.98	0.99	0.99	0.76	2.67
small $\wedge$ red	1.99	0.99	0.99	6.98	5.35	0.38
size = large	0.89	3.89	2.89	4.88	0.53	1.37
size = medium	0.89	3.88	2.89	4.89	0.53	1.37
size = small	2.89	1.89	0.89	6.88	3.54	0.30
color = blue	0.89	0.89	2.89	7.88	2.32	0.85
color = green	0.89	3.89	2.89	4.88	0.53	1.37
color = red	2.89	4.88	0.89	3.89	1.37	0.53
shape = circle	1.89	2.89	1.89	5.88	1.51	0.75
shape = square	1.89	4.88	1.89	3.89	0.90	1.13
shape = triangle	0.89	1.89	2.89	6.88	1.10	0.97

5.2.2 Testing new chunks

The newly added conjunction and negation are introduced into the concept description on a trial basis, casting the formation heuristics of nomination and election into the role of generator and the pruning heuristics of Section 4.3 as tester. This latter process of eliminating ineffective chunks is based on a threshold for performance established at the time the new chunk is added. For the new chunk small $\wedge$ red, this threshold is set to the minimum of the electing, necessity weights of the elements for small and red, or $T_{LN} = \min\{0.47, 1.00\}$. For the new AND to remain in the concept description, it must garner a stronger necessity weight than its predecessors. The new NOT, similarly, must outperform the element inverted by gaining a LS weight in excess of the inverse of shape = triangle: $T_{LS} = 1/0.47 = 2.13$.

These elements are given a fair chance to succeed, for they are allowed to remain in the concept description as long as they are making progress toward their threshold. Progress is measured by examining the slope over a number of previous weight values. If chunk weight appears to be asymptoting to a value less than its threshold, and the corresponding changes become small, then it is pruned. After processing the next 7 examples, the new chunk small $\wedge$ red has surpassed its weight threshold; its current LN weight is more significant (smaller than) its threshold: $0.38 < 0.47 = T_{LN}$. See Table 5.2.

However, the new NOT is not past its threshold of $T_{LS} = 2.13$, and unless the concept changes, it will not ever be. Its lack of improvement, as attested by small changes in its LN weight, is grounds for pruning.

Table 5.3: Backtracking by examining embedded attribute-value pairs.

	LN		
	THEN	NOW	VALID?
small ∧ (blue ∨ red)	0.31	0.68	×
⇓			
size = small	0.31	0.30	√
color = blue	0.90	0.62	×
color = red	0.51	0.35	√
⇓			
small ∧ red	0.68	0.35	√

5.2.3 Backtracking from poor chunks

This generate and test process is not always completely effective at limiting the introduction of overly-general or overly-specific chunks. In fact, these may slip into the concept representation as a by-product of noisy or skewed examples. Typically these chunks have only moderately influential weights, but if a number are allowed to clutter the concept description, matching is adversely affected. As a remedy, backtracking is employed to clean up the concept description whenever the weight of a once-successful element falls below the threshold that established it.

For instance, assume that the chunk small ∧ ¬ green had been overzealously inaugurated but was now failing to meet its threshold of $T_{\text{LN}} = 0.31$. A straightforward recovery technique would be to simply rewind the search to its previous state (e.g., small and ¬ green). However, this information-poor strategy suffers from the maladies of chronological backtracking. A more intelligent approach utilizes the individual predictiveness of the embedded elements in order to side-step from this poor chunk directly to an effective one. Similar to dependency-directed backtracking, those sub-elements that are not contributing to the predictiveness of the ailing chunk are syntactically removed. Contribution is assessed by comparing each single attribute-value element's current weights with those at the time the chunk was established. Table 5.3 lists the embedded initial elements and their weights for this ailing chunk. Note that the negation of the element color = green is implemented as color = blue ∨ red.

This faltering chunk was most recently the result of an AND that used the necessity (LN) weight to elect the conjuncts. Thus, the subparts are examined on the basis this weight. As Table 5.3 illustrates, both the small and red elements are doing at least as well when the chunk was inaugurated. However, the blue element is doing poorly. A new chunk is formed by removing all ailing subparts (blue), leaving the pattern small ∧ red. This new chunk is assigned the original chunk's threshold, or $T_{\text{LN}} = 0.31$. If the new chunk meets this necessity requirement, then the ailing chunk is pruned. In fact, because small ∧ red is already an effective part of the concept description, small ∧ ¬green is immediately pruned. In the case that all

of the sub-elements are valid, no backtracking is done; whereas if none of the sub-elements are valid, the faltering chunk is simply pruned. The net result of this intelligent backtracking mechanism is that a poor chunk is simplified directly into an effective one, side-stepping several iterations of the propose, test, and prune cycle.

Note that this backtracking method utilizes the initial, single attribute-value elements to dissect poor chunks. This is in conflict with the memory limiting process since the latter seeks to prune component chunks whenever a newly effective chunk is inaugurated into the concept description. As a compromise between the two, component chunks are pruned unless they are initial elements. This is why when the chunk **small** $\wedge$ **red** was inaugurated, the single attribute-value elements **small** and **red** were not pruned.

5.2.4 Retained chunks

Figure 5.4 lists the concept description after a large number of examples have been processed. The thick solid and dashed lines radiating from the **small** $\wedge$ **red** chunk indicate that it is both sufficient and necessary for concept expectation. This is an unrealistic situation, as the following two sections will demonstrate. It is atypical that a single chunk will be completely predictive. Note that in addition to the initial, single attribute-value elements and **small** $\wedge$ **red**, there are chunks for **color** $\neq$ **blue** and **color** $\neq$ **green**. The patterns represented by these latter two chunks are necessary for concept expectation, as their strong LN weights, thick dashed lines, indicate. Furthermore, their inclusion in the concept description is a type of redundancy. Boolean chunking is a beam search and will tend to discover a number of effective chunks. Other likely redundant chunks for this task are **size** $\neq$ **medium** and **size** $\neq$ **large**.

In summary, the Boolean chunk formation method forms new, descriptive chunks by coalescing existing elements using AND, OR, and NOT. Though the chunks are introduced into the concept description through a loose beam search, the processes that test them ensure that only the highly predictive ones remain.

5.2.5 Cooperative interactions

Abstractly, the chunking method seeks to reduce the task confronting the Bayesian weighting method to a trivial one. By adding new chunks, the Boolean chunk learning method is rewriting examples in terms of new, higher-level features. This has the effect of expanding the space of concepts learnable by the weighting method. In the limit, with the appropriate chunks present, the weight method's task is reduced from determining the exact pattern of weights for a number of elements (complex) to weighting a single chunk (simple). This property of condensing the representation, or moving from a distributed to a unified descriptive form, is somewhat masked by the need to maintain the original attribute-value elements for the sake

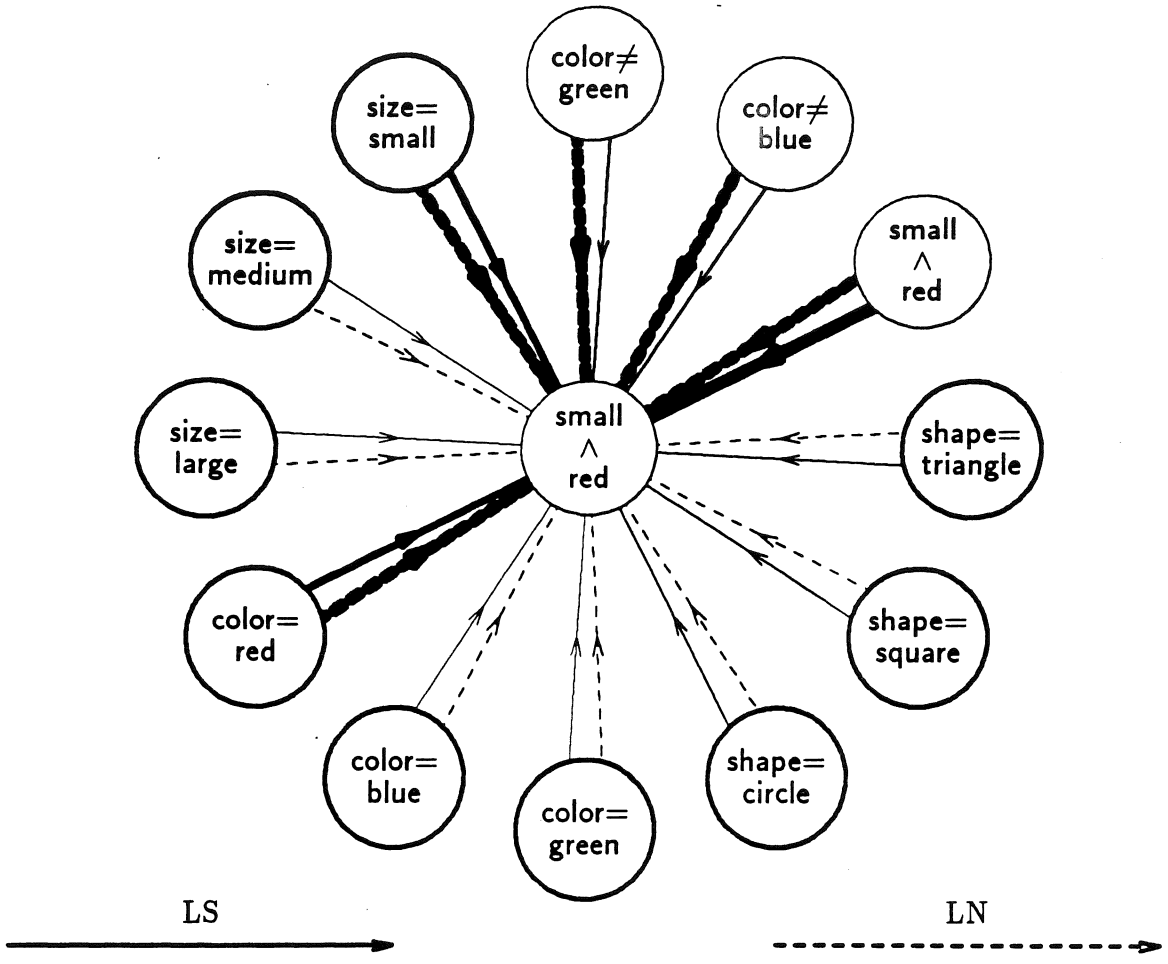

Figure 5.4: Typical asymptotic concept description.

of the backtracking mechanism. Nevertheless, as the following sections illustrate, the bulk of classification comes to reside in a few chunks as acquisition proceeds over time.

5.3 Distinguishing edible from poisonous mushrooms

In the previous section I used a simple task to describe the cooperation between the Bayesian weighting and Boolean chunk learning methods employed in STAGGER. However, the difficulty with a simple, artificially constructed task is that interesting characteristics may be simplified away – perhaps those that would prohibit the methods from effective operation. By way of evidence for the generality of STAGGER, in this section I describe the program's acquisition of a naturally constrained concept: an edible (as opposed to poisonous) mushroom.

5.3.1 The task

Specifically, the task is to assess the edibility of a novel mushroom sample. The available inputs are a series of 3,078 mushroom samples representing 23 species of gilled mushrooms in the *Agaricus* and *Lepiota* family. Each sample is described in terms of the 23 observable attributes (listed in Table 5.4) and identified as edible or poisonous by *The Audubon Society Field Guide to North American Mushrooms* (1981, pp. 500–525).[2] Simple arithmetic indicates that there are approximately 10^{15} (the product of the number of possible values for each attribute) possible mushroom samples in terms of these attribute-values, yet there are only $3,078/10^{15} \ll 1\%$ of the samples actually observed. This is one distinguishing feature between a naturally constrained task and an artificial one.

The *Guide* clearly states that this is not a hobby to be taken lightly and emphasizes that there is no simple rule for mushrooms like the rule "leaflets three, let it be" for Poisonous Oak and Ivy. Determining edibility is complicated because as a mushroom matures, many of its attributes may change. For instance, when the mushroom first erupts, its cap may have a conical shape. As it matures, the cap flattens out and may even become concave. Second, some of the attributes listed in Table 5.4 may only be determined subjectively; what one person terms a pink shade, another may call orange.

This task is at once both easier and harder than it seems. For instance, an informal survey reveals that most people suspect that odor is a highly predictive feature for mushroom edibility. This is the type of feature that an explanation-based or analytic learning system might hypothesize since it is plausible that there is a causal connection between smell and edibility. Smell is predictive for this domain, a fact STAGGER readily discovers. All of the

[2]The data used for this demonstration is too large to be included as an appendix but is available by writing to the author.

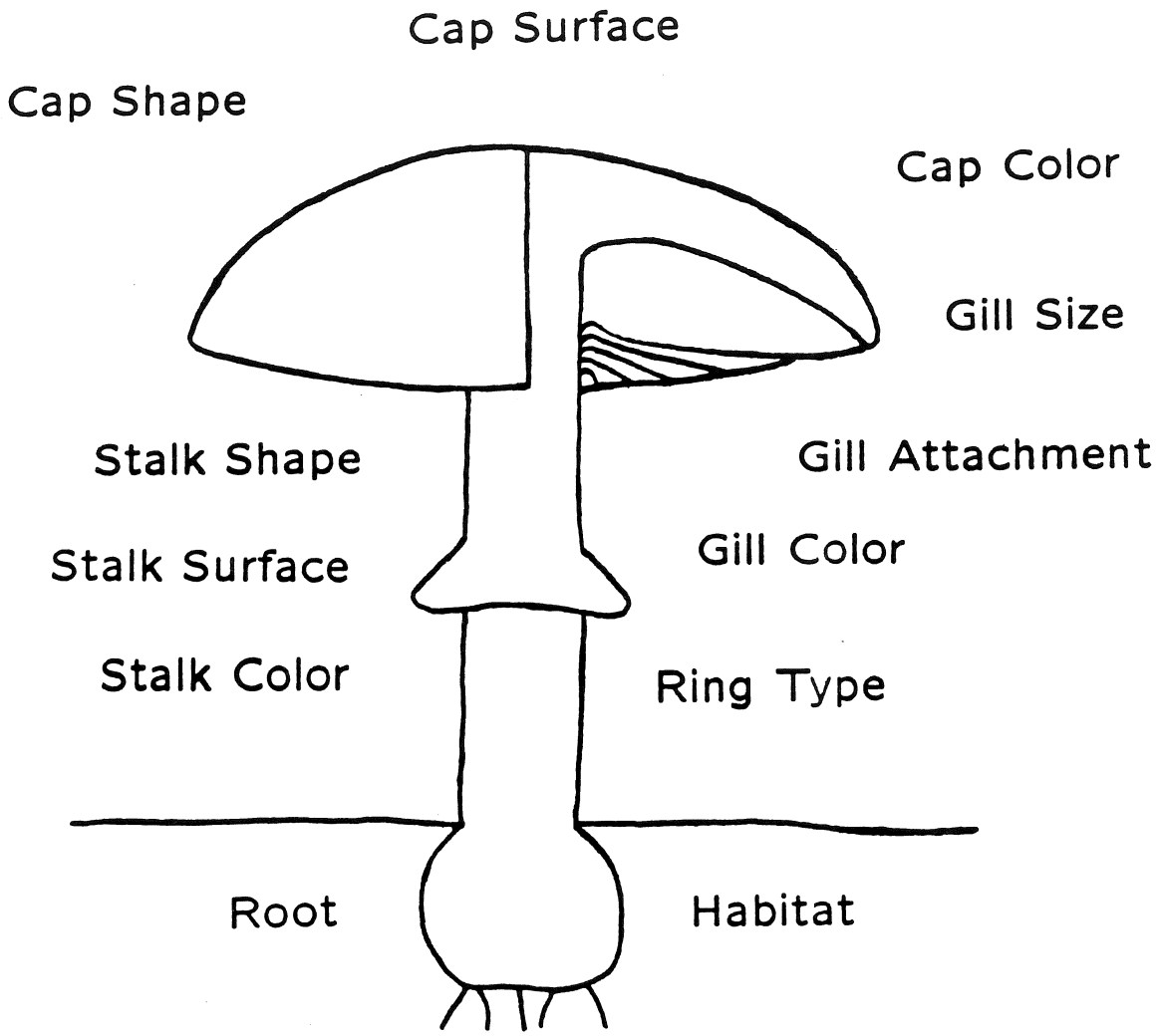

Figure 5.5: Identifying edible mushrooms.

Table 5.4: Attributes and their values for mushroom tasks.

ATTRIBUTE		VALUES
cap-shape	$\in$	{bell, conical, convex, flat, knobbed, sunken }
cap-surface	$\in$	{fibrous, grooves, scaly, smooth }
cap-color	$\in$	{brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow }
bruises?	$\in$	{bruises, no }
odor	$\in$	{almond, anise, creosote, fishy, foul, musty, none, pungent, spicy }
gill-attachment	$\in$	{attached, descending, free, notched }
gill-spacing	$\in$	{close, crowded, distant }
gill-size	$\in$	{broad, narrow }
gill-color	$\in$	{black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow }
stalk-shape	$\in$	{enlarging, tapering }
stalk-root	$\in$	{bulbous, club, cup, equal, rhizomorphs, rooted}
stalk-surface-above-ring	$\in$	{fibrous, scaly, silky, smooth }
stalk-surface-below-ring	$\in$	{fibrous, scaly, silky, smooth }
stalk-color-above-ring	$\in$	{brown, buff, cinnamon, gray, orange, pink, red, white, yellow }
stalk-color-below-ring	$\in$	{brown, buff, cinnamon, gray, orange, pink, red, white, yellow }
veil-type	$\in$	{partial, universal }
veil-color	$\in$	{brown, orange, white, yellow }
ring-number	$\in$	{none, one, two }
ring-type	$\in$	{cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone }
spore-print-color	$\in$	{black, brown, buff, chocolate, green, orange, purple, white, yellow }
population	$\in$	{abundant, clustered, numerous, scattered, several, solitary }
habitat	$\in$	{grasses, leaves, meadows, paths, urban, waste, woods }

Table 5.5: Similarity of edible *Lepiota rachodes* and poisonous *Chlorophyllum molybdites*.

FEATURES	SAMPLES	
	Lep. rach.	*Chl. moly.*
cap-shape	flat	flat
cap-surface	smooth	smooth
cap-color	pink	pink
bruises?	bruises	bruises
odor	none	none
gill-attachment	free	free
gill-spacing	close	close
gill-size	broad	broad
gill-color	white	white
stalk-shape	enlarging	enlarging
stalk-root	bulbous	bulbous
stalk-surface-above-ring	smooth	smooth
stalk-surface-below-ring	smooth	smooth
stalk-color-above-ring	white	white
stalk-color-below-ring	white	white
veil-type	partial	partial
veil-color	white	white
ring-number	two	two
ring-type	pendant	pendant
spore-print-color	white	green
population	several	several
habitat	paths	meadows

mushrooms that smell pleasant are edible, and those that smell unpleasant are poisonous. Regrettably smell is not enough; the majority of the mushrooms in this study, including both edible and poisonous species, have no smell. If a learner wants to maximize performance, it will have to select an additional basis for discrimination.

These unscented mushrooms may be quite close to each other along other attribute dimensions. For instance, Table 5.5 lists the attribute-values for two mushroom samples, one edible and one poisonous. Note that these two mushrooms are highly similar, complicating the task of differentiating between the two and advising heed to the *Guide*'s warning that mushroom hunting in the wild is not a sport for the faint-hearted.

5.3.2 STAGGER's performance

Despite these subtle forms of noisiness, STAGGER is able to form an effective concept description as Figure 5.6 depicts. Note that after processing as little as 1% (300 samples) of the data,

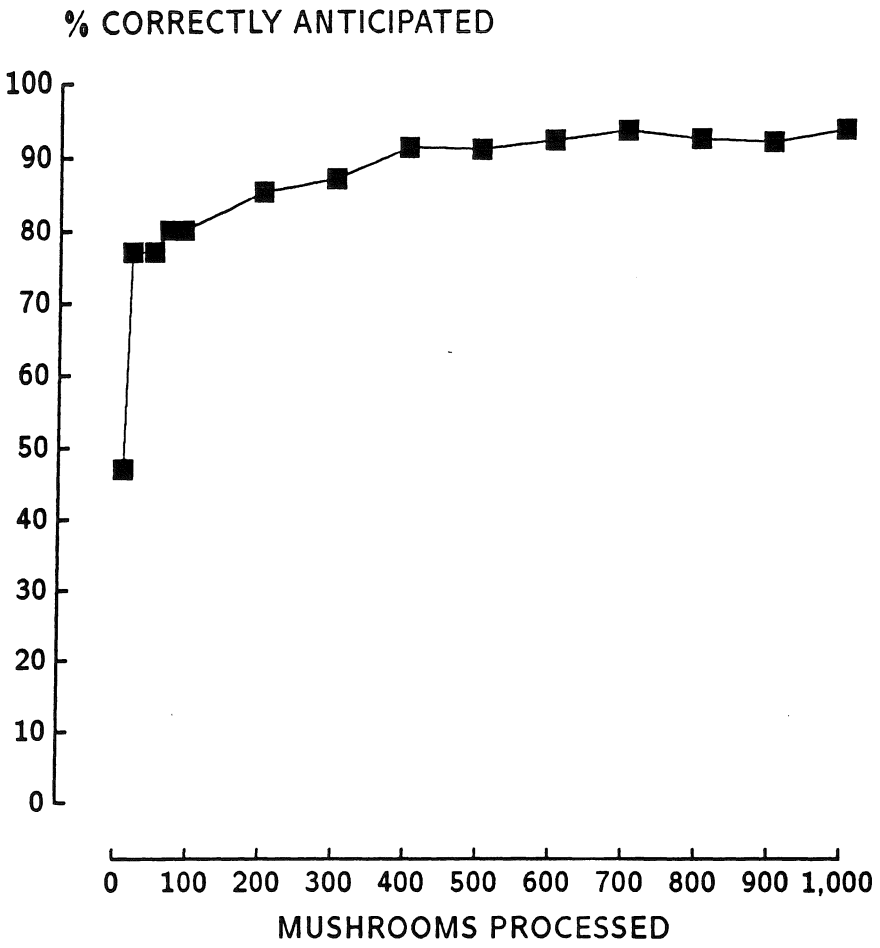

Figure 5.6: Averaged acquisition of edible mushroom concept.

STAGGER is correctly estimating the edibility of 8 of 10 previously unseen mushrooms. This performance slowly rises to approximately 95% correct.[3]

5.3.3 The chunks formed by STAGGER

Though the size of an intermediate concept description prohibits complete enumeration, it may be useful to examine the chunks that have been inaugurated after processing 1,000 mushroom samples. Table 5.6 lists the 33 inaugurated chunks in the order that they were formed.

There are two items to note about the chunks in Table 5.6. First, there is a large degree of overlap between the chunks; many of them share the same disjuncts. This testifies to the validity of the characterization of Boolean chunking as a beam search. Furthermore, several

[3]Each point in Figure 5.6 is an average over 10 executions and represents a running total of the number of mushrooms correctly predicted over the last 100. This is a type of continuous cross-validation and avoids the difficulties associated with identifying a separate training and test set.

Table 5.6: Thirty-three chunks after processing 1,000 mushrooms (part 1).

#	CHUNKS	LS	LN
1.	cap-color ≠ buff	1.03	0.03
2.	population ≠ clustered	1.25	0.34
3.	spore-print-color ≠ chocolate	1.15	0.39
4.	stalk-surface-above-ring ≠ silky	1.56	0.11
5.	odor ≠ foul	1.30	0.00
6.	spore-print-color ≠ black ∧ population = solitary	0.76	5.26
7.	stalk-surface-below-ring = (fibrous ∨ smooth)	1.57	0.40
8.	stalk-surface-below-ring ≠ silky	1.14	0.32
9.	(ring-type = flaring) ∨ (odor = none ∧ population = solitary) ∨ (odor = none ∧ spore-print-color = brown)	281.31	0.53
10.	stalk-surface-below-ring ≠ scaly	1.26	0.46
11.	(population = numerous) ∨ (odor = none ∧ population = solitary) ∨ (odor = none ∧ gill-color = pink)	238.84	0.60
12.	stalk-color-below-ring ≠ yellow	1.12	0.01
13.	odor ≠ creosote	1.08	0.02
14.	cap-color ≠ yellow	1.10	0.50
15.	ring-type ≠ large	1.12	0.01
16.	odor ≠ musty	1.09	0.02
17.	cap-color = (gray ∨ green ∨ purple)	3.00	0.80
18.	bruises? = bruises ∨ cap-shape = flat	1.06	0.90
19.	bruises? = bruises ∨ cap-shape = (flat ∨ sunken)	1.09	0.86
20.	odor = (almond ∨ anise)	107.51	0.78

Table 5.6b: Thirty-three typical, inaugurated chunks after processing 1,000 mushroom samples (part 2).

#	CHUNKS	LS	LN
21.	(spore-print-color ≠ brown ∧ population ≠ solitary) ∨ (stalk-surface-below-ring = (scaly ∨ silky))	0.84	80.08
22.	odor ≠ pungent	1.12	0.02
23.	(habitat = waste) ∨ (odor = almond) ∨ (odor = none ∧ population = solitary ∧ stalk-surface-below-ring = (fibrous ∨ smooth)) ∨ (odor = none ∧ spore-print-color = brown ∧ stalk-surface-below-ring = (fibrous ∨ smooth)) ∨ (cap-color = brown ∧ odor = none ∧ stalk-surface-below-ring = (fibrous ∨ smooth))	319.98	0.24
24.	stalk-color-above-ring ≠ cinnamon	1.08	0.03
25.	(population = numerous) ∨ (odor = none ∧ population = solitary) ∨ (odor = none ∧ spore-print-color = brown) ∨ (cap-color = brown ∧ odor = none ∧ stalk-surface-below-ring = (fibrous ∨ smooth))	251.61	0.33
26.	stalk-color-below-ring ≠ cinnamon	1.08	0.03
27.	population = (abundant ∨ numerous)	46.55	0.88
28.	stalk-root ≠ club	1.14	0.52
29.	spore-print-color ≠ green	1.13	0.02
30.	gill-color ≠ buff	1.11	0.03
31.	gill-color = (black ∨ brown ∨ purple)	2.43	0.87
32.	gill-color = (black ∨ brown ∨ purple ∨ red)	2.59	0.86
33.	gill-color = (chocolate ∨ purple)	1.67	0.94

of the chunks are completely subsumed by others. For instance, Chunk 18 is a subchunk of Chunk 19. This redundancy arises for three reasons: (1) subchunks may be used to form more than one new chunk, (2) chunks may be rederived by the beam search up from initial elements, and (3) not all chunks are pruned as new chunks are inaugurated. This latter factor arises as classification performance does, because as Figure 4.4 indicates, if classification performance is high, STAGGER adopts a "don't fix it if it works" policy and does not prune components as the search frontier advances.

Second, there is a rather large disparity between the strengths of the weights of various chunks in Table 5.6. For instance, Chunk 28 has relatively weak sufficiency and necessity weights whereas Chunk 27 has a strong sufficiency weight and Chunk 29 has a strong necessity weight (much less than one). Furthermore, there are more complex chunks that have strong sufficiency and necessity weights (e.g., Chunk 23). These discrepancies arise among inaugurated chunks because their original establishing thresholds were based on the quality of their predecessors. The chunks with relatively poor weights are descendent from even poorer elements; those with strong weights may have come from strong ancestors. By definition of the search processes, the complex chunks must be highly predictive, for they are descendent from a series of chunks with increasing weights.

Though the 33 chunks in Table 5.6 seem numerous, they represent a relatively small expansion in the concept description size. The memory limiting mechanisms of Section 4.3 serve STAGGER well in this respect. Given that the initial concept description size for this domain is 125 (one for each attribute-value pair), the additional 33 inaugurated chunks represent a small growth of 26%. The average growth rate over the ten executions summarized in Figure 5.6 is 33%. To get a better estimate of the effectiveness of the proposal heuristics, we can calculate the number of opportunities for search expansion as compared with the number of chunks actually retained. On the average, STAGGER makes a running total of 70 expectation errors over the first 1,000 samples; this corresponds roughly to the introduction of $2 \times 70 = 140$ new chunks (one for either AND or OR and another for NOT). In comparison, there are 33 inaugurated chunks, some of which are complex and required the introduction of subchunks. Accounting for this, there were about 62 chunks that proved effective following their introduction, for a retention rate of about 44%.

There are also chunks cohabiting with the 125 initial chunks and the 33 listed above that are waiting to be either inaugurated or pruned. On the average these represent another 53% in growth for a total concept description size of about 183% of the number of attribute-value pairs.

Before closing this examination of mushroom concept learning, consider the semantics associated with the highly predictive chunks listed in Table 5.6. In general, those with a strong LS weight represent statements that are true of some edible mushrooms but are false of

Table 5.7: Correspondence of disjuncts to edible species in Chunk 23.

DISJUNCTS	MATCHING EDIBLE SPECIES
habitat = waste	*Lepiota americana*
odor = almond	*Agaricus abruptibulbus, Agaricus arvensis, Agaricus agustus*
odor = none ∧ population = solitary ∧ stalk-surface-below-ring = (fibrous ∨ smooth)	*Agaricus bitorquis, Agaricus haemorrhoidarius, Agaricus subrutilescens, Lepiota rachodes*
odor = none ∧ spore-print-color = brown ∧ stalk-surface-below-ring = (fibrous ∨ smooth)	*Agaricus bitorquis, Agaricus campestris, Agaricus haemorrhoidarius*
odor = none ∧ cap-color = brown ∧ stalk-surface-below-ring = (fibrous ∨ smooth)	*Agaricus bitorquis, Agaricus campestris, Agaricus haemorrhoidarius, Lepiota acutesquamosa, Lepiota americana, Phaeolepiota aurea*

all poisonous ones. Intuitively, knowing the truth of one of these chunks is enough to conclude that the mushroom is edible. Formally, this can be seen by recalling that the definition of the sufficiency weight is positive implication, or *matched* ⇒ *example*. The contrapositive of this statement implies that ¬*example* ⇒ ¬*matched*, or that these chunks should not be true of any poisonous mushrooms. This analysis is borne out empirically. For instance, Chunk 23 matches 11 edible species (all but 2 of the edible species), yet it does not match any of the 10 poisonous species. Closer examination of each of the disjuncts reveals that they correspond to groups of species as Table 5.7 indicates.

The disjuncts are somewhat redundant, as this *post-hoc* analysis reveals. The fourth disjunct, though syntactically different from the fifth, is semantically subsumed by it. If STAGGER were to host an additional method to simplifying effective chunks, that method might operate at the level of disjuncts and be able to detect this simplification.

Closing our semantic analysis of the chunks formed for this task, note that in general chunks with strong LN weights represent statements that are true of all edible mushrooms but are only false of some poisonous ones. Again, following our intuition, knowing that a necessary chunk is unmatched is strong evidence for the unedibility of the sample. Formally, this is because the necessity weight is defined as the negative implication, or ¬*matched* ⇒ ¬*example*. This is the case with Chunk 5 which has a necessity weight of 1/201, for it explicitly excludes 2

poisonous species. However, it is not generally the case that complexity is required in a chunk in order to achieve a strong sufficiency as opposed to a strong necessity weight.

5.3.4 Conclusions

Concept learning about mushrooms involves examining examples with a large number of attribute-values and building complex chunks. The data requires the ability to split fine hairs and may not be amenable to a causally-driven learning method. However, for the most part, all of the attribute-values were specified; none of the mushroom samples were under-specified – for instance, an unknown spore print color. This quality of the data could prove detrimental as could additional levels of noisiness. To assess the effects of these properties, in the next section I describe STAGGER's performance at acquiring a description of a Democratic Representative in terms of individual voting records: a domain with a high percentage of underspecified examples.

5.4 Forming political concepts

Acquiring a description of an edible mushroom in terms of individual mushroom features requires examining a large number of attribute values and constructing complex chunks as I discussed in the previous section. However, that demonstration did not address the issue of unknown values. Therefore, in this section I describe a domain in which nearly half of the examples have at least one attribute-value unspecified, and for which many of the examples are contradictory. After examining some of the characteristics of this task, I present some empirical evidence of STAGGER's effectiveness, illustrate matching with unknown values, and examine the constructed chunks.

5.4.1 The task

Let us consider the task of predicting whether a U.S. House of Representatives Congressman is a Democrat or a Republican in terms of their individual voting record on a number of pieces of legislation. For this task, the *Congressional Quarterly Almanac* (1985) was consulted which identifies 16 key votes, presents a short analysis of each, and lists for each Congressman their party affiliation and disposition on the key issues.[4] Table 5.8 lists these key votes.

[4]For each Congressman, the *CQA* lists dispositions toward issues as either voted for, paired for, announced for, voted against, paired against, announced against, voted present, voted present to avoid conflicts of interest, and did not make any position known. The first three indications were simplified to a yea vote and the second three to a nay vote. Only the latter three indicates were taken as an unknown disposition. Because the data was taken directly from the *CQA*, it is not reproduced here.

Table 5.8: Key votes used for congressional voting domain.

1984 HOUSE KEY VOTES
1. Handicapped Infants
2. Water Project Cost Sharing
3. Adoption of the Budget Resolution
4. Physician Fee Freeze
5. El Salvador Aid
6. Religious Groups in Schools
7. Anti-Satellite Test Ban
8. Aid to Nicaraguan 'Contras'
9. MX Missile
10. Immigration
11. Synfuels Corporation Cutback
12. Education Spending
13. Superfund Right to Sue
14. Crime Bill
15. Duty-Free Exports
16. Export Admin. Act / South Africa

Table 5.9: Mostly unknown voting record.

REP.	VOTING RECORD															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Hanson (ID)	N	?	?	?	?	?	?	?	?	?	?	?	?	Y	?	?

The formulation of the task is straightforward. The key votes serve as the attributes, with the individual votes as values, and each Congressman as an example of his/her party.[5] Since there are only two political parties represented in the House, STAGGER constructs one for the Democrats and denotes the Republicans as the opposite. The results in the remainder of this section are essentially the same if STAGGER constructs a concept description for the Republicans and predicts Democrats as the opposite.

Close examination of the data reveals that many of the examples are of poor quality. Nearly half (47%) of the Congressmen did not vote on one or more of these issues though only 6% of the votes are unspecified. As Table 5.9 indicates, the distribution of undervoting is not completely even.

Compounding matters further, the individual examples are somewhat contradictory. Many of the Congressmen do not vote strictly according to their party line, perhaps deferring to

[5]In passing, note that as with the other naturally constrained task examined in this chapter the number of possible examples ($2^{16} = 65,536$) far exceeds the number of observed examples (435).

Table 5.10: Nearly contradictory voting records among Democratic Congressmen.

REP.	VOTING RECORD															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Barnes (MD)	N	Y	Y	N	N	N	Y	Y	Y	Y	N	N	N	N	Y	Y
Daniel (VA)	N	N	N	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N
Bates (CA)	Y	Y	Y	N	N	?	Y	Y	Y	Y	N	N	N	N	Y	?
Ray (GA)	N	N	?	N	Y	Y	N	N	N	N	Y	Y	Y	Y	N	Y
Feighan (OH)	Y	N	N	N	N	N	Y	Y	Y	Y	N	N	N	N	Y	Y
Hall (TX)	N	Y	Y	Y	Y	Y	N	N	N	N	Y	Y	?	Y	N	N

the needs of their constituency. Table 5.10 lists a few pairs of Democratic Congressmen who voted opposingly on nearly every issue. Undoubtedly there are contradictory pairs among the Republican representation as well.[6]

5.4.2 STAGGER's performance

The initial concept description consists of 32 attribute-value elements, one for each vote on each issue. STAGGER weights and augments these elements and is able to correctly estimate the party of a Congressman 9 of 10 times after seeing 100 examples. Figure 5.7 shows the rise in classification performance over previously unseen examples. Asymptotic performance appears to be about 90% correct anticipation of example identity as can be seen by duplicating the data until 1,000 examples have been processed.[7]

5.4.3 Matching with unknown values

Several of the Congressmen failed to indicate a preference on some of these key issues, and thus matching must proceed on the basis of remaining, known votes. Elements that cannot be matched may not be used to adjust expectation of a Democrat, but they can serve to refine the range of expectation as Section 2.3 describes. Simply, the largest of the two element weights is multiplied into the upper bound on expectation, reflecting the maximal degree of expectation, and the smaller of the weights is included in the product of a lower bound on expectation. If the resulting maximal/minimal bounds are close, then expectation is tightly constrained

[6]The point of listing specific Congressmen is not to defame the individuals or institution, but rather to convince the reader that statements about the quality of the data are valid.

[7]Points in Figure 5.7 represent averages over 10 executions, and the individual data result from continuous cross-validation.

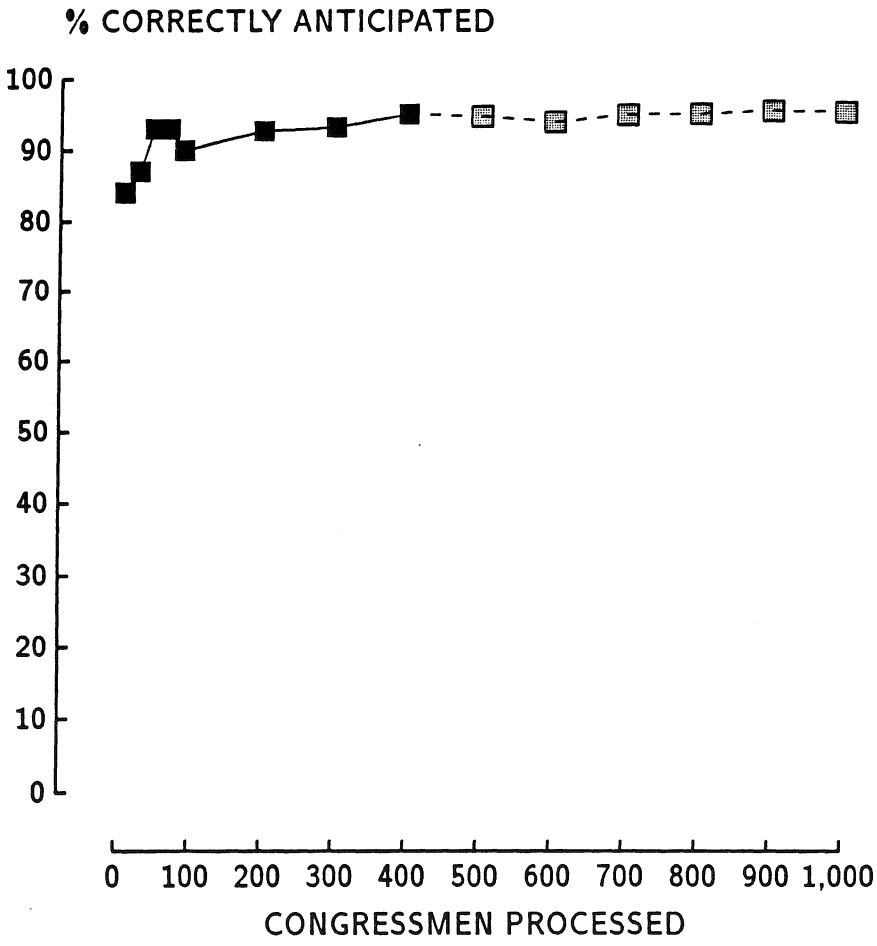

Figure 5.7: Acquisition of Democrat concept.

and there can be confidence in the resulting value. On the other hand, a wide gap between the upper and lower bounds on expectation indicates that the computed expectation is of poor quality. As an example of this, consider matching Congressman Hanson's voting record (listed in Table 5.9) with the concept description resulting after processing Barnes' voting record (listed in Table 5.10. After updating the appropriate counts, the concept description following processing Barnes is listed in Table 5.11.

Given Congressman Hanson's voting record, some of the elements in the concept description can be matched. In particular, note that on the two votes for which Hanson indicated a disposition, he agrees with Barnes on one and disagrees on the other. Given this sparse basis for deriving an expectation, our intuitive expectation would be to to base expectation on the prior odds in favor of a Democrat. After processing only one example these odds are estimated at $odds(Democrat) = 1.56$. In fact, this is about what STAGGER comes up with given that only 4 of the 32 elements in Table 5.11 can be matched. Table 5.12 lists these four elments and the results of applying the matching Equation 2.3 to their weights. The resulting expectation is close to the prior odds of a Democrat.

Because of the sparsity of values in Hanson's voting record, the other 28 chunks cannot be matched, and their weights cannot be used to aid in determining this Congressman's party affiliation. However, multiplying $odds(Democrat|Hanson)$ by the larger of each of the unmatchable element's weights yields an upper bound of $1.18 \times 1.36^{28} = 6,471$. Deriving a lower bound by taking the product with the smaller of each pair of weights yields an equally divergent quantity, $1.18 \times 0.64^{28} = 0.0000044$. Mapping these odds into probabilities we have $0.0000044 \leq p(Democrat|Hanson) \leq 0.99985$, though the best guess would be $p(Democrat|Hanson) = 0.54$. The large gap is a quantitative indication of the lack of confidence in this expectation, in part due to the inexperience of the concept description in Table 5.11, but largely due to the absence of most features.

In contrast to the lack of confidence arising from a sparse voting record, examining a nearly complete record indicates a basis for confidence expectation. For instance, consider matching Congressman Bates' voting record (Table 5.10) with the concept description in Table 5.11. In this case 28 of the elements can be matched and only 4 cannot. Table 5.13 lists these 28 elements and the result of computing Equation 2.3.

Adjusting a lower and upper bound on this expectation does not alter it much. Multiplying $odds(Democrat|Bates)$ by the larger weight for the four unmatchable elements increases the upper odds to 6,477; taking the product with the smaller of the four weights yields an odds of 317. In terms of probabilities this is a tight bound on expectation, $0.99686 \leq p(Democrat|Bates) \leq 0.99985$, indicating a nearly complete match and cause for confidence in the resulting odds of expectation.

Note that this inexperienced concept description does not discriminate between relevant

Table 5.11: Concept description after processing Congressman Barnes' record.

PATTERNS	COUNTS				WEIGHTS	
	C_P	I_N	I_P	C_N	LS	LN
handicapped = yea	0.90	0.90	1.90	0.90	0.64	1.36
handicapped = nay	1.90	0.90	0.90	0.90	1.36	0.64
water-project = yea	1.90	0.90	0.90	0.90	1.36	0.64
water-project = nay	0.90	0.90	1.90	0.90	0.64	1.36
budget-resolution = yea	1.90	0.90	0.90	0.90	1.36	0.64
budget-resolution = nay	0.90	0.90	1.90	0.90	0.64	1.36
physician-fee-freeze = yea	0.90	0.90	1.90	0.90	0.64	1.36
physician-fee-freeze = nay	1.90	0.90	0.90	0.90	1.36	0.64
el-salvador-aid = yea	0.90	0.90	1.90	0.90	0.64	1.36
el-salvador-aid = nay	1.90	0.90	0.90	0.90	1.36	0.64
religious-groups = yea	0.90	0.90	1.90	0.90	0.64	1.36
religious-groups = nay	1.90	0.90	0.90	0.90	1.36	0.64
anti-satellite = yea	1.90	0.90	0.90	0.90	1.36	0.64
anti-satellite = nay	0.90	0.90	1.90	0.90	0.64	1.36
nicaraguan-contras = yea	1.90	0.90	0.90	0.90	1.36	0.64
nicaraguan-contras = nay	0.90	0.90	1.90	0.90	0.64	1.36
mx-missile = yea	1.90	0.90	0.90	0.90	1.36	0.64
mx-missile = nay	0.90	0.90	1.90	0.90	0.64	1.36
immigration = yea	1.90	0.90	0.90	0.90	1.36	0.64
immigration = nay	0.90	0.90	1.90	0.90	0.64	1.36
synfuels-corp = yea	0.90	0.90	1.90	0.90	0.64	1.36
synfuels-corp = nay	1.90	0.90	0.90	0.90	1.36	0.64
education = yea	0.90	0.90	1.90	0.90	0.64	1.36
education = nay	1.90	0.90	0.90	0.90	1.36	0.64
superfund-sue = yea	0.90	0.90	1.90	0.90	0.64	1.36
superfund-sue = nay	1.90	0.90	0.90	0.90	1.36	0.64
crime = yea	0.90	0.90	1.90	0.90	0.64	1.36
crime = nay	1.90	0.90	0.90	0.90	1.36	0.64
duty-free-exports = yea	1.90	0.90	0.90	0.90	1.36	0.64
duty-free-exports = nay	0.90	0.90	1.90	0.90	0.64	1.36
south-africa-act = yea	1.90	0.90	0.90	0.90	1.36	0.64
south-africa-act = nay	0.90	0.90	1.90	0.90	0.64	1.36

Table 5.12: Matching to Congressman Hanson's record.

MATCHED PATTERNS	LS	UNMATCHED PATTERNS	LN
handicapped = nay	1.36	handicapped = yea	1.36
crime = yea	0.64	crime = nay	0.64
$\Pi =$	0.87	$=$	0.87
$odds(Democrat \mid Hanson) = 1.56 \times 0.87 \times 0.87 = 1.18$			

Table 5.13: Matching to Congressman Bates' record.

MATCHED PATTERNS	LS	UNMATCHED PATTERNS	LN
handicapped = yea	0.64	handicapped = nay	0.64
water-project = yea	1.36	water-project = nay	1.36
budget-resolution = yea	1.36	budget-resolution = nay	1.36
physician-fee-freeze = nay	1.36	physician-fee-freeze = yea	1.36
el-salvador-aid = nay	1.36	el-salvador-aid = yea	1.36
anti-satellite = yea	1.36	anti-satellite = nay	1.36
nicaraguan-contras = yea	1.36	nicaraguan-contras = nay	1.36
mx-missile = yea	1.36	mx-missile = nay	1.36
immigration = yea	1.36	immigration = nay	1.36
synfuels-corp = nay	1.36	synfuels-corp = yea	1.36
education = nay	1.36	education = yea	1.36
superfund-sue = nay	1.36	superfund-sue = yea	1.36
crime = nay	1.36	crime = yea	1.36
duty-free-exports = yea	1.36	duty-free-exports = nay	1.36
$\Pi =$	34.84	$=$	34.84
$odds(Democrat \mid Bates) = 1.56 \times 34.84 \times 34.84 = 1,893.57$			

features and irrelevant ones; bounds on expectation of Bates' party affiliation would be equally tight no matter which two votes he abstained from. It is unlikely that this is truly the case in general, and in this domain, a concept description with additional training yields a differential expectation bound depending on whether the missing features are irrelevant (tight bound) or relevant (large bound). If an irrelevant feature is unknown, the weights are close to one; using the larger does not increase the upper bound by much, nor does the smaller decrease the lower bound significantly.

Both of these examples were purposely illustrated with a poorly trained concept description to afford discussion of the distinction between confidence arising from completeness of match and confidence from accruing experience. In the former, expectation is confident because all of the salient features for prediction are measurable, and there is no need for uncertainty. In the latter case, accumulated experience clearly identifies the relevant features, thus eliminating the novice's uncertainty. The confidence quantitatively calculated by computing a lower and upper bound on expectation addresses the first issue, and to its credit, as experience refines the concept description driving the matching process, the bounds reflect the degree to which salient features are known. The second type of confidence is not represented as well in STAGGER, though in a minor way it is represented implicitly through the accumulation of the counts underlying the LS and LN weights.

5.4.4 The chunks formed by STAGGER

Because this domain involves somewhat fewer attribute-values, the size of the resulting concept description is amenable to graphical depiction. See Figure 5.8. Note that for the most part, only three of the 16 key votes exhibit a significant predictive weight (working counter-clockwise from the top): Physician's Fee Freeze (4), Education Spending (12), and Crime Bill (14). Second, the chunks are in many cases more predictive than their components. For instance, near the upper left of the figure, AST $\wedge$ CR (7. Anti-satellite test ban, and 14. crime bill) has a thicker solid line (stronger LS weight) than either of its component elements AST or CR. By forming these chunks the Boolean method is rewriting example features, so the Bayesian weighting method may easily capture concept regularity.

Following the presentation in the previous section, consider in detail a typical set of inaugurated chunks after processing 1,000 examples of Congressional Representatives (Table 5.14).

Two items are worthy of note with respect to these chunks. First, their relatively small LS and LN weights reflects the fact that examples in this domain are noisy. The classification difficulties arising from similarity between the poisonous and edible mushrooms seen in the previous section are exaggerated in this domain. As Table 5.10 illustrates, many of the Democrats do not vote according to their party's platform. Second, there are a number of

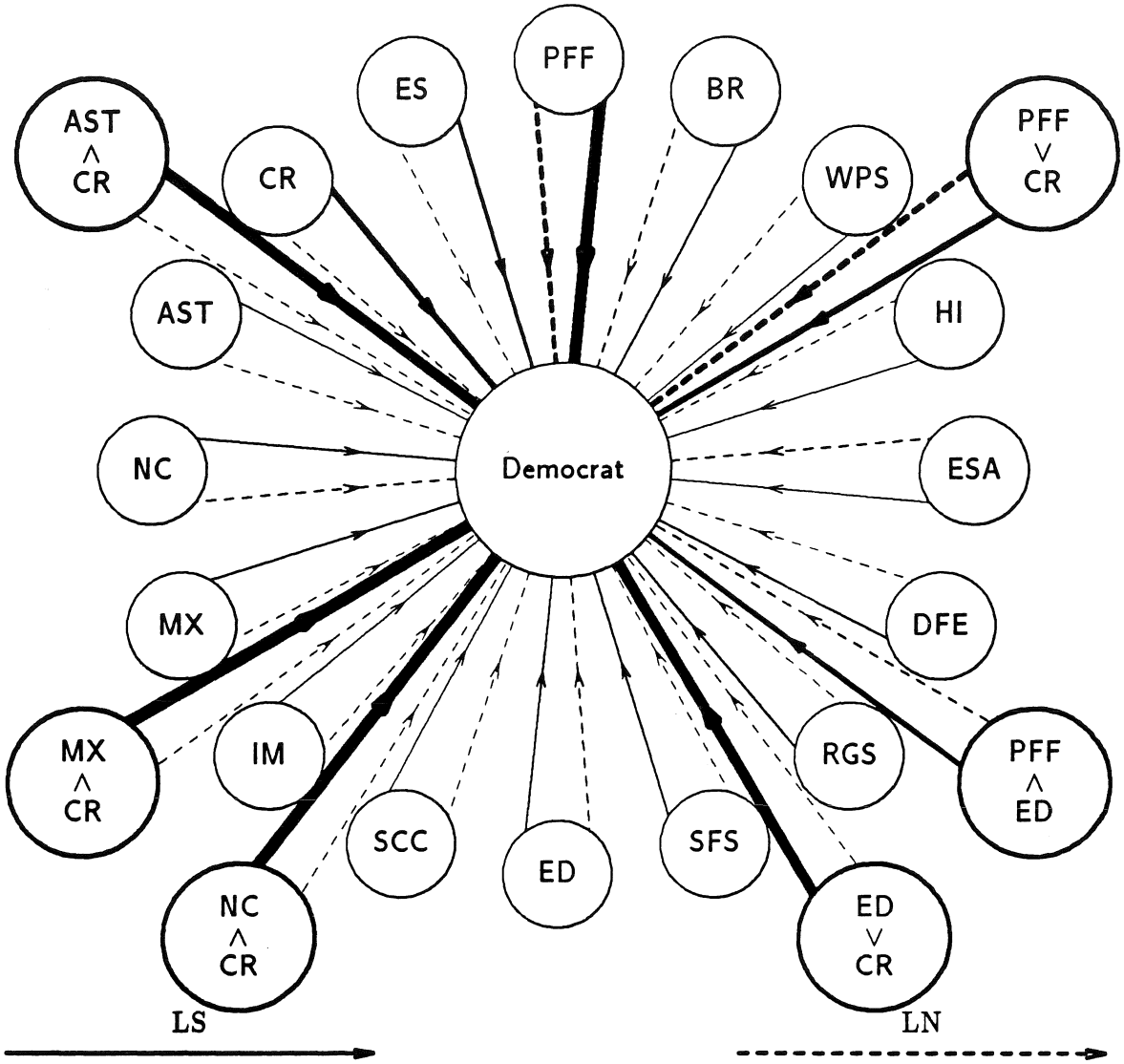

Figure 5.8: Typical concept description for a Democrat.

Table 5.14: Six chunks after processing 1,000 Congressmen.

#	CHUNKS	LS	LN
1.	(el-salvador-aid = nay) ∨ (physician-fee-freeze = nay)	18.14	0.04
2.	(crime = nay) ∨ (physician-fee-freeze = nay)	34.91	0.05
3.	(physician-fee-freeze = nay) ∨ (adoption-of-the-budget-resolution = yea ∧ el-salvador-aid = nay)	25.36	0.05
4.	(physician-fee-freeze = yea ∧ education-spending = yea ∧ crime = yea) ∨ (adoption-of-the budget-resolution = nay ∧ crime = yea)	0.12	10.37
5.	(physician-fee-freeze = nay) ∨ (adoption-of-the-budget-resolution = yea ∧ el-salvador-aid = nay ∧ education-spending = nay)	32.29	0.05
6.	physician-fee-freeze = yea ∧ duty-free-exports = nay	0.02	14.18

inaugurated chunks that have LS weights smaller than one, or LN weights larger than one. Their inversely predictive weights indicate that these chunks are functioning as negated patterns. In time, the chunk formation heuristics will attempt to invert them, so that they may accrue strong, positive LS and LN weights.

Concept acquisition in this domain confirms the assertion made in Chapter 4 that concept description size is effectively limited by the pruning heuristics. The inauguration of 6 chunks amounts to only a 19% growth in the size of the concept description. The averaged growth rate for the ten executions summarized in Figure 5.7 is a mere 13%. This is consonant with the mushroom data results. Estimating the effectiveness of the chunk proposal heuristics indicates that 10% of the chunks formed are inaugurated. (On the average, STAGGER makes 59 expectation errors over 1,000 examples, corresponding to $59 \times 2 = 118$ opportunities to add new chunks. Taking into account the complexity of the retained chunks, approximately 12 were at one time inaugurated.)

Though both the mushroom domain and Congressional voting domain afford approximately the same number of opportunities for adding new chunks, many more of the unproven chunks are waiting to be either pruned or inaugurated in the Congressional domain than with the mushrooms. For the latter domain these unproven chunks accounted for a 53% growth, yet in the Congressional voting domain these chunks account for a growth of 169%. Differences like this between the two domains, and STAGGER's relatively high performance in each, lends credibility to the assertion that the combined Bayesian weighting and Boolean chunking methods are widely applicable.

A bit of semantic analysis of the chunks in Table 5.14 reveals their intuitive validity in

this domain. For instance, Chunk 5 corresponds to votes in favor of policies that retain social spending, promote civil rights, while limiting defense spending; a statement highly consistent with the expressed policies of the Democratic party.

5.4.5 Conclusions

The task of determining the party affiliation of a Congressman in terms of votes already cast may not seem like a useful concept learning problem in and of itself. Perhaps it would be more useful to identify possible votes on new issues as a way of determining whether a bill was likely to pass. This would require the incorporation of background information regarding the nature of each bill. STAGGER is not fully equipped for this task; the weight and Boolean methods do not use any external information to guide the formation of a concept description. The required modifications would not be extreme, and many of the lessons learned could be retained.

However, this task has served to bolster the claim that STAGGER's methods are capable of handling poor quality data. The Congressmen, serving as examples, exhibit both unknown values and noise; for its part STAGGER consistently constructs effective descriptions of their party affiliations.

5.5 A comprehensive learning task revisited

In the previous sections of this chapter, I described STAGGER's ability to effectively classify novel examples in three domains: simple objects, edible versus poisonous mushrooms, and Democratic versus Republican Congressmen. In each case, the Boolean chunk learning method is able to form clear and effective chunks that improve the accuracy of the concept description. Though these latter two learning tasks reflect naturally occurring characteristics, they fail to exercise the range of possible concept learning situations. In Section 3.36, I introduce a set of 19 concept types as a way of assessing the completeness of STAGGER's learning methods. The conclusion drawn there is that the Bayesian weight adjusting method alone is insufficient; in the least it is representationally inadequate. The Boolean chunking method, which expands the concept representation, promises to alleviate this constraint. In fact it does as I illustrate in this section.

Recall that in Section 3.36, I defined an artificial domain of three binary-valued attributes. These enumerate eight possible objects, and if we define concepts as subsets of objects, there are 2^8 or 256 possible concepts. Simple attribute-value name substitution reduces this number of 19 distinct possibilities. Included among these are the trivial concepts of all objects, no objects, a single object, and a simple conjunctive concept, each of which STAGGER is able to accurately acquire using only the weight-adjusting method. However, these possibilities

also include the concept of exclusive-or for which the Bayesian weight method alone is insufficient. A primary hypothesis is that if the input to the weighting method were appropriately structured, then it could acquire any of the 19 concept types. The Boolean chunk learning method was developed to provide this representational augmentation for the Bayesian weighting method.

5.5.1 Empirical results

Repeating the empirical study of Section 3.5 indicates that combining the two methods results in effective acquisition for nearly all of the 19 concepts. As Figure 5.9 indicates, adding the chunking method extends STAGGER's acquisition capabilities. The height of the dark columns in the figure correspond to the average number of classification errors made prior to correctly classifying all examples for the combined methods. The background line represents the similar performance for the weight method alone (from Figure 3.36). The number above each column represents the number of executions (out of ten) that converged to a correct concept description after processing a maximum of 80,000 examples.

Roughly, there are three panels to Figure 5.9. On the left are the eight concepts that are learnable by the Bayesian weight method alone. Note that the combined methods are also able to acquire these concepts, and thus adding the Boolean chunking method to the weighting method does not significantly detract the latter's capabilities. Indeed, in some cases the combined methods incur fewer classification errors together (e.g., concept 3b) than the weight method alone (e.g., concept 7).

The middle panel of Figure 5.9 includes seven concepts that the weight learning method alone is unable to acquire. However, when combined with the recoding capabilities of the chunking method, these concepts are learnable. Generally, the increased time to converge on the concepts denoted to the right (e.g., concept 4a) reflects the need for search within the space of possible chunks. The left concepts require no search and are readily acquired by the Bayesian weighting method alone. However, those on the right require the formation of chunks to serve as new representational primitives. A number of examples are required before the appropriate elements for chunking are weighted, and further examples are needed to accrue valid chunk weights. This differential readiness to acquire some concept descriptions over others suggests a series of psychological experiments designed to test the ease humans and animals exhibit in acquiring the same concepts. I develop this point more fully in Section 7.4.

The rightmost panel of Figure 5.9 includes four concepts for which the combined methods did not always converge on a correct concept description despite the opportunity to process a large number of examples. In the worst case (concept 3a) the two methods form a correct concept description barely more than half of the time and on the average incur a large number of classification errors when they do. The clear implication is that no matter how effective

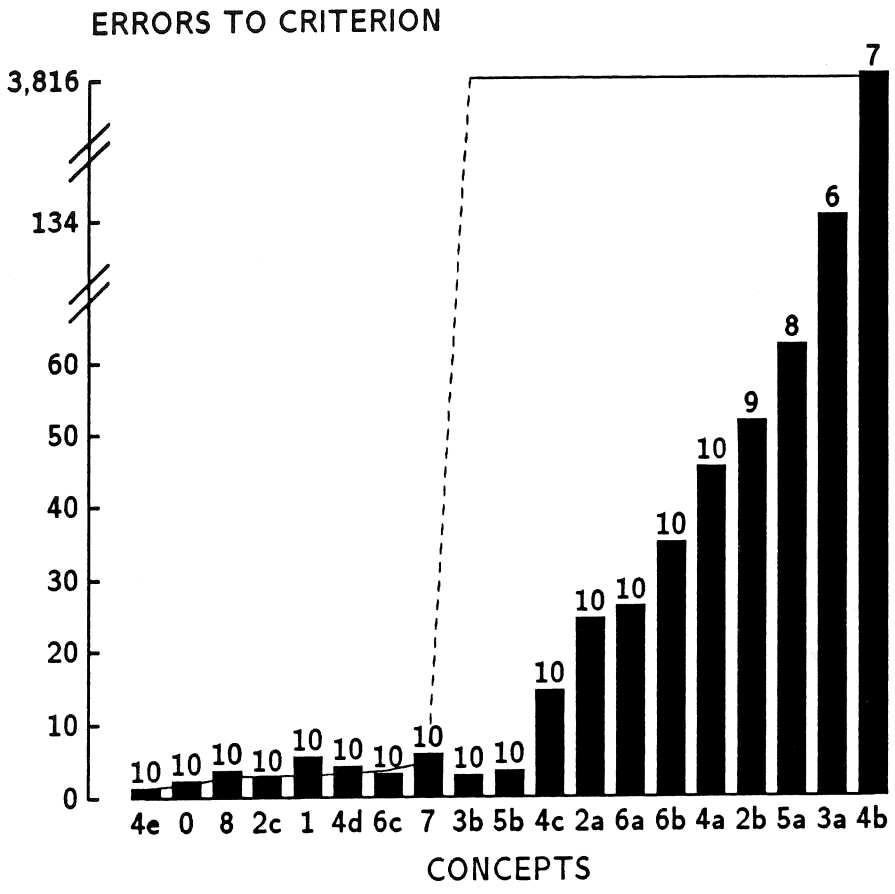

Figure 5.9: Learnable concepts with combined weighting and chunking.

the Bayesian weighting and Boolean chunking may be at acquiring some classes of concepts, they are not complete; they do not guarantee an optimal solution.

5.5.2 Appraising the results

Some of the concepts inflict more errors than others prior to convergence, and this provides an emergent indication of the difficulty STAGGER has in acquiring each of the concepts. Yet an indication of the *apriori* difficulty of learning each of the concepts is missing; an independent measure of the difficulty of each concept would help frame the results shown in Figure 5.9. The key question to be answered is whether STAGGER performs poorly on the concepts to the right in Figure 5.9 because its is a poor model or because the concepts are difficult. Following Fisher (1987), we can measure the dependence of the concept on its attribute-values. Equation 5.1 lists one possible measure.

$$D = \text{Max}_i \sum_j \sum_k p(A_i = V_{i,j}) \times p(C_k | A_i = V_{i,j})^2 \tag{5.1}$$

This equation measures the maximal ability to predict the class of an example C_k given the values $V_{i,j}$ of attribute A_i. Note that if the probability of a class is equal to one, then $p(C_k|A_i = V_{i,j})^2 = 1$ for all attributes. Thus $D(0) = 1$ and $D(8) = 1$ since concepts 0 and 8 (Table 3.18) each contain a class whose probability is one. Similarly, if knowing the value of one attribute perfectly indicates example class, then $p(C_k|A_i = V_{i,j})^2 = 1$ for that attribute, independent of the prior value of $p(C_k)$. In the case of concept 4e, the first attribute perfectly identifies the concept, and thus $D(4e) = 1$. On the other hand, if the examples are evenly distributed between classes ($p(C_k) = 1/2$ for all classes), and the class is not dependent on any single attribute, then the dependency is minimal. The exclusive-or concept of 4a illustrates this property since the examples are divided between classes and individual attributes do not increase the ability to predict example classes; $D(4e) = 1/2$. On the average (assuming that each of the 19 concepts are equally likely) the dependency measure is 0.74. Concepts with ratings below this are more difficult than average, while those with higher scores exhibit higher dependency and should be easier to acquire. Table 5.15 lists the 19 concepts and their associated dependency ratings.

Given this measure, we are now in a position to determine whether STAGGER does poorly on the concepts to the right of Figure 5.9 because the concepts are hard or because the model is poor. Figure 5.10 plots the ranking of each concept by Equation 5.1 and the corresponding difficulty STAGGER has with the concept.[8] To the left of the figure are difficult concepts (low dependency), and for the most part, these points are above the midline indicating that

[8]The vertical axis is a logarithmic scale. The two most difficult concepts (4b and 3a) have been omitted for scale considerations.

Table 5.15: Difficulty ratings for the 19 concepts.

	D	DESCRIPTION
0.	1.00	No objects.
1.	0.81	One object.
2a.	0.63	No attribute has a common value.
2b.	0.75	One attribute has one common value.
2c.	0.75	Two attributes have one common value.
3a.	0.56	All attributes have two common values.
3b.	0.81	One attribute has three common values.
4a.	0.50	All attributes have two common values.
4b.	0.63	One attribute has three common values.
4c.	0.63	Two attributes have three common values.
4d.	0.63	All attributes have three common values.
4e.	1.00	One attribute has four common values.
5a.	0.56	Inversion of 3a.
5b.	0.81	Inversion of 3b.
6a.	0.63	Inversion of 2a.
6b.	0.75	Inversion of 2b.
6c.	0.75	Inversion of 2c.
7.	0.81	Inversion of 1.
8.	1.00	Inversion of 0.

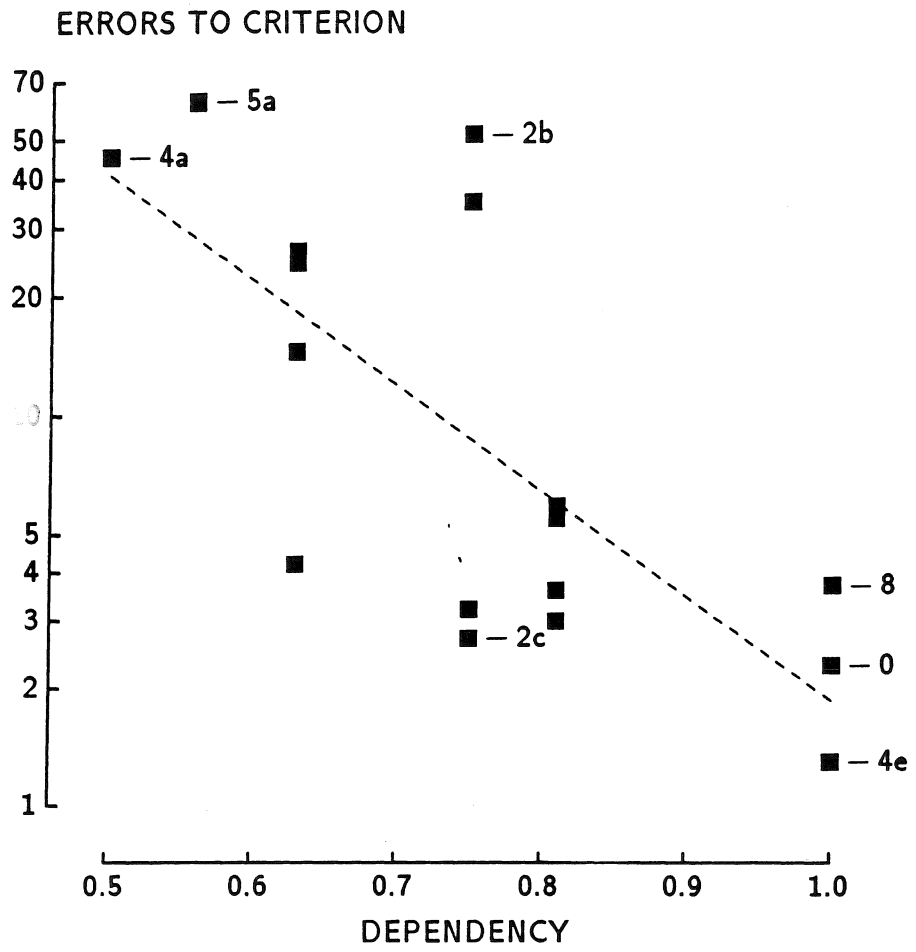

Figure 5.10: Learning speed as a function of concept dependency.

STAGGER had difficulty with them. On the right are highly dependent concepts that should be relatively easy to acquire. Correspondingly, these points lie in the lower half of the figure.

Recall that the five toughest cases for STAGGER are the concepts 4a, 2b, 5a, 4b, and 3a. Except for 2b, these concepts are ranked by Equation 5.1 as the most difficult. It seems that concept 2b should be easier than STAGGER's performance indicates. Conversely, though getting 75% correct for this concept is quite easy (6 of 8 examples are negative), in order to classify all examples correctly, the values of all three attributes must be tested. An example of this concept contains objects 1 and 4 (Table 3.17). The minimal Boolean function to distinguish these two objects from the other six is:

$$A_1 = V_{1,1} \wedge \begin{pmatrix} A_2 = V_{2,1} \wedge A_3 = V_{3,1} \\ \vee \\ A_2 = V_{2,2} \wedge A_3 = V_{3,2} \end{pmatrix} \tag{5.2}$$

In fact, 2b is incorrectly rated by Equation 5.1 as being as difficult as concept 2c. The

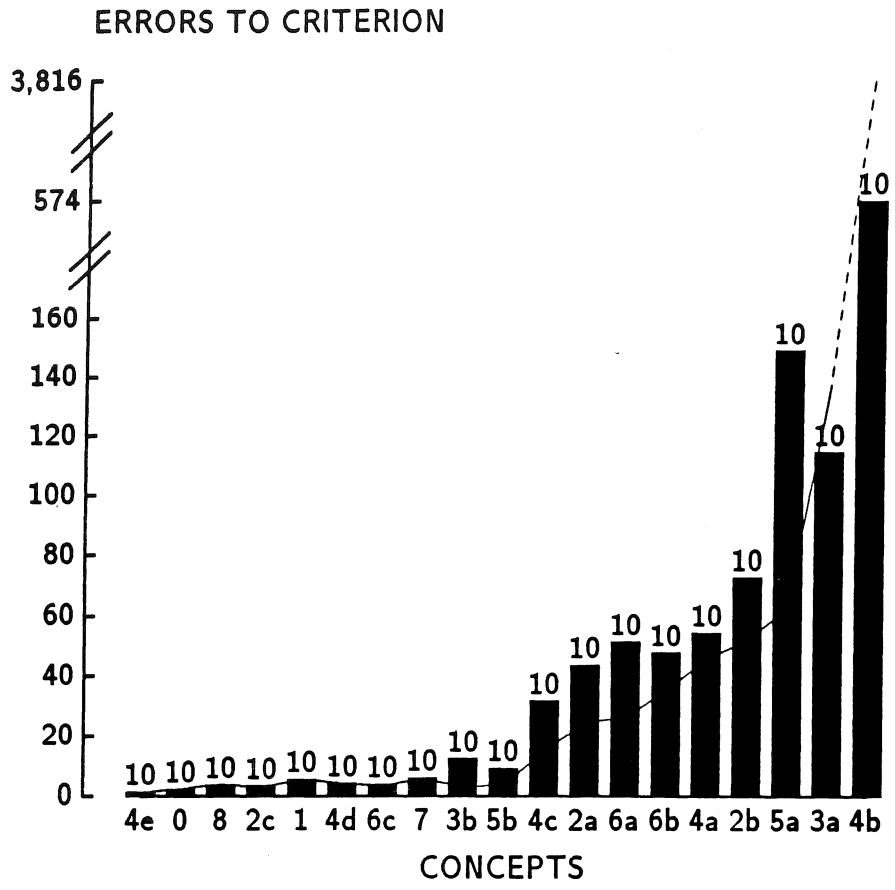

Figure 5.11: Learnable concepts with random chunk proposal.

latter is a simple, conjunctive concept and though they each share the same distribution of examples between classes (6 of 8 are negative), for 2c only two attributes need to be tested and a much simpler Boolean function suffices. From Figure 5.10 it is clear that 2c is easier for the Bayesian weight and Boolean chunking methods than acquiring concept 2b.

Though the use of Equation 5.1 allows appraising the performance results of STAGGER depicted in Figure 5.9, it has some shortcomings. Rather than introduce another measure (with perhaps its own deficiencies), a variant of STAGGER was developed to serve as a benchmark. Simply, the Boolean chunking method was modified so that instead of heuristically proposing new chunks (Section 4.2), they were randomly introduced following expectation errors. This small change retains the testing, inauguration, and backtracking processes illustrated in Figure 5.1. These results are shown in Figure 5.11, where the height of the dark column indicates averaged classification errors for random chunk proposal, and the background line indicates the equivalent results for heuristic chunk proposal.

There are three results of note in Figure 5.11. First, note that the two methods largely agree on which concepts are easy to acquire and which are difficult. This supports the conclusion

that STAGGER's weak performance on the concepts to the right of Figures 5.9 and 5.11 is due to the complexity of the tasks. Similarly, the concepts to the left are easily learned because they are simple.

Second, dropping the heuristic chunk proposal mechanisms results in reliable convergence for all of the concepts. Randomly adding new chunks guarantees that the proper intermediate chunks are proposed. If they are also rewarded by the Bayesian weighting method, they will be retained and can lead to the ultimately effective chunks in the long run.

On the negative side, random chunk proposal results in somewhat slower acquisition (incurs more expectation errors) than its heuristic counterpart for many of the middle concepts in Figure 5.11 as is evident by the line that cuts through the dark columns. These are concepts for which the Bayesian weight learning method requires assistance and the Boolean chunking method is able to provide it. To make the differential speed more apparent, Figure 5.12 plots the expectation errors of the random chunk proposal technique normalized by the preformance of heuristic chunk proposal. At the top of the figure, random is incurring three times as many errors, and at the bottom, 1/3 as many. For many of the concepts, randomly adding new chunks results in more errors, indicating that the heuristics of Section 4.2 are effective for these concepts. For two of the concepts the situation is reversed; the chunk proposal heuristics are negatively effective in these cases.[9] On the average, as indicated by the difference between the solid and dashed lines, randomly adding new chunks is incurs more errors in the course of concept acquisition.

The tradeoff between acquisition speed and the ability to reliably find an optimal concept description illustrates deficiencies in the chunk proposal heuristics outlined in Section 4.2. To their credit, these heuristics improve acquisition in many situations, but their lack of robustness contrasts sharply with other portions of STAGGER. It may be that subsequent refinement and simplification of the chunk proposal process will alleviate this deficiency.

5.5.3 Scale-up and the effects of irrelevant attributes

One of the limitations of this demonstration is that there are only three attributes, and it stretches the definition of induction when each possible example has been processed many times. This question of scale should not be ignored, for in many real-world tasks even if there are only three relevant attributes, it will be up to the learner to select them from among the many available.

Though an increased number of attributes means an increase in the number of elements, both of STAGGER's learning methods handle the expanded situation with relative ease. Since

[9]Not surprisingly, these two concepts are included in the four for which heuristic chunk proposal does not reliably converge. Indeed, if we consider the nonconvergent cases as highly erroneous, then there would be two more cases in which the random technique outperforms the heuristic one.

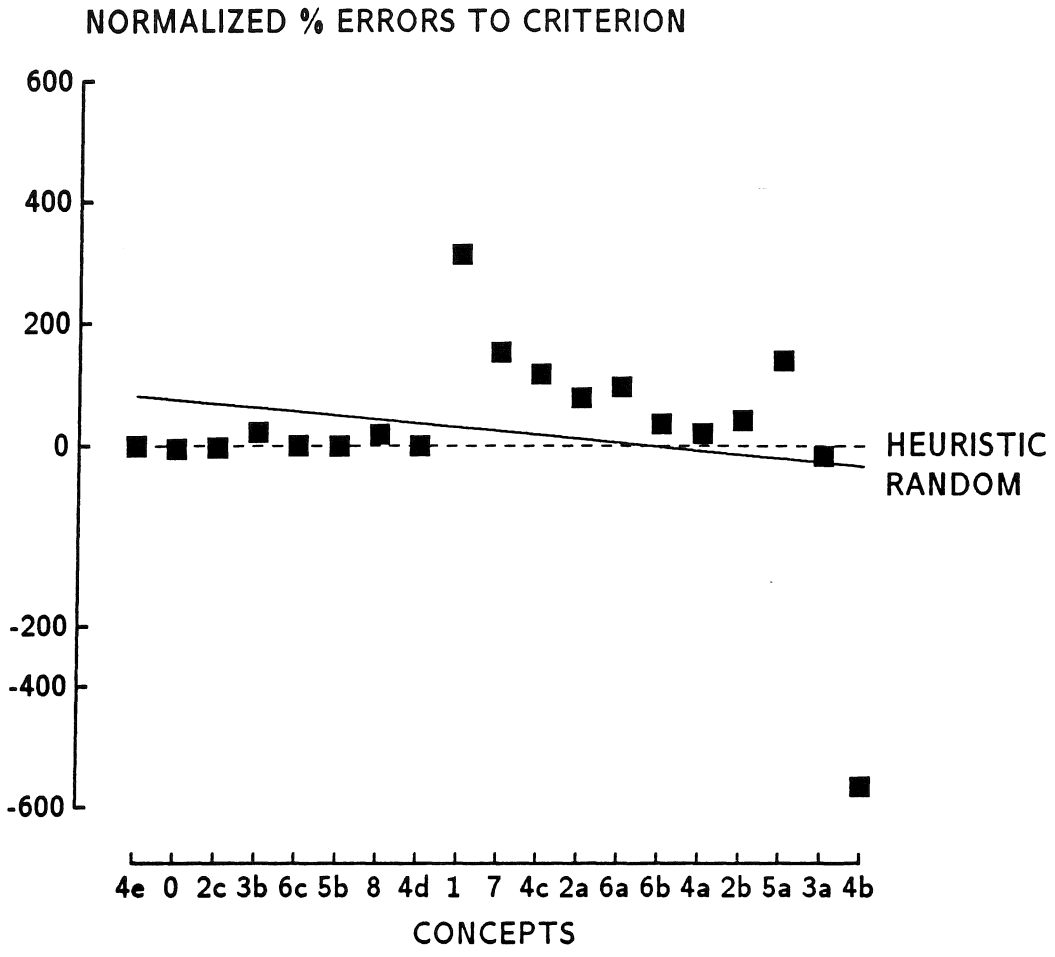

Figure 5.12: Learning speed for heuristic and random chunk proposal.

Bayesian weighting adjusts the weights of each element independently of the others, an increase in their number implies more processing per example but not more examples to obtain reliable weights. Each initial element essentially performs learning about its attribute-value pair in parallel of the others.

Boolean chunking may be more sensitive to an increase in the number of attributes since it spells hyper-exponential growth in the space of all possible chunks. Recall that Boolean chunking relies on the weights of existing elements to guide the formation of new chunks; it prefers to combine strongly predictive elements (Section 4.2.2). Yet, in the worst case, Bayesian weighting cannot provide this heuristic aid. For instance, in the exclusive-or concept (4a), the features are independent of the concept and, by definition, LS = LN = 1. However, even in this case once chunking gets going, the situation improves, and search is reduced. Though new chunks may be randomly added, only the effective ones remain. These then heuristically guide the formation of subsequent, highly effective chunks.

Some concept acquisitions situations will be even better. In the best case, the Bayesian weight learning method is able to identify the relevant attributes for the Boolean chunking method. Though the chunk search space is larger due to an increased number of attributes, search is properly directed by the element weights.

As empirical evidence for these claims, 3 of the 19 concept types above were modified to include additional irrelevant attributes. One concept tested is a simple conjunction (2c) and is learnable by the Bayesian weight method operating in isolation; the other two require both weighting and chunking. Of the latter pair, weighting provides some heuristic guidance for chunking in concept 6b; this test serves as a best-case for the combined methods sensitivity to example description size. The last test concept (4a) represents a worst-case, for the weight learning method cannot provide any heuristic guidance.

Each of these three test concepts (2c, 6b, and 4a) were generated as before and then randomly assigned a value for each of the additional binary-valued attributes. Besides adding just a few attributes, an extreme case was also examined where the irrelevant attributes outnumbered relevant ones 24 to 1. The relative difficulty of acquisition was measured by counting the number of classification errors made by the combined weighting and chunking methods until 100 consecutive examples were classified correctly. Each concept was tested over 20 executions with an additional 1, 2, 3, 6, 15, 33, and 69 irrelevant attributes. The averaged results appear in Figure 5.13.

The lowest line in Figure 5.13 indicates the ease of learning a simple concept for which the weighting method is sufficient (2c). As the number of irrelevant attributes grows, there is a slight increase in the errors made prior to reaching criterion. The middle line (with open symbols) denotes the differential ease of learning a concept (6b) requiring both learning methods. In this case too, there is a slight effect from an increased number of attributes.

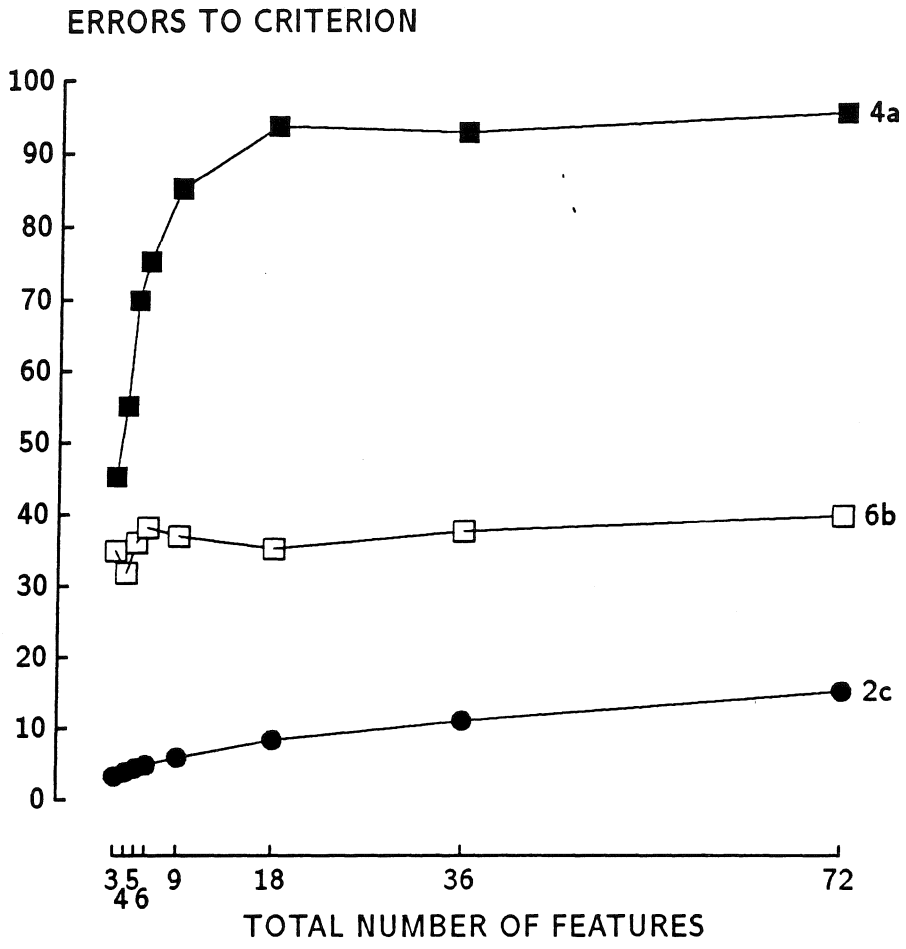

Figure 5.13: Learning speed as a function of irrelevant features.

The worst case, as expected, is shown by the upper line in Figure 5.13. Here the exclusive-or concept (4a) prevents the Bayesian weighting method from guiding chunk formation. Boolean chunking must initially proceed without the aid of these heuristics. Only after some partially effective chunks are discovered can chunking proceed normally. The initially steep rise in the curve is due to an initially sharp decrease in the percentage of relevant attributes; if percent relevant attributes were instead used as the horizontal axis, then errors rise evenly.

5.5.4 The effects of bias on learning

The premise of this section is that the Boolean chunk formation method is a necessary assistant to Bayesian weight learning for concept learning tasks denoted in Figure 5.9. This aid may be viewed as a process of relaxing the inductive bias inherent in the weighting method. Briefly, every inductive learning process assumes that concepts can be described in terms a restricted representation. For instance, LEX (Mitchell, 1982), assumes that the concepts to be acquired may be expressed by a conjunction of example features. Some restriction is necessary, for in the completely unrestrained case, there are an infinite number of hypotheses that account for the data. The restriction is termed *bias*, and without it a learner could do no better than guess randomly at a definition of the concept (Utgoff, 1986; Utgoff & Mitchell, 1982). This rigidity contrasts sharply with the intended flexibility of a learning system, yet in order to guarantee adaptiveness, some inflexibility must be embraced.

STAGGER's Bayesian weight method employs a strong bias since it makes highly restrictive assumptions about the concept form. I conjecture that only those concepts that are linearly-separable by a plane in the hyper-space of features can be represented and acquired. Intuitively, the weight method assumes that the concept is a simple conjunction or disjunction of example features. For instance, this strong bias does not include exclusive-or, and thus using only the Bayesian weighting method is inadequate for the acquisition of this concept.

The assisting role of the Boolean chunking method is a form of dynamic bias relaxation. By adding new chunks, the Boolean method is increasing the number of concepts that can be discovered by the weight learner. For instance, adding a new disjunctive chunk means that the concept could now involve an OR nested within an AND since the weight method can only form either a conjunctive or disjunctive description. Conversely, when the pruning heuristics used by the Boolean method remove a chunk, this strengthens the bias because it reimposes a restriction on the possibly learnable concepts. This continuous rewriting of example features allows the simple weight learning method to effectively search a large space of possible concepts.

However, the Boolean chunking method imposes its own bias on concept acquisition. Some concepts are more readily learned than others; some are not reliably learned at all. This is depicted in Figure 5.9. The tradeoff between comprehensive learning capability, e.g., acquiring

all of 19 concept types, and speedy acquisition is made evident by removing heuristic chunk proposal mechanisms are comparing the results. Figures 5.11 and 5.12 show this, for randomly proposing new chunks gains ground with respect to completeness but loses it on learning speed.

5.5.5 Conclusions

The Bayesian weight learning method is not sufficient for many concept learning tasks. In Chapter 3, I defined a set of 19 concept types and showed that the Bayesian weighting is capable of acquiring only 8 of them. Adding the Boolean chunking method described in the previous chapter allows STAGGER to acquire these concepts, though not all reliably. The interaction between the two methods is a type of representational learning, with Boolean chunking changing the substrate from with the Bayesian weight method's learning proceeds. However, due to limited size, this demonstration is perhaps best seen primarily as a platform exhibiting the necessity of augmenting Bayesian weight learning with Boolean chunking. The deficiencies of a weighting method and the need for an augmentation of its inputs is not a new observation. Hampson (1983) and Hampson and Kibler (1983) note this for a bi-layer perceptron-like learning method. Chapter 8 discusses this work in more detail.

Lest we judge the Boolean chunking method too harshly (it was not able to acquire all of the concepts all of the time) note that many well respected learning systems assume that the concept will be conjunctive and are therefore searching an exponentially smaller space of possible descriptions. These methods are capable of less than the Bayesian weight learning method alone and are in need of their own assisting technique.

5.6 Overview

This chapter illustrated the function of the Boolean chunking method and motivated its existence. Chunking relies on a generate and test paradigm for finding effective combinations of existing descriptive elements. Generation is triggered by expectation errors and forms new chunks by conjoining, disjoining, or negating existing ones. The predictive indicators assigned by the Bayesian weighting method serve to direct the choice of new chunks; they also act as a threshold to test a new chunk's validity. By pruning ineffective chunks, the Boolean chunking method retains a concise concept description, and those that survive the test are likely to be strongly involved in concept expectation. However, if a poor chunk manages to surpass its threshold and become a part of the concept description, backtracking is triggered to remove the ailing chunk. Examining each of the subelements within a chunk allows reintroducing a simpler element that lacks the ill features.

In the first section, I stepped through the acquisition and refinement of a simple concept description. Repeating the sequence of examples utilized in the first demonstration in Chap-

ter 3 demonstrated the error-driven formation of new chunks and their testing. Those that were incorrect for this concept were pruned, while the few accurate ones were inaugurated into the description. Further, poor chunks were improperly accepted, a situation easily remedied by the dependency-directed backtracking process as it introduces a simpler and more accurate chunk.

In the second section, I turned from this trivial task to a more realistically constrained one. Here, STAGGER acquired a description of an edible versus poisonous mushroom. This task involves coping with a large number of attribute-value pairs and a correspondingly large space of possible concepts. Furthermore, the task requires making fine distinctions, for in at least one case, there are examples of poisonous and edible species that are remarkably similar. STAGGER is able to construct a relatively effective description of an edible mushroom, and the chunks formed by the Boolean method correspond nicely to the underlying structure of mushroom species.

The third section serves as a platform to discuss the effects of unknown values within examples on concept learning. The specific task is to describe Democratic versus Republican Congressmen in terms of their voting records. The resulting descriptions are on the average quite accurate considering that nearly 1/2 of the examples have at least one of their values unspecified and that many of the examples are nearly contradictory. Both this demonstration and the mushroom learning task empirically support the claim that concept description size is bounded by the number of attribute-value pairs.

Lastly, I returned to the comprehensive set of learning tasks presented at the end of Chapter 3. In this section I showed that the addition of the Boolean chunk learning method dramatically improves the acquisition ability of the Bayesian weighting method over the 19 concept types. This served as an opportunity to characterize the interaction between the weight and chunking methods as a form of representational learning, with the latter serving to change the learning substrate for the former. This type of cooperative adjustment of concept bias may prove useful for other learning methods.

This chapter has the twin focuses of clarity and generality. The first focus is served in part by the initial, simple concept learning example, and the latter is served by the inclusion of two differing, naturally constrained domains. If this chapter has succeeded in clearly illustrating STAGGER's methods, it has yet to fully explore the generality of them. For instance, in both the mushroom and Congressional learning, the issue of noisy examples arises, yet this chapter has done little to explore their ultimate effects. Moreover, the Congressional learning task indicates the need for learning methods to face the possibility that examples may be underspecified. What are the eventual effects of these unknown values? The next chapter seeks to answer these and related questions by framing simple, abstract learning tasks and then introducing these observed characteristics into them in a controlled manner.

Chapter 6

Learning with Poor Quality Examples

6.1 Introduction

The second chapter presented the Bayesian weighting method that was capable of acquiring simple concepts. The next chapter then explicated this method in a number of concept learning situations, closing with a demonstration of its limitations. These limits served as motivation for the introduction of the Boolean chunk learning method, which forms new combinations of existing descriptive pieces. After describing this processes of this method in Chapter 4, I turned in Chapter 5 to an explanation of the method's operation within a number of domains. That chapter closed by demonstrating that the combined methods are sufficient for learning a larger class of concepts than just the Bayesian weighting. Along the way, the characteristics of poor quality examples arose in a number of the concept learning tasks. At each outcropping, the text focused on the ability of the methods to withstand the poor data, but in general the effects of low quality data have yet to be considered fully. This chapter seeks to examine the effects of differential amounts of degradation in the data through a series of experiments varying the quality of the data and measuring the resulting effects on derived concepts.

Specifically, I explore three classes of example degradation in this chapter: noisy examples, concept drift, and unknown values. The first focuses on the effects of contradictory examples, or noisy data. This was evident in the domains of information retrieval, Congressional voting, and mushroom classification; it typically arises whenever there is the possibility of measurement error, subjective interpretation, or ill-defined thresholds (Quinlan, 1983). A second class of poor data arises when the concept itself changes. Anecdotally, consider the fox who adapts as summer brings out the color in his prey's coat and winter removes it. STAGGER's tolerance to drift was briefly illustrated in Section 3.3 where the goal location in a simple spatial world moved. Generally, concept drift may arise whenever the concept is inherently based on seasonal variations, economic factors, or influences outside the learner's scope of attention. The third class of example degradation arises when examples are underspecified. This may have a

serious effect on derived concept quality and is a marked characteristic of the Congressional voting domain.

In this chapter I discuss these topics in individual sections. The centerpiece of each discussion is an empirical study of the effects of poor data on STAGGER's ability to form effective concept descriptions. To maintain control over the results, I have construct artificial tasks for these studies. If naturally constrained domains were used, it would be difficult to quantifying the existing effects of the factors under study.

6.2 Noisy examples

Concept examples may be of suboptimal quality for a number of reasons. In this section I consider poor quality data in the form of noisy examples. Data can become noisy due to the effects of faulty measurement (poor quality instrumentation), ill-defined thresholds (poorly defined assessment standards), and subjective interpretation (Quinlan, 1983). The naturally constrained domains examined in previous chapters exhibit some forms of example noise. In Chapter 3, applying STAGGER to the task of improving information retrieval required the ability to tolerate inconsistency both in the query formulation and in the user's assessment of document relevancy. Chapter 5 also included some demonstrations of potentially noisy domains. In the mushroom classification task, determining edibility required assessing the color and smell of the sample, both of which are highly subjective measurements. The presence of noise is perhaps more noticeable in the Congressional voting domain. In this domain describing a Democrat is confounded by Congressmen who did not vote according to their party's platform.

6.2.1 Uniform noise

Within the framework of induction, noise may be defined as examples that are in differing classes (positive and negative examples) with identical descriptions. For instance, in the Congressional voting domain, if a Democrat and a Republican voted identically on the votes surveyed, then these two examples would be noisy with respect to each other. However, the actual type of contradictoriness noted in that demonstration was where two Democrats voted opposite on nearly every issue. These two types of noisiness are functionally identical; consonance can be achieved in either case by simply relabeling one of the two examples as belonging to the other class. Changing the party of either of the agreeing Democrat and Republican removes the contradiction; moving either of the two contrary Democrats into the Republican party also does.

STAGGER's learning methods are able to make the most of noisy situations. Within a noisy example, features are incorrectly paired with the identity of the example. The features of a

true positive example are falsely associated with a nonexample and vice-versa. The weight method is relatively tolerant of deviant examples, because any single example does not have an overriding influence on the relative magnitudes of the counts underlying computation of the weights (Equation 2.6). The weights instead are based on a summative effect of the influence of a large collection of examples.

The Boolean chunking method is also relatively tolerant of noisy examples. Incorrectly identified examples tend to cause STAGGER to make expectation errors, thus triggering the proposal of new chunks. If the introduced chunks prove to be effective members of the concept description, then so much the better. Otherwise, the weighting method allows the pruning processes to determine that they were added spuriously. Like the weighting method, the Boolean method is not completely influenced by an single example. A consistent regularity is necessary for a chunk to be introduced and subsequently retained.

To empirically assess these claims, I tested STAGGER's acquisition ability in an artificially constructed domain of three trinary-valued attributes similar to those used for the detailed examples at the beginning of Chapters 3 and 5. The two concepts tested were a simple conjunction and disjunction of two attribute-values. First, the methods were trained on varying degrees of noisy examples, and then the quality of the derived descriptions was tested by classifying noise-free examples. Noise was introduced into a set of examples by assigning some percentage of the examples a random identity.[1] The experiments were repeated 10 times for the noise percentages of 0, 20, 40, 60, 80, and 100. Figure 6.1 depicts the asymptotic performance of STAGGER on a simple conjunctive concept as measured by classifying noise-free examples after training on noisy examples.

The flatness of the curve and its abrupt drop closely approximate the best possible performance: acquiring the correct concept description whenever the examples convey any information. STAGGER is able to acquire a high quality concept up to the point where all of the examples are contradictory and there is no information in the data. Similar results hold for simple disjunctive learning as well. Figure 6.2 plots performance after training on a noisy disjunctive examples and then testing on noise-free examples. This curve is also close to the best possible performance, and it indicates that methods like STAGGER's are able to effectively tolerate high levels of noisy data.

In a deep sense, STAGGER's tolerance of noise arises becauses it does not attempt to account for the identity of every example. This appears to be in contrast to the desirable qualities of *completeness* and *consistency* (Michalski, 1983). The former asserts that abstractions of examples should include all positive examples in the positive class; the latter, that abstractions should exclude all negative examples. If some examples are noisy, then it is not possible to

[1]Alternatively, some percentage of examples could have been assigned the incorrect class. This approach is functionally equivalent to randomly identifying twice as many examples.

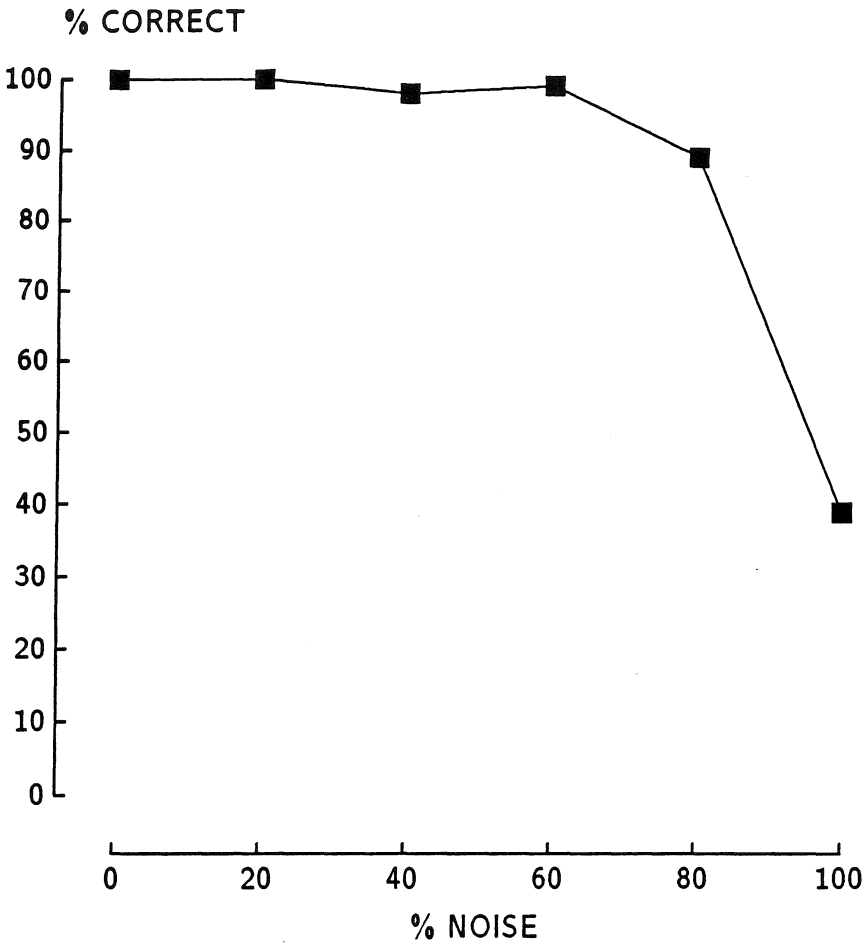

Figure 6.1: The effect of noise on acquisition of a conjunctive concept.

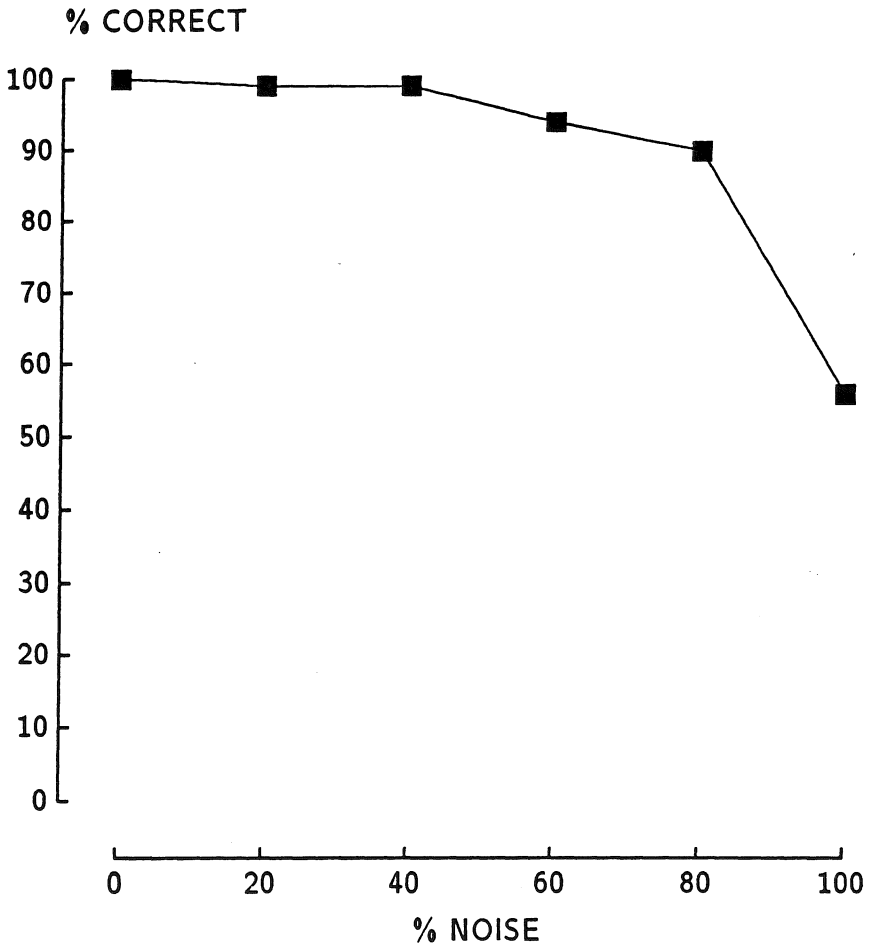

Figure 6.2: The effect of noise on acquisition of a disjunctive concept.

meet these two criteria. By definition, if two examples have identical values but come from differing classes, then it is impossible for any abstraction to match all positive examples but none of the nonexamples. The generalization of these two measures is of course simply to maximize their tradeoff, to find an abstraction that includes most of the positive examples and excludes most of the negatives ones. The next subsection differentiates between types of noise and in doing so sets the stage for generalizing the completeness and consistency criteria.

6.2.2 One- versus two-directional noise

The previous subsection focused on the effects of uniformly noisy examples on concept acquisition. However, considering a finer distinction between possible noisy situations clarifies a relationship between concept acquisition, psychological learning studies, and statistical dependence. Rather than assume that all of the examples are equally likely to be incorrectly identified, positive examples may be noisier than negative examples, or vice-versa. For instance, in the Congressional voting domain one party's representation may be chaotic while the other's is consistent.

This distinction between types of noisy situations arises naturally if we consider the functioning of physical measuring devices. Some instruments are prone to one-directional errors, like the leaky rain gauge that may read too low, but never reads too high. Other instruments tend toward two-directional errors; they are not predisposed to any particular type of inaccuracy. The temperature gauge that is rated to within 10% is an example of this type of noise; it may read too high or too low. In both cases the occasion for error is random, but in the one-directional situation, the type of error is highly constrained.

One-directional noise preserves a large amount of information. Knowing that the rain gauge is empty does not indicate either rain or drought, but a positive reading from it definitely indicates precipitation. Another one-directional error-prone instrument might be a smoke alarm that sometimes "false alarms" at cooking bacon but would not fail to siren in a blaze. Information is preserved in this case not in the alarm, but in its absence.

Mapping back into the Congressional voting domain, a one-directional errorfull situation is where only one parties' Congressmen are contradictory. If the Democrats were in an uproar over their platform and the Republicans were consistent, then information would be retained in the latter party.

These situations have a strong analogue with psychological experiments. The one-directional situation corresponds closely to partial reinforcement experiments. For instance, subjects in classical conditioning experiments may receive some cue as an indication of the onset of an aversive stimulus. If the cue frequently false alarms but never fails to indicate the noxious stimulus, then subjects readily learn that the cue is a predictor of the stimulus (Fitzgerald, 1963). This is the situation with the overzealous smoke alarm and its owners.

This phenomena also has a strong mathematical foundation. One-directional noise preserves the notion of statistical dependence. If the probability of the stimulus is different in the presence of the cue than in its absence, then the two are not independent events (Fine, 1973). In the partial reinforcement experiments, this arises because given the absence of the cue, the probability of the stimulus is zero.

None of these information properties hold true for the extreme two-directional noisy situation. Complete contradiction between the two political parties, a highly noisy temperature gauge, a cue that sometimes predicts a stimulus and sometimes does not, all leave the situation with little more information then they found it. In psychology this is the primary result leading classical conditioning researchers to adopt the random (two-directionally noisy) condition as a control (Rescorla, 1967). Similarly, in terms of statistical dependence, in an extreme two-directional noisy situation between a cue and the stimulus, the probability of each is independent of the probability of the other.

There is a continuum of conditions ranging from no-noise conditions, to one-directional noisy conditions, to completely two-directional noisy conditions. Each of these conditions has a precise correlate in terms of the notion of statistical dependence. Two events are statistically independent if and only if the probability of their joint occurrence is equal to the product of their individual probabilities, or $p(E_1 \wedge E_2) = p(E_1) \times p(E_2)$ (Fine, 1973). This equation can be plotted as a surface in the space of four possible joint event probabilities as shown by Figure 6.3.[2] The volume shown in this graph is the space of all possible probability distributions over the two events. The axes correspond to joint event probabilities with the vertical axis indicating the probability that the two events will co-occur, and the two horizontal axes depicting the probabilities that one event will occur without the other. The fourth probability, that neither event will occur, is fully constrained by the other three since the four probabilities must sum to one.[3] The surface in Figure 6.3 results from highlighting those probability distributions (points) for which $p(E_1 \wedge E_2) = p(E_1) \times p(E_2)$.

Every possible correlational situation between a cue (E_1) and a stimulus (E_2), between an instrument's reading (E_1) and the actual state (E_2), or between a Congressman's vote (E_1) and his party (E_2) may be placed somewhere in Figure 6.3. Noise free situations, were the two events are perfectly correlated, correspond to points along the vertical axis. Here the probabilities of events occurring separately is zero (no displacement along the horizontal axes). Slightly noisy situations lie in the area between the origin of the axes and the surface of independence. In situations that lie along the surface, the two events are statistically independent, and one event does not convey any information about the other.

[2]The derivation of the equations for this surface as well as a consideration of the psychological implications of it appear in (Granger & Schlimmer, 1986). An independently discovered derivation of similar nature appears in (Gibbon, Berry, & Thompson, 1974).

[3]In terms of the x, y, and z coordinate axes, the equation $p(E_1 \wedge E_2) = p(E_1) \times p(E_2)$ is $y = z(1-x-z)/(x+z)$.

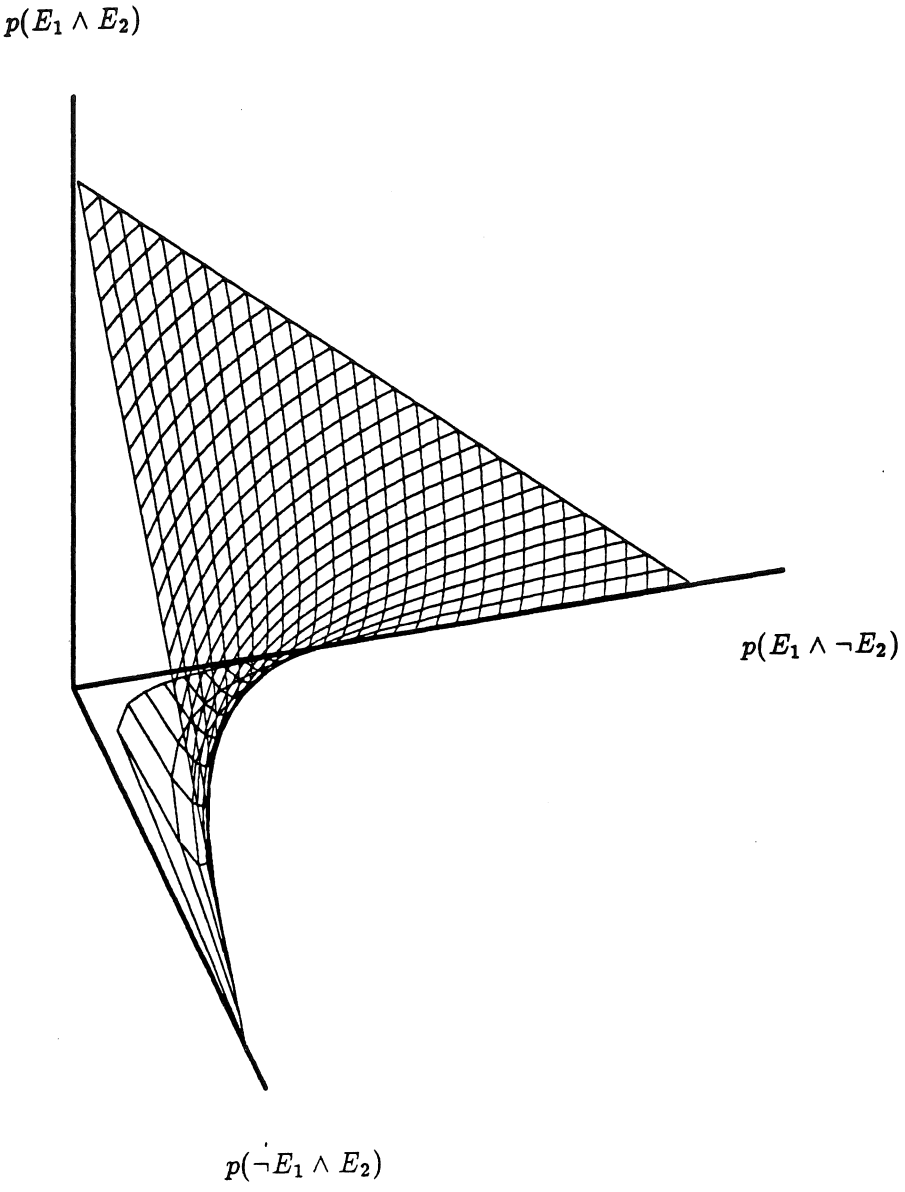

Figure 6.3: Surface of statistical independence.

The saddle shape of this surface reflects the close correspondence between the intuitive characterizations of one-directional noisy situations as information full and two-directional situations as wholly noisy. To make this visual mapping, reconsider a few hypothetical faulty instruments. A perfect instrument would always report the event when it occurred and never when it did not. This corresponds to $p(E_1 \wedge \neg E_2) = p(\neg E_1 \wedge E_2) = 0$ and thus to points along the vertical axis.

The one-directionally noisy instrument exhibits the absence of one of the types of solitary events; either $p(E_1 \wedge \neg E_2) = 0$ or $p(\neg E_1 \neg E_2) = 0$. There are no occasions where the house is burning while the smoke alarm is silent, though it may sound off during cooking. Graphically, these types of noisy situations correspond to the planes delimited by one of the horizontal axes and the vertical axis as Figure 6.4 portrays. The entire space of either of these planes is within the area of statistical dependence. In any completely one-directional noise situation, the two events are statistically dependent.

Conversely, two-directional noisy instruments create both types of solitary events. The temperature gauge may falsely indicate either overheating or cooling, thus $p(E_1 \wedge \neg E_2) \neq 0$ and $p(\neg E_1 \wedge E_2) \neq 0$. These situations correspond to the area between the two one-directional noise planes. In the uniform noise case examined in the previous subsection, $p(E_1 \wedge \neg E_2) = p(\neg E_1 \wedge E_2) \neq 0$, and these situations lie along the diagonal plane hinged on the vertical axis and located between the two horizontal axes as depicted by Figure 6.5. Many of these two-directional noise cases exhibit some statistical dependence, typically when the noisy events are overwhelmingly of one type or the other (and thus the situation is more one-directional), or when the noisy events are relatively few in number.

STAGGER's weight definitions reflect the distinction of statistical dependence shown in Figure 6.3. This can be shown mathematically through a relatively simple proof that demonstrates the equivalence between relevant LS and LN weights and dependence between a feature and an outcome (Schlimmer, 1986). Following the interpretation of Figure 6.3, STAGGER's weights readily accommodate noise-free and one-directionally noisy situations and are somewhat more reticent to assign predictiveness to features that exhibit two-directional noise.

These measures generalize the completeness and consistency criteria for inductive abstractions. As I noted above, it is not possible for an abstraction to include all positive examples and exclude all negative examples if some examples are noisy. However, it is possible to find a tradeoff between the two, to uncover an abstraction somewhere between matching all examples (and thus being complete) and matching none (guaranteeing consistency). The two element weights correspond closely to these criteria; the LN weight measures completeness, and the LS weight indicates consistency. To see this, note that an element with a strong necessity is one whose absence indicates a nonexample ($\neg matched \Rightarrow \neg example$), or an element that is never omitted in an example (contrapositive, $example \Rightarrow matched$). Thus a strong necessity

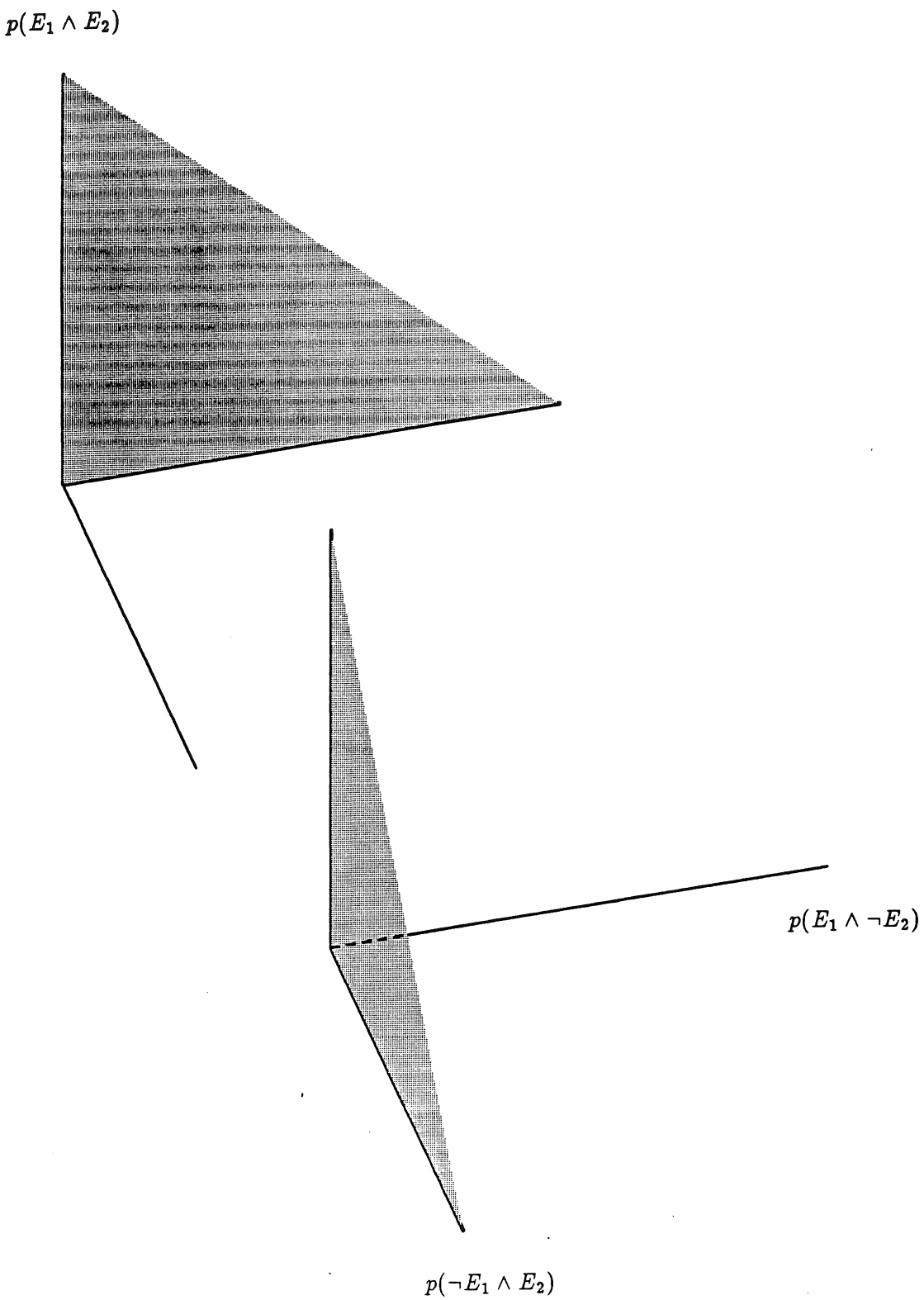

Figure 6.4: Planes corresponding to one-directional noise.

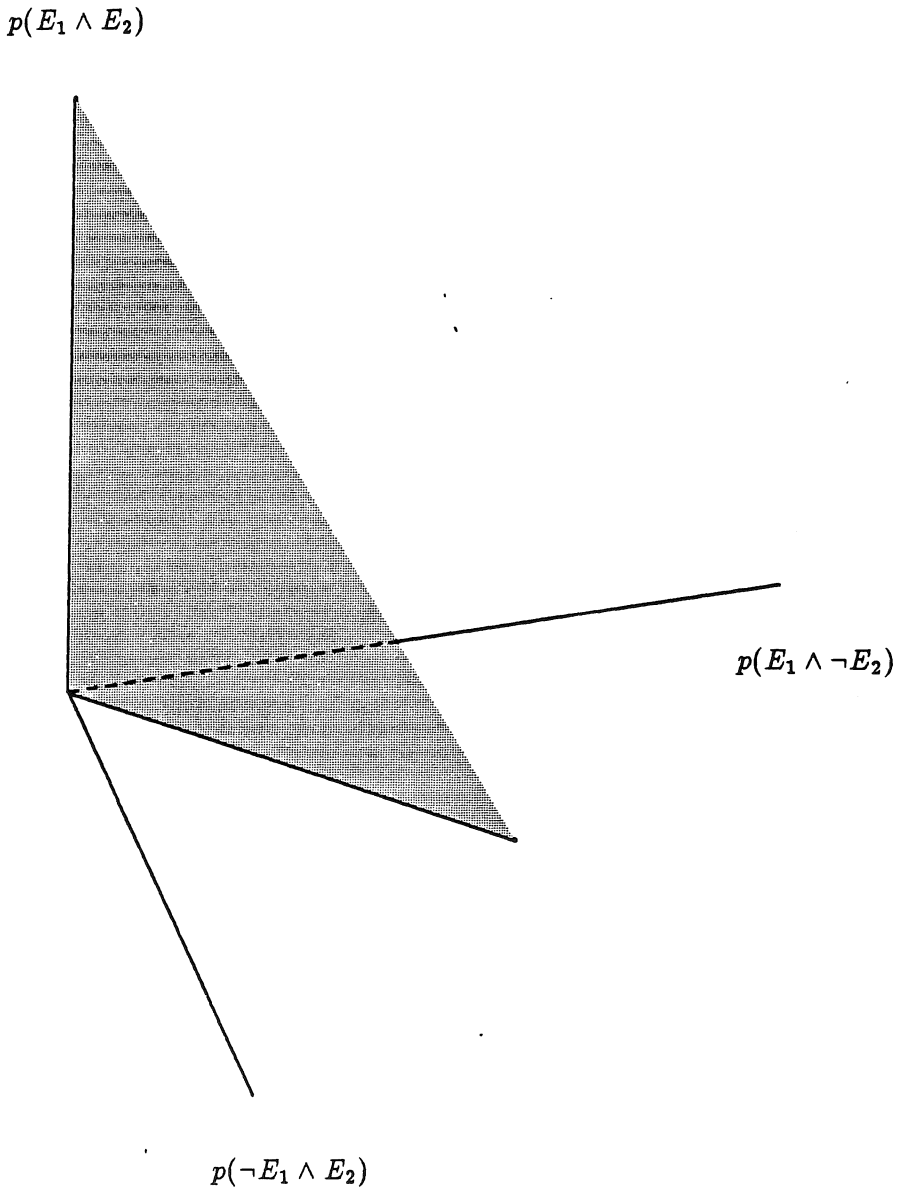

Figure 6.5: Plane corresponding to even, two-directional noise.

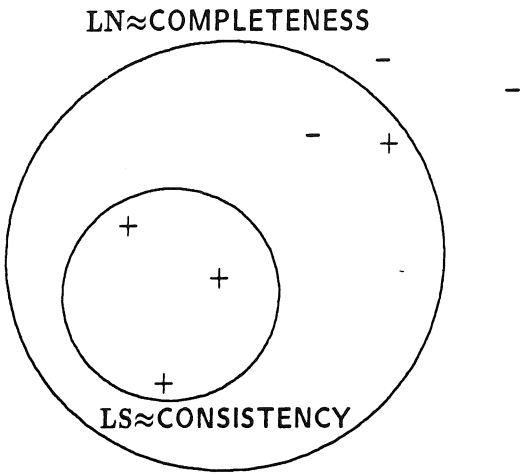

Figure 6.6: Interrelation between LS, LN and Completeness, Consistency.

weight is an indication of an element that matches all positive examples. Conversely, a strong LS weight indicates an element that is never true of a negative example (contrapositive of *matched* $\Rightarrow$ *example* is $\neg$*example* $\Rightarrow$ $\neg$*matched*); an element that is consistent.

This interrelation is graphically displayed in Figure 6.6. In noise-free domains it is possible to find an abstraction that will perfectly cover the positive examples and none of the negative examples (the circles will equal). This is equivalent to saying that it will be possible to find a single Boolean element that has infinitely strong LS (high) and LN (low) weights. Experience with a few domains in Chapter 5 indicates that this is typically not the case; rather it is more likely that some sufficient and some necessary chunks are formed.

Empirical studies confirm the estimate of STAGGER's tolerance of noisy examples. Following the same basic procedure used for the uniform noise experiments, I trained the methods to asymptote on varying percentages of randomly identified examples. Then the quality of the acquired descriptions was measured by classifying noise-free examples. This set of experiments subsumes the others, for instead of just testing an equal percentage of noisy positive and negative examples, these noise levels were varied independently over the two classes at the noise levels of 0, 20, 40, 60, 80, and 100. Figure 6.7 displays the surface resulting from training STAGGER on varying levels of noise in a simple conjunctive concept and test classifying noise-free examples. The vertical axis corresponds to the quality of the learned concept description, and the two horizontal axes indicate the degree of noisiness in the positive and negative examples separately. The noise-free training situation is located at the top center of the diagram, completely noisy training at the bottom center, and the uniform noise training cases along the diagonal line connecting the two. Figure 6.7 indicates that the quality

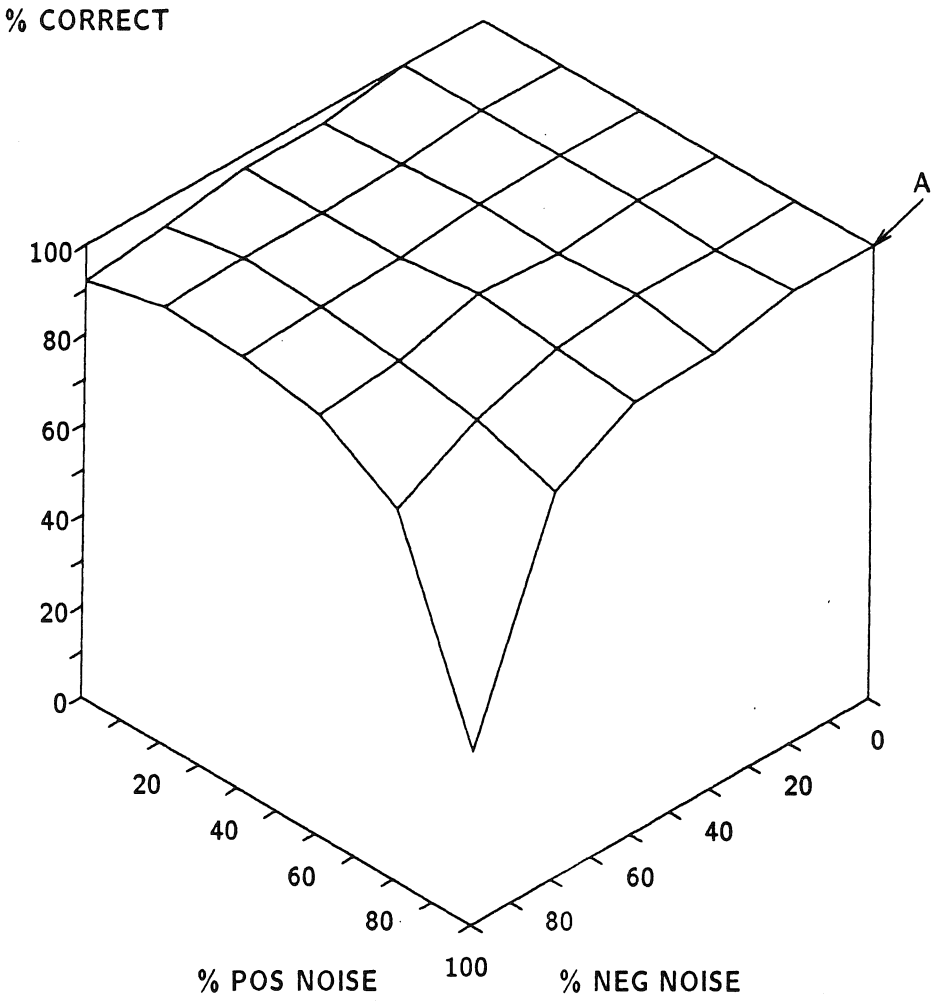

Figure 6.7: The effects of noise on conjunctive concept acquisition.

of the acquired concept description closely approximates the best possible, for classification performance remains high up to the point where all of the examples are completely noisy.

Following the positive noise axis out to its maximum of 100% and leaving the negative noise at 0%, Point A in Figure 6.7 corresponds to the situation where all of the positive examples have been randomly identified as positive or negative, but the identities of the nonexamples remain intact. Even in this extremely noisy situation, STAGGER is able to acquire an effective concept description. This training situation is the direct analogue of the leaky rain gauge, for some of its positive examples (rainy days) appear as negative ones (due to leaks), but all of its nonexamples (dry days) are preserved intact. Examining the point opposite point A indicates that this performance is also exhibited in training with the extreme one-directional noisy negative examples.

Duplicating this experimental setup for a disjunctive concept indicates that methods like STAGGER's are also capable of acquiring effective ORs across a range of noisy situations, up

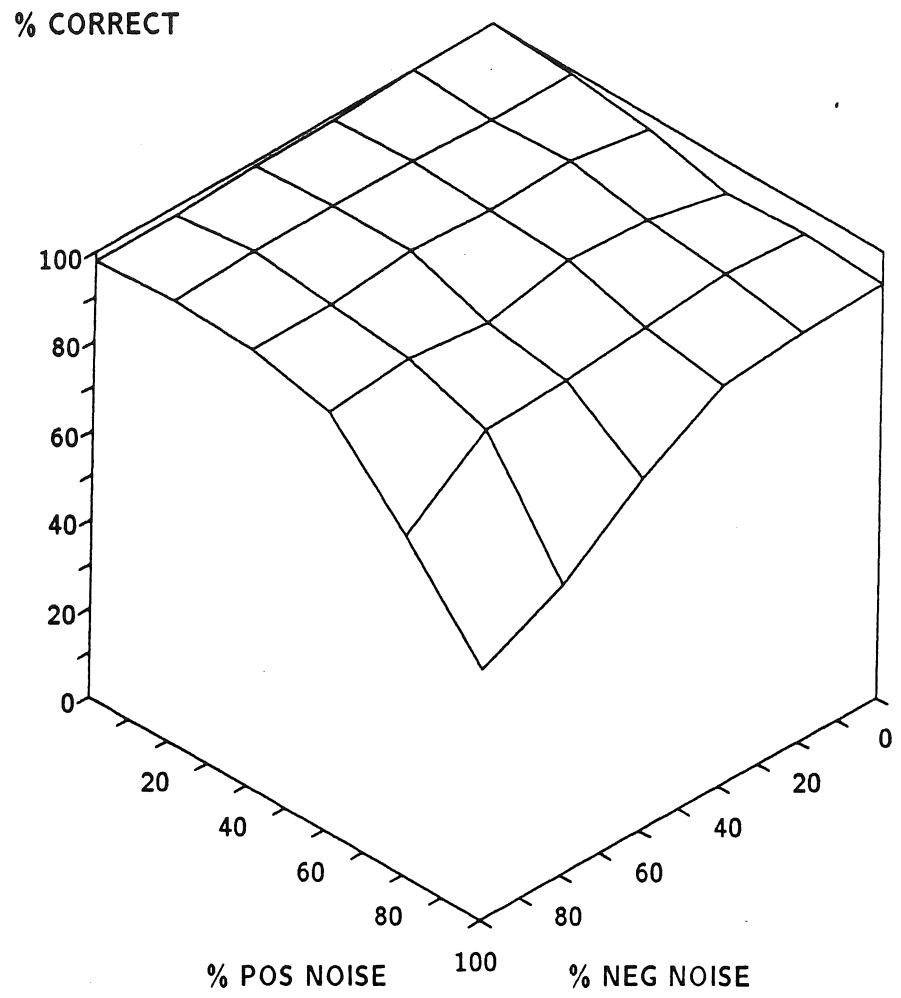

Figure 6.8: The effects of noise on disjunctive concept acquisition.

to the point where all of the examples are noisy (bottom center). See Figure 6.8.

Careful inspection of the two figures reveals an important asymmetry between the differential types of one-directional noise and the concept's form. Specifically, point A (corresponding to only noisy positive examples) in Figure 6.7 and the point opposite it on the extreme left (corresponding to only noisy negative examples, full displacement along the negative noise axis but none along the positive noise axis) are significantly different. In STAGGER, noisy negative examples are more detrimental to the acquisition of a conjunctive concept than noisy positive examples. Conversely, as Figure 6.8 indicates, noisy positive examples degrade disjunctive learning more strongly.

The source of this asymmetry is based on the notions of sufficiency and necessity. First, consider the case for a conjunctive concept like *father*: a parent and a male. Both of these features are necessary for identification of a father, yet each occur frequently in nonexamples; a brother is also a male; mom is another parent. Considering the role of each of the conjuncts,

this maps directly into a one-directional situation where only the positive examples are noisy. Adding more noise to the positive examples does not qualitatively change the predictiveness of the individual conjuncts, as Figure 6.3 suggests; the situation is still one-directional. However, adding noise to the negative examples introduces the second type of directionality into the noise; this moves each of the conjuncts away from one of the planes in Figure 6.4 (a predictive situation) to somewhere in the middle of Figure 6.3. The result is that each of the conjuncts is less predictive, and the Boolean chunk learning method is less likely to form the appropriate combinations.

Figure 6.8 also displays an asymmetry between averaged classification performance on a disjunctive concept. Following the same line of reasoning, each of the disjuncts are already exhibiting some negative noise, for each is sometimes omitted from positive examples. Adding more noise to the negative examples does not infirm the predictiveness of the disjuncts any further; the one-directionally noisy situation does not change. However, noise in the positive examples adds the second noise type, weakens the predictiveness of the disjuncts, and makes it more difficult for the Boolean chunking method to add the correct elements.

Examining the asymmetry between Figures 6.7 and 6.8 clarifies the relationship between representation level and noise. Prior to forming an AND for a conjunctive concept, each of the conjuncts appears to be in a noisy situation; afterwards the "noise" disappears. Referring back to our original definition of noise, what happens is not that the examples themselves change, but rather that their descriptions do. This adjustment of representation may occur through an aggregative process (like the Boolean chunking process described in Chapters 4 and 5), through a divisive process to refine the granularity of attribute values, or through the addition of dimensionally new attributes altogether.

6.2.3 Conclusion

Contradictions in the data can be detrimental to concept acquisition, but it need not be fatal. STAGGER's Bayesian weighting and Boolean chunking methods are relatively tolerant of noisy learning situations. A closer analysis indicated that there are two fundamental types of noise, anecdotally exhibited by hypothetical instruments. The one-directional noise is typified by a leaky rain gauge while the two-directional variety is seen with a poor quality thermometer that occasionally reads too high or too low. Statistical theory indicates that the one-directional situation retains information while the other does not. Experiments with STAGGER indicate that it mirrors this distinction and closely approximates the best possible asymptotic performance.

Noise is not the only possible poor quality lurking within a set of examples. Concepts themselves may change over time. For instance, the salient factors for predicting the daily low temperature may change drastically following a large volcanic eruption. What effect does

concept change have on a learning system? In the following section, I raise and answer similar questions through a series of empirical studies of STAGGER's two learning methods.

6.3 Changing concepts

Beyond the potentially poor qualities of noisy values, examples may also be suboptimal due to change in the underlying concepts. Consider the concept of prey in which rabbits (serving as examples) are not completely stable because their coats are brown in summer and white in winter. Changes in word usage may affect relevant retrieval keys for particular documents, adding some and forcing others into disuse. Predictive features of edible mushrooms may change as industrial buildup in a region and changing wind patterns raise the level of acidity in local rain. Congressional party prediction may be strongly affected by a shift in the executive branch from conservative to liberal leadership. In each of these cases, the task of acquiring a concept description may be confounded by the apparent inconsistency of examples. As the concept drifts, previously predictive features may become irrelevant and vice-versa. Such situations are different from the noisy cases described in Subsection 6.2. In the latter there is no concept description that will allow perfect classification of the examples. On the other hand, when a concept drifts, a completely effective description may exist for each of the phases of the concept.

This section seeks to assess the effects of unstable concepts on the acquisition and retention of concept descriptions. First, I examine the functional properties of STAGGER's two learning methods in light of these considerations and then demonstrate robustness through a series of empirical tests. Like the previous section, the results are encouraging, for the combined Bayesian weighting and Boolean chunking methods are able to track a changing concept, efficiently utilize prior learning during change, and distinguish between intermittent noise in the concept and true drift.

6.3.1 Methods and drift

STAGGER's Bayesian weighting method readily adapts to concept changes, for the numerical underpinning of the weights is relatively flexible. As previously predictive features fall into irrelevance, old counts are overcome by new correlations, allowing the weights to return to irrelevant values. Similarly, elements that are now predictive override previously accrued counts through new patterns of matching in examples.

The Boolean chunking method is also able to cope relatively well with concept changes. The formation of newly effective chunks is facilitated by the adaptive properties of the weights, allowing previously ineffective elements benefiting from the concept change to enter into new combinations via the election heuristic (Table 4.3). Of primary concern is the outcome of

large, previously effective chunks when concept drift renders them ineffective. If a complex concept changes drastically, then new chunks must be formed, and there is no significant role for the previous compound elements to play. However, if the drift is slight, how will the effective portion of the previous series of proposals and inaugurations be retained? Here the backtracking mechanism comes to the rescue. By definition, a complex chunk encoding most of the concept description will become less predictive following drift, and at some point it will fall below its inaugurating threshold. Backtracking is called to examine the predictiveness of each of the embedded elements. By comparing these element's current weights to their previous values, this process extracts the unpredictive ones, leaving a simpler chunk that retains the reusable portions of the chunk search. This has the desirable side-effect that a maximal amount of prior learning is retained as the concept drifts, and through retention, speeding subsequent recovery.

6.3.2 Empirical results

With these intuitions in hand, consider STAGGER's ability to track a drift in the concept's definition. The domain I use for this study contains ten attributes of three values each. Each test concept consists of a particular target value for each of five attributes. The disjunctive concept is defined as any one of these five attribute-values, and the conjunctive concept is true only for all five. For both test concepts, the probability of an example is approximately equal to that of a nonexample.

As an initial assessment of the effects of drift, STAGGER was trained on a conjunctive concept of two attribute-values (chosen from a domain of five trinary-valued attributes). After it was able to correctly classify 50 examples, the concept was changed so that one of the previously effective conjuncts was now irrelevant, and an irrelevant attribute-value pair now took its place. This train and change cycle was repeated a second time, resulting in three acquisitions and two recoveries. In each of ten executions, STAGGER formed the appropriate conjunctive chunk (as well as other redundant chunks), backtracking simplified it following concept drift, and the Boolean chunking method reconstructed an appropriate chunk. Figure 6.9 depicts a series of acquisition and recovery curves corresponding to a typical execution over a pair of small drifts in a disjunctive concept. Acquisition and recovery curves for disjunctive concepts are similar.

Note that in Figure 6.9, the concept drift was arranged to coincide with the point at which STAGGER formed an effective concept description. At one extreme, if the concept drifted before STAGGER had formed any description, then recovery would be trivial since there would be no prior training to overcome. At the other extreme, a large amount of experience with the prior concept, STAGGER would be likely to exhibit increasing reluctance to release the prior concept description in favor of new concept regularity, primarily because the counts underlying

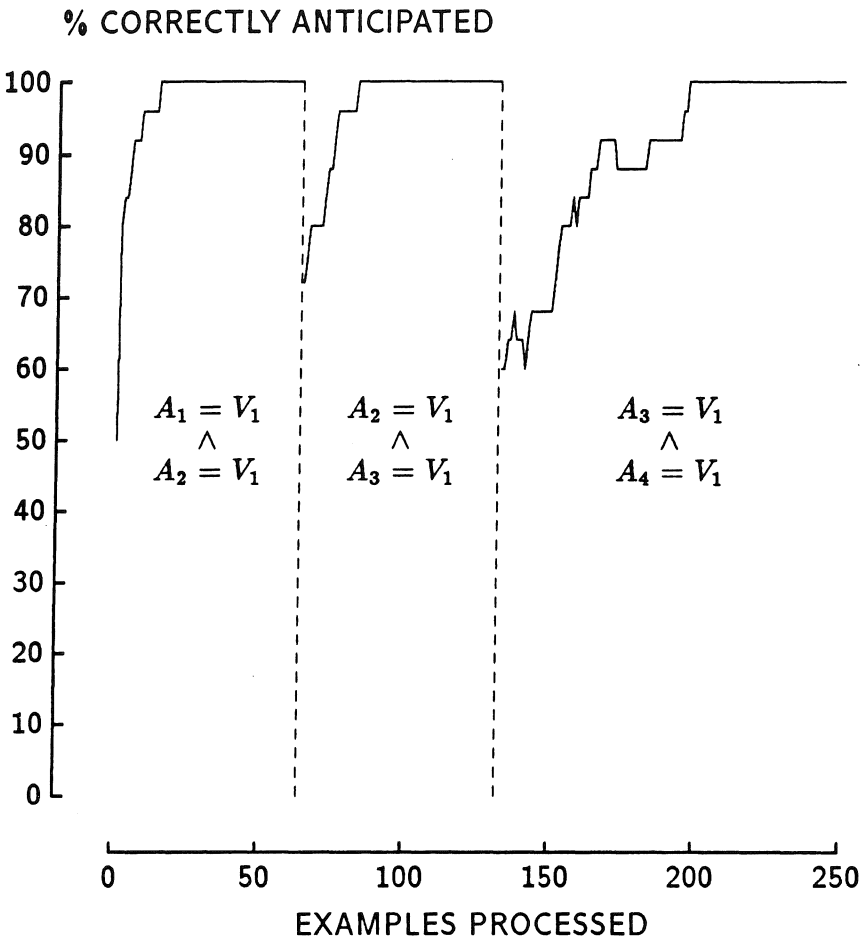

Figure 6.9: Recovering from concept drift.

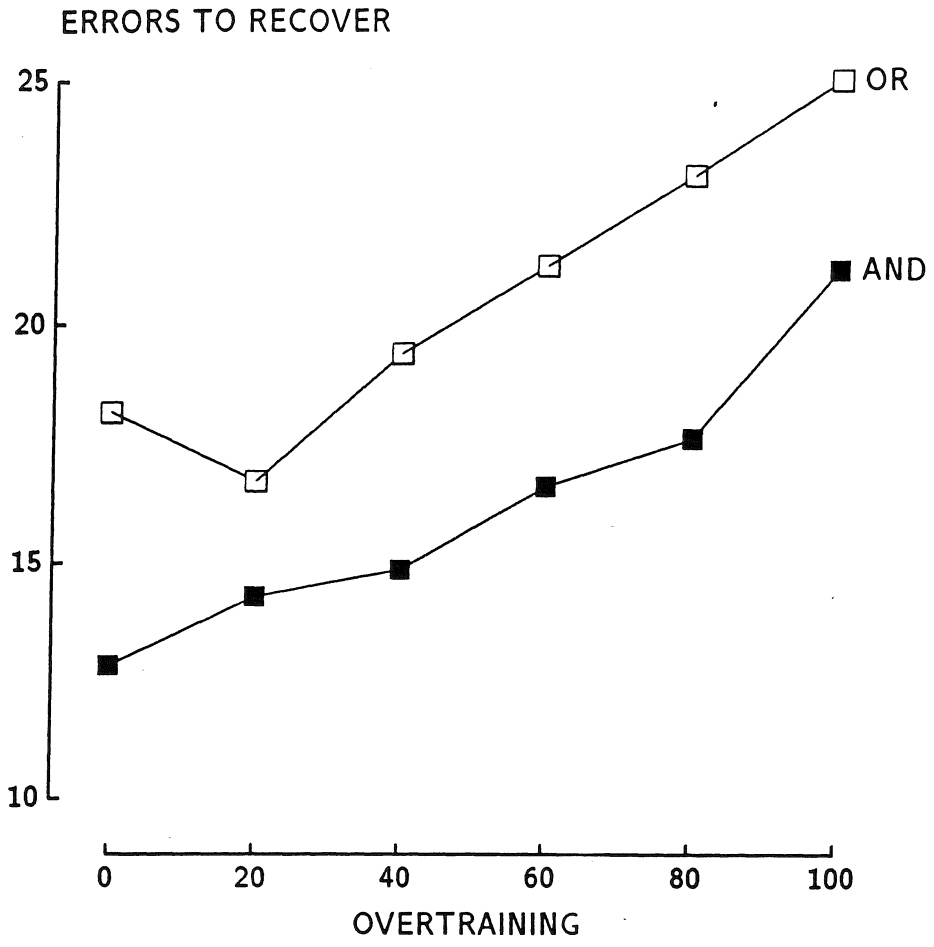

Figure 6.10: Recovery speed as a function of overtraining.

element weights would have the opportunity to become quite large and therefore resistant change. Figure 6.10 plots the results of an experiment designed to test this latter hypothesis: that STAGGER would become increasingly reluctant to adopt a new concept definition with increasing experience.

In this experimental procedure, STAGGER was first trained to criteria on either the conjunctive or disjunctive concept specified above. Then an additional number of examples of this concept were processed, leading to a quantitative increase in experience. Following this, the concept was changed slightly, replacing one of the two conjuncts or disjuncts, and the number of classification errors made until criterion were measured. This process was repeated 20 times for the overtraining amounts of 0, 20, 40, 60, 80, and 100 examples. The resulting behavior confirms the hypothesis; STAGGER reluctantly changes a successful concept description if it has a good track record. This is an implicit heuristic that allows withstanding small, local inconsistencies in the data.

The initial experimental results shown in Figure 6.9 also exemplify performance when the

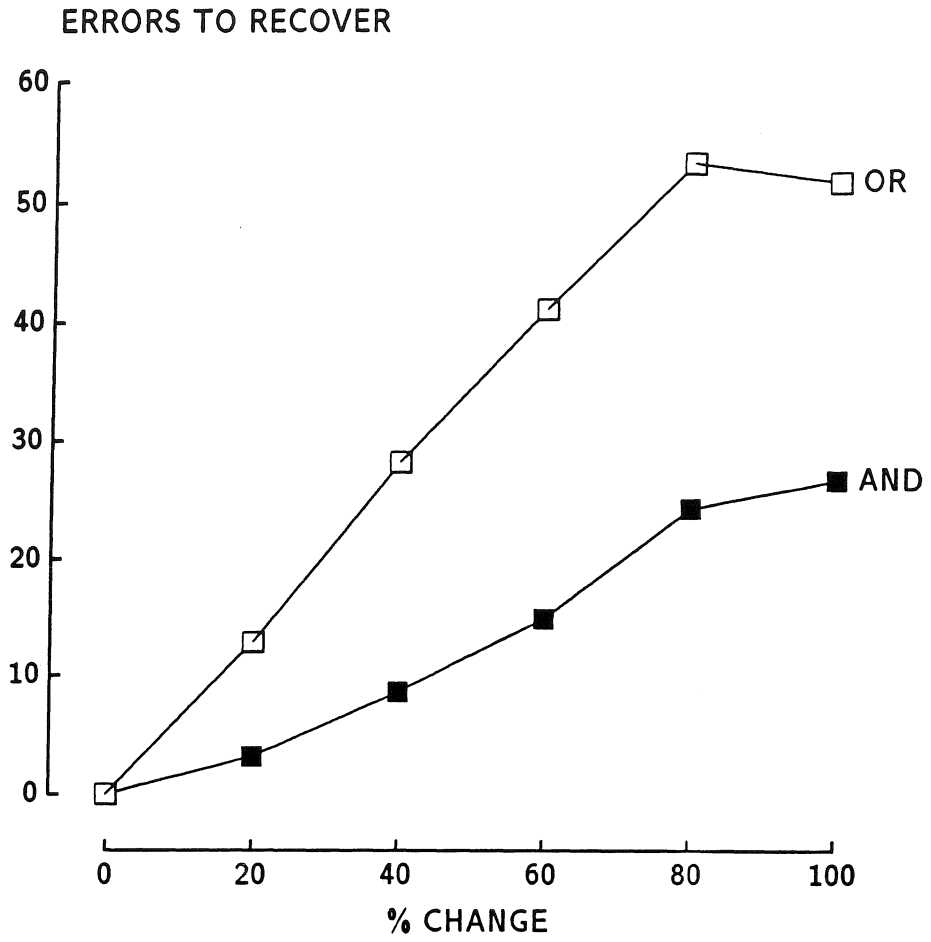

Figure 6.11: Recovery speed as a function of degree of drift.

concept drifts slightly. STAGGER should also be able to recover from drastic changes, though more slowly. To determine if this is the case, a third study was formulated using a larger domain: ten trinary-valued attributes. The conjuncts and disjuncts for the example concepts were defined as five particular attribute-value pairs. After training STAGGER to criterion on the conjunctive or disjunctive concept, 1, 2, 3, 4, or all 5 of the conjuncts or disjuncts were replaced with previously irrelevant attribute-values. Figure 6.11 summarizes the results averaged over 10 executions.

The smoothness of the resulting curves speak well of STAGGER's ability to reuse previous learning as the concept changes. With a small change in the concept, few examples are misclassified until performance is complete. However, larger changes necessitate a greater change in the underlying concept description, and therefore more examples are misclassified on the way to stable performance. This retention of experience is facilitated through the partial preservation of effective chunks. Large, previously effective chunks are broken up by the dependency-directed backtracking mechanism. It individually examines the subcomponents

of an ailing element and introduces a new one without any of the ill components. If a concept changes only slightly, then large effective chunks can, for the most part, be retained with only minor modifications. On the other hand, as the increase in errors indicate (Figure 6.11), larger changes contradict prior experience, and thus backtracking can do little to salvage the current chunks.

An alternative strategy for tracking concept drift would be to discard the concept definition the moment an inconsistent example was processed. Besides being unable to handle noise, this strategy would preclude reusing any previous learning. When the concept changes drastically, most of the inductive work must be redone. However, if the concept changes by a small amount, and only a small portion of the concept description requires revision, this technique necessitates complete relearning. A flat recovery is typical of methods that cannot reverse modifications made to the concept description (e.g., [Mitchell, 1982]). Without the ability to undo prior learning, a method is doomed to poor recovery given slight changes in the concept definition over time.

As a refinement, it might be possible to save the prior concept definition (and initialize to a new one) on the chance that it might prove effective in the future. This approach has the advantage that if the concept cycles between a small number of definitions, recovery could be efficient and depend only on the number of definitions, not on either prior experience or concept complexity.

Finally, to bolster the claim that there is a fundamental distinction between noise and a change in a concept's definition, STAGGER was trained on a the initial disjunctive concept of Figure 6.9 until 50 consecutive examples were processed correctly. Then, instead of changing the underlying definition, examples of the concept were subjected to a 25% uniform noise level. Figure 6.12 plots the resulting performance. Note that following the introduction of noise, performance when tested on noise-free examples remains high. The inconsistency of the noisy examples does not lead STAGGER to abandon a previous concept description as concept drift does.

This behavior arises because the weights assigned to effective chunks are not readily affected by noisy data. As Section 6.2 indicates, the Bayesian weighting method is able to assign appropriate predictive strengths to elements even if they are subjected to noise. Concept noise after effective training is certainly no more infirming than it is prior to training; in fact, it should be tolerated more easily with an effective description already in place. Conversely, when the concept drifts, chunks consistent with the prior definition and not the new one are subjected to a new correlational situation. The counts for these chunks come to reflect the new regularity in the environment, and their LS and LN weights begin to indicate this change. When the chunk weights fall below the inaugurating threshold, dependency-directed backtracking is invoked in an attempt to salvage as much of the chunk as possible.

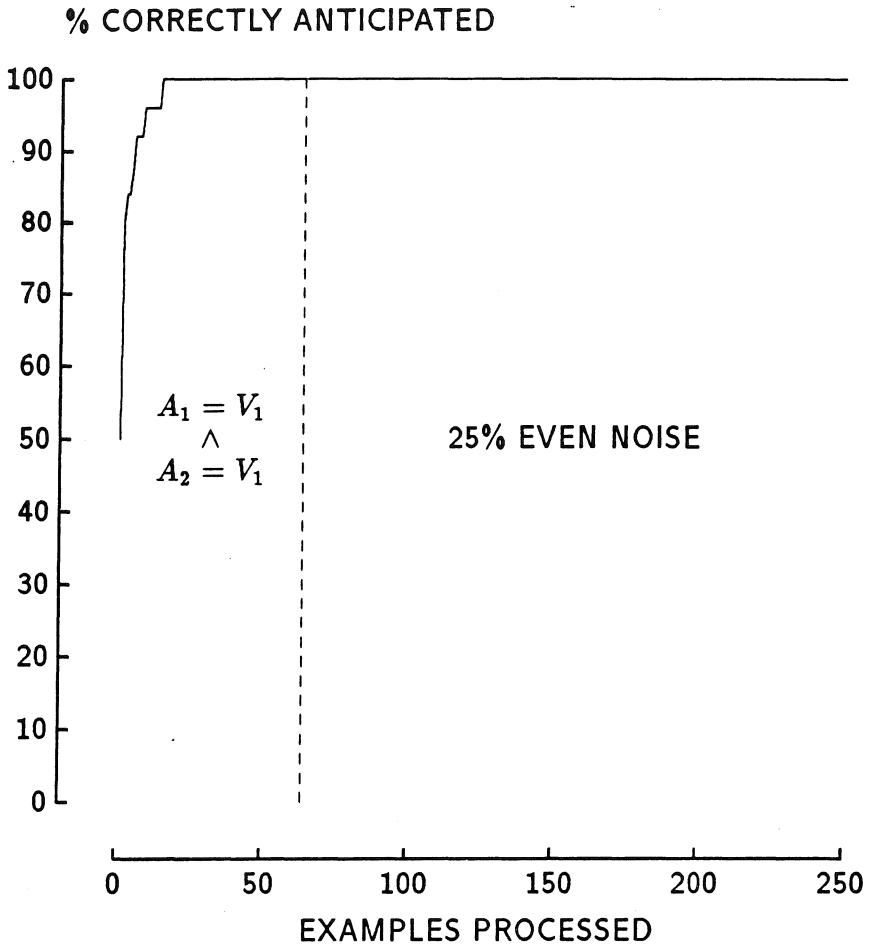

Figure 6.12: Maintenance of performance given noisy data.

Noise and unstable concept definitions are two of the stronger forms of example degradation. However, weaker types are also prevalent, and to round out this discussion of STAGGER's robustness, one of these is examined in the following section.

6.4 Unknown values

In addition to noisy data and concepts that change over time, example values may be unspecified. Though a weaker type of degradation, unknown values are nonetheless prevalent. Examples may be underspecified for a number of reasons. In the Congressional domain, Representatives do not vote on legislation because doing so would entail a conflict of interests, because logistics and travel prevent attendance, or because of the perceived political entanglements of siding with a controversial issue. Other domains incur their own reasons for underspecified examples. In a medical domain risk of a diagnostic test for a patient may prevent its assessment. The financial expenses of market evaluation undoubtedly limits the completeness of data used to assess new product viability.

6.4.1 Learning with incomplete examples

STAGGER's distributed representation allows it to make the most of available data in learning. Basically, each element in the distributed representation acquires its weights independently of the others. This parallel utilization of available information means that if a particular attribute-value is unknown, only those elements that reference this attribute-value are unable to update their predictive weights. Weights for those that do not depend on the unknown attribute are able updated in the normal manner.

Some complex chunks are able to update their weights even though they reference unknown attributes. If an unspecified attribute is only a disjunct, and the chunk is already matched by specified attribute-values, then the absence of this feature does not affect matching or weight update. Similarly, if the unknown value is a conjunct within a chunk, and the chunk fails to match based on the values of known attributes, the chunk's weights can also be updated normally.

6.4.2 Classifying with incomplete examples

STAGGER's partial matching allows it to make the most of available data in classification. Given that the values of some attributes are unknown, this means that STAGGER will be unable to determine the matching status of some of the concept description elements. Consequently, the weights of these elements do not directly modify expectation through Equation 2.3. Their

weights are instead used to adjust the range of expectation, with the smaller of the two weights modifying the lower bound on expectation and the larger of the two raising the upper.

However, there will be some elements that do not reference the unknown values. The weights of these, along with the prior odds of a positive example, will be used to form an expectation of the class of the example, albeit a less informed one. An example of this in action is presented in Section 5.4. In the limit where all of the values of an example are unknown, the result of matching is based entirely on the prior odds of a positive example. The resulting large expectation range indicates a poorly informed decision.

6.4.3 Experimental results

To assess the effectiveness of STAGGER's methods in dealing with unknown values, a pair of experiments was devised in which the dependency between the quality of acquired concept descriptions and the level of unspecified example values could be directly observed. For these experiments, an artificial domain similar to the one in Section 6.2 was used. In each study, example values were randomly deleted according to some preset percentage. After training STAGGER on these examples, concept description quality was assessed by classifying completely specified examples. The process was repeated 10 times for 0, 20, 40, 60, 80, 95, and 100 percent unknown values.

Figure 6.13 indicates that STAGGER is able to construct effective descriptions of a simple conjunctive concept throughout a range of possible underspecified conditions. Specifically, it indicates that though trained on varying percentages of unknown example values, when tested on completely specified examples, asymptotic classification performance remains high up to the point where all of the values are unknown. The flatness of this curve and its abrupt drop closely approximate the theoretically optimum performance.

STAGGER exhibits the same type of classification performance on a simple disjunctive concept. Figure 6.14 shows classification on completely specified examples after training with varying amounts of underspecified example values. This curve is slightly shallower and a bit further from the optimum. Nevertheless, it indicates that the concept descriptions constructed are effective in all situations but those where every example value is unknown.

Unknown values are not without effect on the learning methods within STAGGER. As the discussion at the beginning of this section indicates, the acquisition of a concept description is slowed by increasingly larger amounts of underspecified example values. Figure 6.15 portrays the learning rates for three degrees of unknown values.[4] The quality of intermediate concept descriptions were tested by classifying completely specified examples.

Figure 6.16 indicates that this effect also is exhibited during acquisition of a simple dis-

[4]Each point in Figures 6.15 and 6.16 is an average over 10 executions.

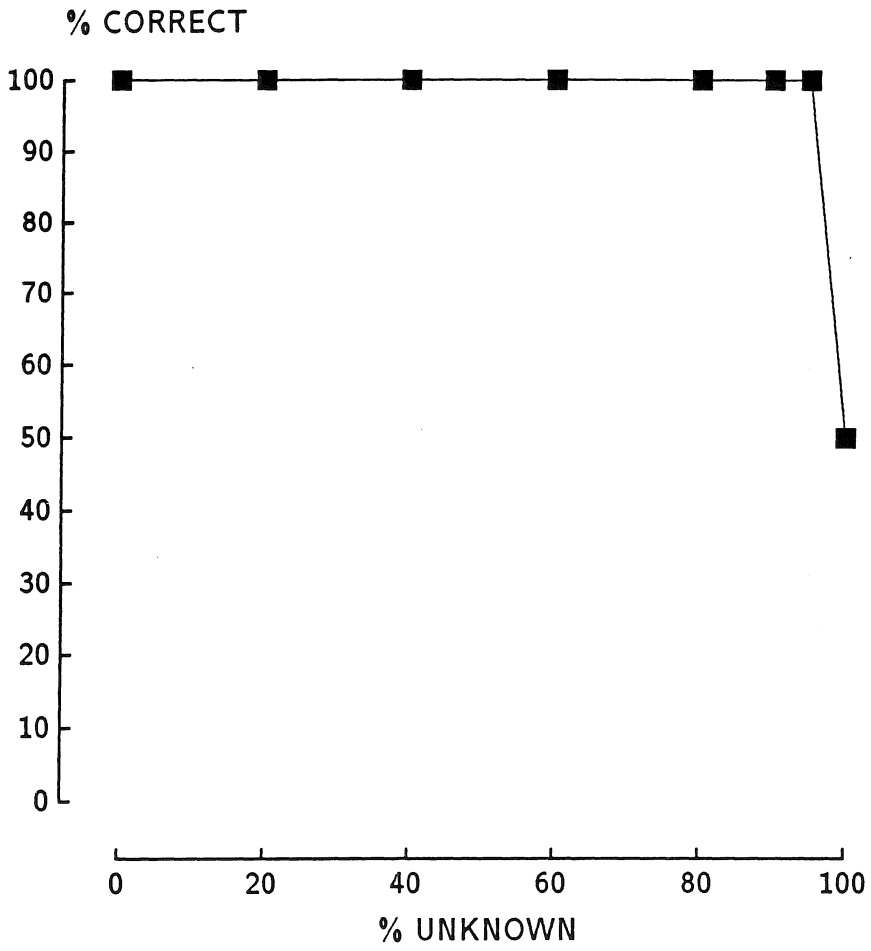

Figure 6.13: The effect of unknown values on acquisition of a conjunctive concept.

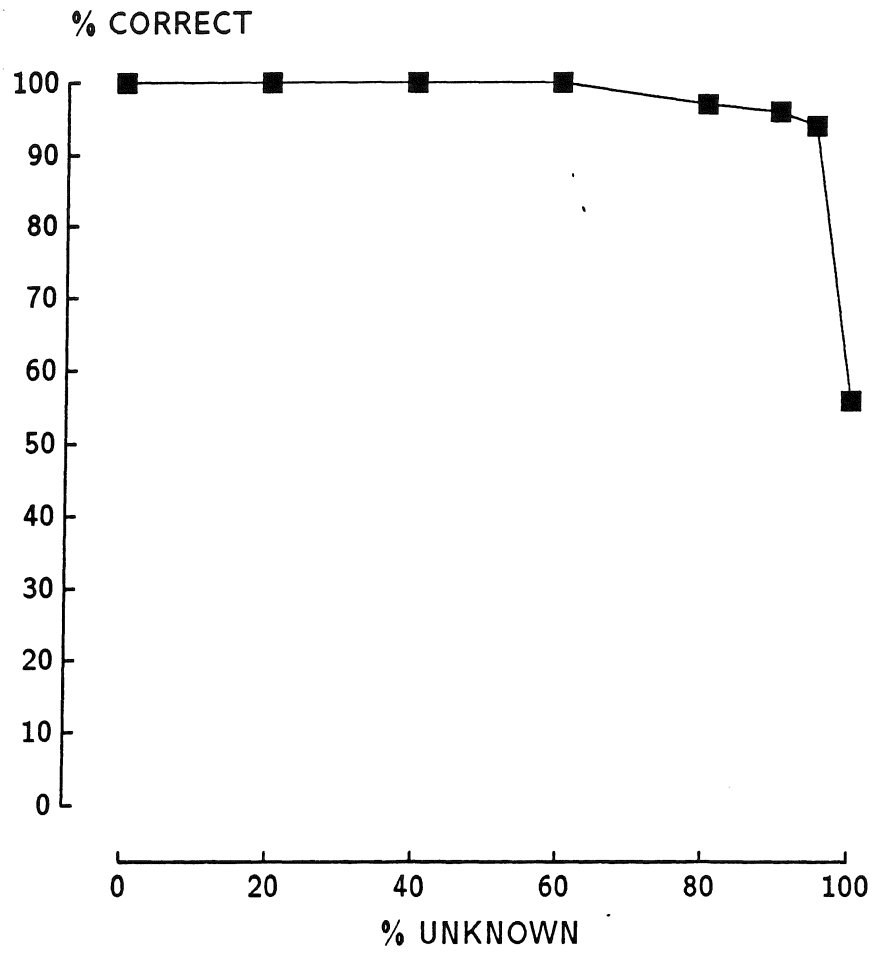

Figure 6.14: The effect of unknown values on acquisition of a disjunctive concept.

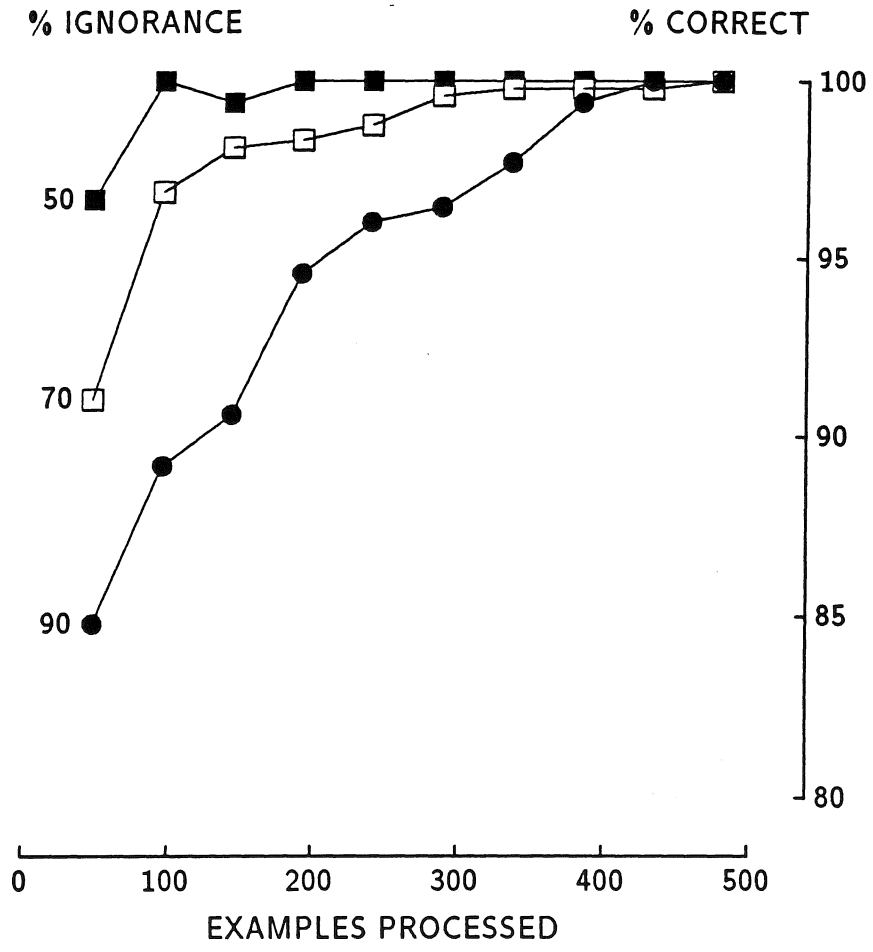

Figure 6.15: Unknown values slow learning of conjunctive concepts.

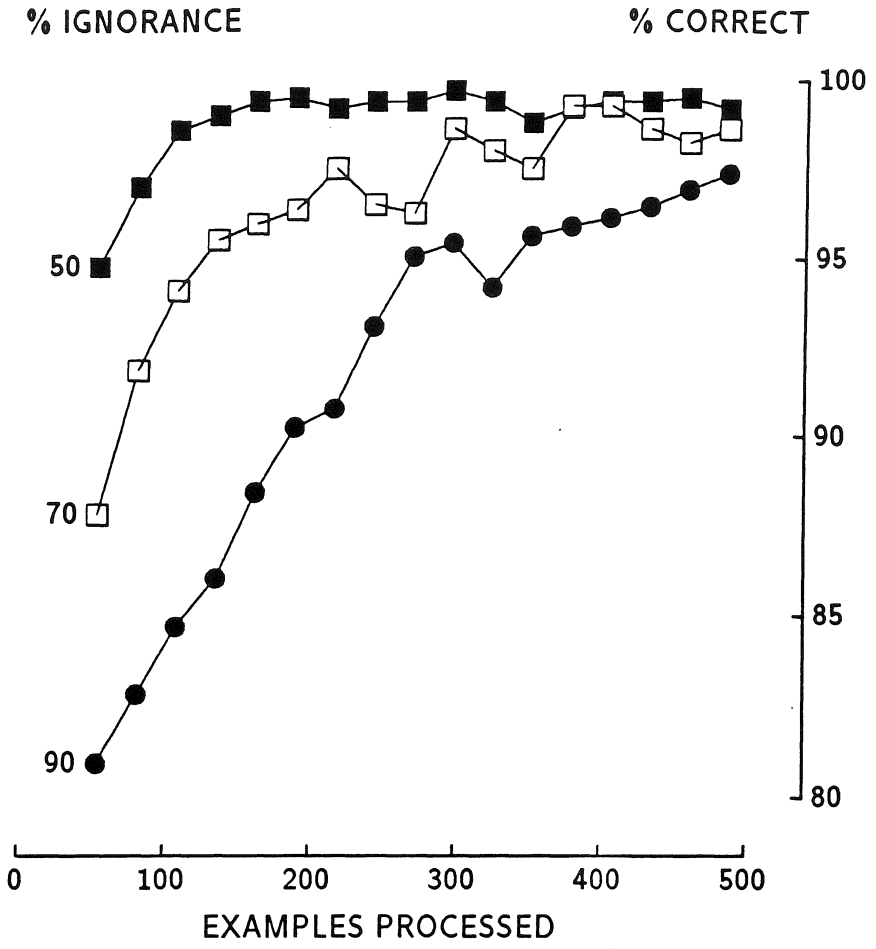

Figure 6.16: Unknown values slow learning of disjunctive concepts.

junctive concept. Learning is slowed because when an attribute-value is unknown, the weight method is unable to adjust the LS and LN weights for the corresponding elements. Similarly, the Boolean chunking method relies on an ability to match elements with examples in its nomination heuristic (Table 4.2), so unknown values delay the introduction of the relevant chunks.

The random assignment of unknown values throughout these experiments ensures that over time the relevant attribute-values will be specified, and thus that the weight and Boolean methods will be able to perform their work. The same general robustness holds even if the distribution of known values is not even. If an irrelevant attribute's value is consistently unspecified, there is no effect on learning, for STAGGER would functionally behave as if that attribute did not have any representation in the concept description. On the other hand, if the value of a relevant attribute were typically unknown, then the examples give no indication of the concept's form and any method could do no better than guess.

The presence of unknown values is only one type of data degradation. The values within

examples may not only be unknown, they may be incorrect. The previous two sections have discussed somewhat stronger forms of example degradation, with the conclusion that combining Bayesian weighting and Boolean chunking result in a robust learning method.

6.5 Overview

The concept learning tasks studied in Chapter 5 illustrate two forms of artifact in concept examples: unknown values and contradictory data. This chapter has explored these two ill qualities of examples as well as a third confounding possibility that the concept may change over time. In each case, an analysis of the operation of the learning methods in STAGGER indicated tolerance of these characteristics, a conclusion that is confirmed by a series of empirical investigations of the implemented system's behavior. The net effect is that if methods like those employed by STAGGER are used, system engineers do not need to be overly concerned by the possibilities that not all example values are known, that the instruments responsible for measurements are completely accurate, or that the concept itself is completely stable.

The behavioral properties studied in this chapter have an emergent flavor. This is especially the case with STAGGER's tolerance of concept drift, for there is no module to explicitly detect a change in the concept, nor is there a specific procedure for recovering from it. Instead the natural operation of the weight adjusting method and size limiting heuristics yield effective behavior. This is not to say that there is no use for explicit, high-level methods within these tasks. For instance, there is adequate conceptual room for the addition of a method to save away concept descriptions after a drastic concept drift. Following the scientific preference for simple explanations, STAGGER is an attempt to understand the capabilities of a pair of relatively low-level methods. Where there are deficiencies, there is room for higher level, knowledge-intensive methods.

Chapter 7

Psychological Constraints and Implications

7.1 Introduction

There are a number of candidate mechanisms that can be used in the construction of any model of intelligent behavior, and it is the researcher's responsibility to form a set of plausible reasons for choosing between them. In some cases, the choice of a mechanism is driven by a preference for simplicity, or by apparent harmony with other parts of the model. More often, these qualities are not readily apparent, and the researcher must rely on other properties to favor one approach over another.

In STAGGER I have used two specific criteria for restricting the candidate mechanisms. One restriction arises by considering the underlying computational complexity of potential processes. For instance, given two processes that search the space of possible chunks, this criterion prefers one that uses a bounded amount of memory over one that requires exponentially large storage. A second source of constraint on the set of possible mechanisms arises from psychological considerations; these are the focus of this chapter. For instance, there are an infinite number of scoring measures that may be used to assess the effectiveness of new chunks. For STAGGER, choosing between the measures is done by comparing their predictions with the results of psychological experiments.

This concern with psychological findings may indicate that STAGGER intends to be a model of human or animal learning. This is not strictly the case. Rather, it is an attempt to model efficient learning in a manner consistent with main psychological results and known computational limitations. A model that solely focused on psychological properties might rightly ignore a number of computational issues, while conversely, many computational models address their tasks effectively with little attention paid psychological plausibility. This middle position is largely dictated by the use of the two criteria of computational and psychological

plausibility; STAGGER does not attempt to explain psychological results at the expense of computational viability, nor does it attempt to be an efficient model that bears no resemblance to natural intelligence.

As the criteria for selecting possible mechanisms are twofold, so are the implications resulting from the model's behavior. Studying the methods may indicate that there are inevitable computational limitations and that an alternative approach would be useful. The inability of the Bayesian weight method to acquire exclusive-or concept descriptions is an example of a computational implication which led to the construction of the Boolean chunking method. Similarly, the model may exhibit interesting behavior in situations that have yet to be studied in psychological experiments. The presence of an inspectable model lends additional transparency to the mechanisms underlying the predictions.

In this chapter I focus on the second constraint on acceptable methods within STAGGER: psychological plausibility. The next two sections present direct experimental results and then indicate their constraint on the formulation of the Bayesian weight adjusting method. Alternatively, the third section describes some implications for psychological studies arising from the operation of the Boolean chunk learning method. Experimental results will undoubtedly complete the cycle by constraining possible methods for combining features.

7.2 Constraints from contingency

There are many ways that STAGGER could evaluate the predictiveness of elements. A set of results from experimental psychology constrain the possibilities. These results indicate that human and animal subjects consider a feature predictive of an outcome only when $p(O|F) > p(O|\neg F)$. As a constraint, this behavioral statement eliminates several measures from consideration, leaving only those that reflect the notion of statistical dependence as candidates. The weights used by STAGGER were chosen for this property; as Section 6.2 indicates, they mirror this constraint.

7.2.1 Contingency results

The conditioning paradigm provides a close analogy to the task of learning from examples, and thus it serves a a potential source of constraint. In conditioning experiments, the ability of subjects to associate previously uncorrelated features is measured. Typically, some innately salient stimulus is used that invokes an involuntary reaction. For instance, a small puff of air may be used since it causes a subject's eye to blink. Other novel, meaningless features are introduced as potential predictors of the innate stimulus (e.g., short tones, buzzers). Experimenters are able to assess the degree to which subjects attribute predictiveness to novel cues by presenting the isolated cues and observing the subject's tendency to exhibit the involuntary

response initially reserved for the innate stimulus. (E.g., How often do subjects blink after the tone?) Varying presentation conditions allows experimenters to determine the precise conditions under which subjects attribute predictiveness.

This conditioning setup is quite similar to learning from examples. The novel cue corresponds to the features of examples, and the presence of the innately salient stimulus maps into identification of this example as a positive one. The binary feedback, and the range of possible cue combinations make for a relatively close fit between the two tasks. This correspondence is developed more fully by Granger and Schlimmer (1986).

Using the basic conditioning setup, researchers discovered that subjects will consider a novel cue to be predictive of an innate stimulus even if the two do not always co-occur. Specifically, given that the novel cue precedes the innate stimulus, subjects still exhibit a response to the novel cue even if it frequently occurs alone (Fitzgerald, 1963). Because the novel cue is only sometimes followed by the innate, reinforcing stimulus, this experimental condition is termed *partial reinforcement.* However, if the innate stimulus is sometimes presented without a preceding presentation of the novel cue, subjects do not exhibit any associative learning between the two.

Given that subjects learn with isolated occurrences of the novel cue, but not with isolated presentations of both the novel cue and the innate stimulus, Rescorla (1968) set out to discover the precise effect of the isolated presentations of the innate stimulus. He devised a series of experiments in which the amount of novel cue overpresentation could be varied independently of the number of isolated presentations of the innate stimulus. The experiments confirmed the findings of partial reinforcement studies, for subjects readily identified novel cues as predictive of the innate stimulus even though the former occurred frequently in isolation. Moreover, subjects failed to learn the association when both the cue and the stimulus were presented separately. These experiments led Rescorla to form a mathematical hypothesis for a necessary constraint on the formation of an association: the probability of the innate stimulus must be greater given the novel cue than it is without (see Equation 7.1).

$$p(\text{innate stimulus}|\text{novel cue}) > p(\text{innate stimulus}|\text{no novel cue}) \tag{7.1}$$

This condition on learning is only a necessary one. There are many other factors that can prevent subjects from forming a predictive association between the cue and stimulus. For instance, if the stimulus already has a highly predictive cue, subjects attribute little predictiveness to novel, redundant cues (Kamin, 1969).

A logical test of Equation 7.1 is the situation where the novel cue always signals a presentation of the innate stimulus, but the stimulus sometimes occurs without the novel cue. Mathematically, the conditional probability inequality holds, so Equation 7.1 predicts that subjects should assign predictiveness to the novel cue in this situation as they do in partial

reinforcement cases. Granger and Schlimmer (1986) have proposed a set of experiments to test this hypothesis and have termed the condition *partial warning* since the stimulus is only partially warned by the novel cue. To date, these experiments have not yet been run.

Roughly speaking, these experiments define four classes of conditioning situations. In the first, neither the novel cue nor the innate stimulus are presented in isolation of the other. A second situation, partial reinforcement, the cue is presented in isolation along with joint presentations of the cue and stimulus. The third situation involves joint presentations and isolated occurrences of the innate stimulus; it is termed partial warning. Lastly, in the fourth situation, joint presentations are accompanied by independent occurrences of both the cue and stimulus. Equation 7.1 predicts learning in the first three situations, and this has been confirmed for all but the third. Even in mild versions of the fourth situation, subjects fail to consider the novel cue a predictor of the innate stimulus.

These four situations are those discussed in Section 6.2, where the surface corresponding to statistical independence is described (Figure 6.3). When the two conditional probabilities in Equation 7.1 are equal, an association is not formed by subjects between the cue and stimulus, and this corresponds directly to statistical independence between the cue and stimulus (Schlimmer, 1986). To make the mapping from the noisy situations described in Section 6.2 and the four conditioning situations, simply consider the novel cue as the reading of a potentially faulty instrument and the innate stimulus as the true measurement. If the instrument exhibits one-directional noise after the manner of the smoke alarm, then it is acting like the novel cue in the partial reinforcement conditioning experiment. The alarm's sound is only partially reinforced by the actual presence of a fire. Conversely, a one-directional noisy instrument like a rain gauge acts like the novel cue in the partial warning experiment, for by leaking the gauge only occasionally reflects the true precipitation. Both of these instruments yield information about their intended measurement, even though they are faulty. However, the two-directional noisy instrument does not convey much information. For instance, an inaccurate temperature gauge sometimes incorrectly indicates overheating and sometimes fails to indicate this. It corresponds to a conditioning situation with both independent presentations of both the novel cue and the innate stimulus.

7.2.2 Conformance

As Section 6.2 argues, the LS and LN measures used by the weight method reflect the notion of statistical dependence and thus the constraint of Equation 7.1. This correspondence is by design, for there are many weights that could be used by STAGGER for this purpose. A simple alternative would be to increase the weight of the element every time it matched correctly and decrease it every time it did not, yet this simple weight would not yield the same independence surface seen in Figure 6.3.

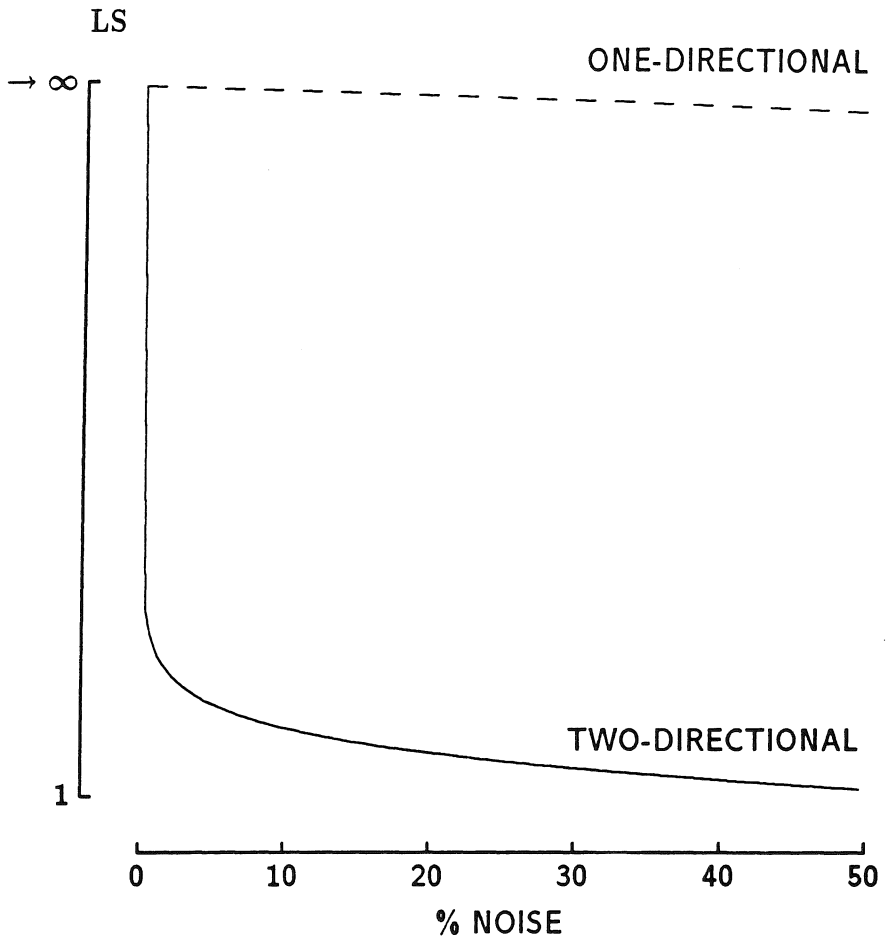

Figure 7.1: Differential ratings of one- and two-directional noisy situations.

To provide further evidence of LS and LN's effective evaluation of noisy situations, consider Figure 7.1 which depicts the relative ratings of one- and two-directional noisy situations by LS. Progressing along the horizontal axis, a greater percentage of examples are randomly identified as positive or negative. At the extreme right, half of the examples are randomly identified as positive or negative examples in the case depicted by the lower line. Similarly, in the case represented by the upper line all of the negative examples are identified randomly also amounting to half of the examples.

Note that a one-directional situation is rated nearly as well as a no noise situation, reflecting the fact that information is preserved. Conversely, a small amount of mixed noise is rated much more poorly. One result of these differential ratings is depicted in Figure 6.7 of Chapter 6. That experiment showed that even though all of the positive examples were noisy, STAGGER was still able to form an effective concept description. This corresponds to a one-directional noisy situation like partial reinforcement and the upper line in Figure 7.1. However, when both the positive and negative examples are noisy, as Figure 6.7 indicates, STAGGER's performance

drops as would be expected of a two-directional noisy situation and as predicted by the lower line of Figure 7.1.

Unlike the Bayesian weight method, the construction of the Boolean chunking method is not constrained by the experimental results summarized by Equation 7.1. Principally this is because these results are concerned with the unary predictiveness of individual cues whereas the chunk learning method focuses on the construction of predictive combinations of features. There are experimental results that have implications for this method's construction, and they are the focus of Section 7.4.

The conditioning experimental literature yields other constraints on the functioning of possible weighting methods. The next section focuses on a relatively intuitive set of experimental findings which indicate that learning is affected by prior experience.

7.3 Constraints from prior experience

The previous section described how results arising from partial reinforcement experiments constrain the class of acceptable weighting measures. This section is similar, for in it I describe another set of experimental results that constrain the Bayesian weight method's formulation. Specifically, the results clearly indicate that prior experience with conditioning cues affects subjects' subsequent willingness to consider them predictive. As a result, the computation underlying LS and LN is based on an accumulation of experience, and thus STAGGER's combined methods are able to replicate the basic experimental results.

7.3.1 Prior training results

Intuitively, prior experience is an undeniable factor in learning. At one extreme, if a conditioning cue already has a successful track record of predicting an innately salient stimulus, then no learning is needed. At the other extreme, where a cue has a history of being negatively predictive, subjects presumably require a significant amount of training to unlearn the cue's prior role and acquire a positive predictiveness.

The results of psychological experiments confirm these intuitions. Siegel and Domjan (1971) investigated a set of five experimental conditions, each varying the type of prior conditioning experience. In the simplest condition, subjects were completely naive with respect to both the novel cue and the innately salient stimulus. Typically, this is the desired state for a subject prior to the beginning of an experiment, and it was included here as a control. Two other conditions preexposed subjects to either the novel cue or the innate stimulus. In a fourth condition, subjects were randomly presented with the cue and stimulus, and in the fifth condition, the cue and stimulus were negatively correlated. Following preexposure, all subjects were given presentations of the cue predicting the occurrence of the stimulus.

The groups of subjects exhibited a differential willingness to consider the cue predictive of the innately salient stimulus. The naive group conditioned the most quickly, followed closely by the cue only, stimulus only, and random groups. The negative correlation group required significantly more time to form a predictive opinion of the initially novel cue. These primary results have been replicated (Baker, 1976; Kremer, 1971; Randich & LoLordo, 1979; Rescorla, 1969).

7.3.2 Conformance

The computation underlying the LS and LN weights was chosen to reflect these effects of prior experience. Recall that in Chapter 2.4, preference was given to the count based method for computing element weights. Briefly, by keeping track of the number of times a concept description element is matched in an example, matched in a nonexample, unmatched in an example, and unmatched in a nonexample, it is possible to compute estimates of the conditional probabilities used in the definitions of the LS and LN weights. As these counts are accrued, they represent an accumulation of experience. The resulting effect that after processing a large number of examples, the weights are less sensitive to the effects of any particular one, an effect depicted in Figure 2.6.

This technique for computing the weights has the desired effect, for STAGGER's methods readily replicate the main findings of Siegel and Domjan (1971). In the cases where subjects are preexposed to either the novel cue, the innate stimulus, or a random correlation between the two, counts for each of the concept description elements grow, storing away their irrelevance for the outcome. Each of these situations should require processing additional examples to overcome the accrued counts as compared with a naive learner. When the cue and stimulus are inversely correlated, the counts also accumulate experience, but in this case, they come to represent a weight that will have to be reversed when conditioning is set on the normal path. More than the others, it will necessitate processing more examples: not simply a matter of imposing a regularity on irrelevant counts, but of reversing the encoding already stored there. This intuitive description of the behavior of the weight method on each of the five conditions studied by Siegel and Domjan is borne out empirically as Figure 7.2 illustrates.

There are five lines in Figure 7.2 corresponding to the five conditions used in (Siegel & Domjan, 1971). The final training concept was a single attribute-value pair chosen from ten trinary-valued attributes. To imitate preexposure to the novel cue, all examples were identified as negative ones. Thus, there were no occurrences of the innate stimulus. For pretraining to the stimulus, negative examples were randomly identified as being positive or negative. This prohibited the introduction of the cue (the disjunction) while allowing for the stimulus. To simulate the random group, examples were randomly identified after the manner used to study acquisition in a completely noisy environment (Section 6.2). Negative correlation was supplied

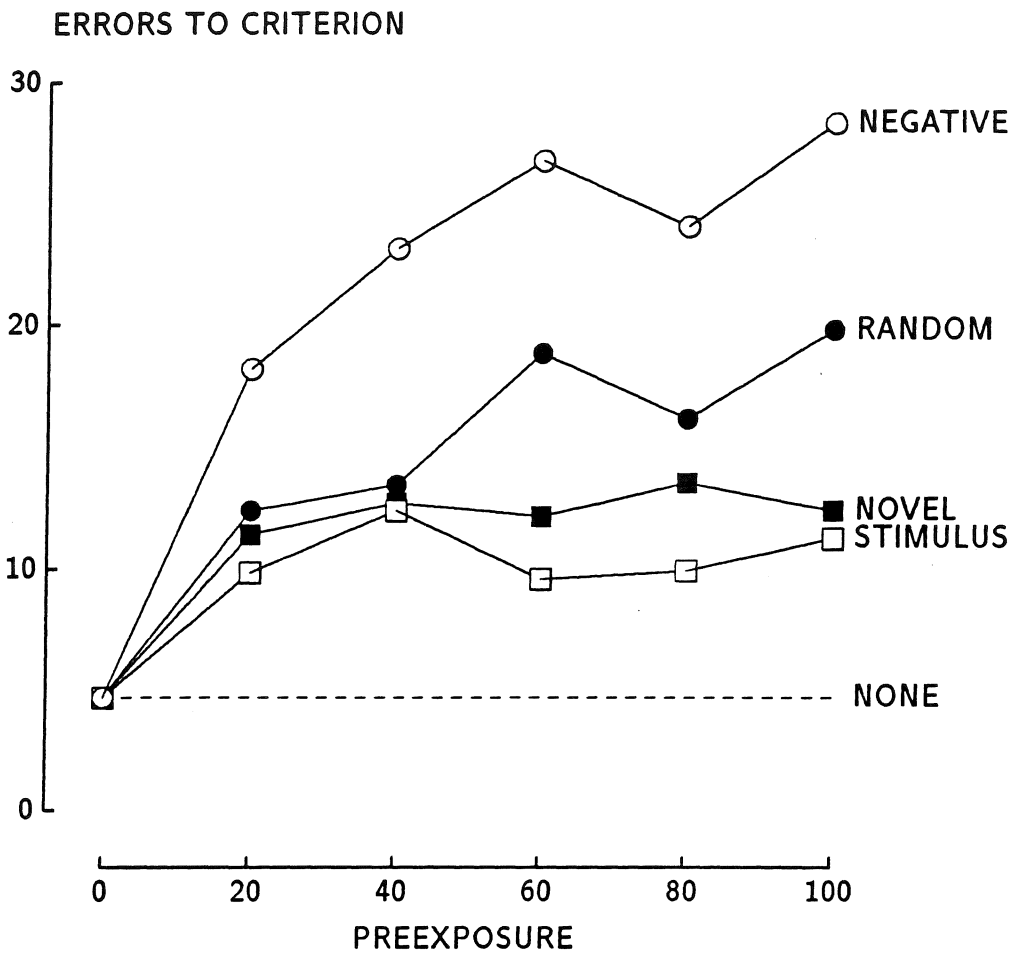

Figure 7.2: Acquisition following preexposure used by (Siegel & Domjan, 1971).

by inverting the identity of examples for preexposure: positive examples became negative and vice-versa. For each case the amount of preexposure was varied from 20, 40, 60, 80, or 100 examples, and STAGGER's performance was averaged over 20 executions. Along the vertical axis are plotted the average number of classification errors made following pretraining prior to correctly classifying 100 examples correctly.

Examining the data after preexposure of 40 examples closely corresponds to the findings of Siegel and Domjan. The novel cue, the innate stimulus, or random pretraining conditions result in some small suppression of subsequent learning, the negatively correlated condition requiring substantially more error recovery time. As preexposure is increased, both the random and negative correlated conditions incur additional recovery in the form of additional errors.

The overtraining effects demonstrated in Section 6.3 may also be seen in Figure 7.2, for it took STAGGER longer to recover from a preexposure of 100 negatively correlated examples than it did from 20. The corresponding degree of drift lies somewhere between the naive condition depicted as a horizontal line and the negatively correlated curve near the top.

Experimental results indicate that prior experience does indeed have a effect on subsequent learning. In nearly every case, learning of a new association is slowed, though more so in some cases than in others. STAGGER's method of computing the LS and LN weights was derived with these results in mind, and therefore it is no surprise that they are accommodated. There are some psychological phenomena that STAGGER was not explicitly designed to mirror. The next section describes a set of conditioning situations and describes STAGGER's behavior in them. The results may be interpreted as a hypothesis about human and animal subjects' abilities in similar situations, and the hope is that the results of experiments will serve as constraints on reformulations of the Boolean chunk learning method.

7.4 Implications for configural learning

This chapter is primarily concerned with demonstrating how psychological results serve to constrain the formulation of methods within STAGGER. The previous two sections serve as examples, for they each have described a set of results and then have shown how the Bayesian weight learning method is different because of them. The first described partial reinforcement and related experiments. The findings there correspond closely to statistical dependence and the results on noisy data presented in Section 6.2. In conformance, the Bayesian method uses LS and LN to weight each description element since these measures mirror that constraint. A second constraint arises from experiments that test the role of prior experience with conditioning cues on subjects' subsequent willingness to assign predictiveness to them. The results are intuitive, and in response, weights are computed as a result of accumulated evidence. This section has a somewhat different flavor; instead of describing a set of psychological results

that constrain STAGGER's formulation, it focuses on a set of implications arising from the combined methods' operation.

Specifically, in Section 5.5 I described the Boolean and weight methods' learning ability with respect to 19 concept types. Each of these concepts is not equally easy for STAGGER, and their difficulties closely correspond to the amount of work required of the Boolean chunk learning method. It should be possible to test subjects' ease at learning each of these 19 concepts. Some concepts involve a single relevant cue (attribute) and two irrelevant cues. Acquisition of these concepts is undoubtedly similar to the simple conditioning experiments explored in the previous two sections. However, some of the more difficult concepts for STAGGER require the ability to assign predictiveness to a combination of features or cues. This type of task has not gone without attention in conditioning research; in fact, the study of subject's assignment of predictiveness to compound cues falls under the heading of *configural conditioning*. While a number of experiments have been run to test learning to different configurations of novel cues, researchers have failed to identify and study a comprehensive set of tasks. The 19 cases identified in Section 3.5 may serve as such a set, and STAGGER's differential ease of acquisition, as a set of predictions.

To set the stage for consideration of the implications of the 19 concepts and STAGGER's performance, I first review a few configural conditioning experiments. Their results bode strongly against current theories of conditioning. After reviewing the 19 concepts, I discuss the possible implications for a set of conditioning experiments and interpretation of possible outcomes.

7.4.1 Simultaneous configural association

Ever since researchers discovered that subjects would associate a novel cue with an innately salient stimulus, they have sought to understand the limitations of this phenomena. One dimension of this task focuses on the potential complexity of the novel cue that comes to be predictive in the mind of the subject. Termed *configural conditioning* because it is the study of the acquisition of configurations of cues, the resulting set of experimental data has proved elusive for a number of learning theories.

For instance, consider an early experiment by Woodbury (1943). In this experiment subjects (dogs) were trained that particular combinations of a pair of buzzing sounds indicated an opportunity to lift a bar and cause a small food pellet to appear. The subjects readily learned that individual buzzing sounds were predictive of the food reward behavior. Moreover, subjects were also able to learn that lifting the bar for the pellet was effective only when both of the buzzers were on. This is a simple demonstration of the subjects' ability to assign predictiveness to a conjunction of cues. Conversely, subjects also learned to respond correctly when the bar lift was effective after one or the other of the buzzers but not both. Thus subjects

Table 7.1: Saavedra (1975) experimental conditions.

INNATE STIMULUS	NO STIMULUS
A_1L_1	A_1L_2
A_2L_2	A_2L_1

appear able to assign predictiveness to an exclusive-or configuration of features.[1]

Saavedra (1975) has also studied configural conditioning. However, unlike Woodbury's experiments, in each configuration all of the novel cues were from different sensory modalities. Instead of two cues, four were used in pairwise combinations such that each cue was equally likely to be followed by either the innate stimulus or not. In the primary experimental group, the pairings of a tone (auditory cue A_1) with a flickering light (L_1) and the pairing of a clicker (A_2) with a steady light (L_2) were followed by the innate stimulus. Presentations of the opposite pairwise configurations of A_1 with L_2 and A_2 with L_1 were not followed by the stimulus. Table 7.1 summarizes these experimental conditions.

Subjects readily learn that these configurations predict a presentation of the innate stimulus despite the fact that individual cues are as often followed by the stimulus as not.

Woodbury and Saavedra carefully constructed their experiments to test three potential hypothesis for the mechanisms underlying configural conditioning. First, researchers have hypothesized that the strength of a configuration of cues is the sum of the predictiveness of each of the cues (Rescorla & Wagner, 1972, Mackintosh, 1975). By adjusting the stored estimates of the predictiveness of the cues, these models select the most predictive cues. (For convenience in the following paragraphs, I will refer to this class of models as feature selection models.) A configuration consisting of any of a number of cues (disjunctive) is easily accommodated, for if each cue is predictive, their sum is also. Furthermore, a conjunctive configuration may also be learned; here each cue is only slightly predictive and individually are below threshold. Collectively, they have sufficient predictiveness to emulate a predictive configuration. However, an exclusive-or configuration like the one tested by Woodbury requires that the individual cues have a qualitatively different predictive value than their combination. Using a simple feature selection model it is not possible to find predictive values for the individual cues that are strong enough to carry prediction when only one is present that are not also too strong and incorrectly predict the innate stimulus when both cues are present.

This difficulty is usually finessed by assuming that the co-occurrence of two cues (e.g., a

[1]Woodbury noted a differential learning speed, with subjects acquiring the simple discriminations faster than the compound ones. Though there was not a significant difference between the rate of acquisition for a conjunctive and exclusive-or concepts, such results would have interesting implications for the formulation of the Boolean chunking method used here.

red light and a blue light) results in some new *resonant* property of the cues (a purple light). A feature selection model can then assign a strong predictiveness to each of the individual cues in an exclusive-or configuration and a negating predictive value to the resonant cue arising from their co-occurrence. There are two unsatisfactory consequences of assuming the presence of resonant cues. First, the number of cues from which the feature selection model must select grows exponentially with the number of cues that may be configurally predictive. For instance, if there are three cues (A, B, C), then there must be four supplementary cues (AB, AC, BC, ABC) in order to select any configuration. Razran (1965) reports on experiments where six simultaneous cues were conjunctively configured; 57 resonant features would be required if a feature selection model were to account for this data through the use of resonant cues. In general, the number of resonant cues required by these models is $2^n - 1 - n$, where n is the number of cues. Requiring a feature selection model to choose between $O(2^n)$ cues may be computationally infeasible. Second, while it seems plausible to assume that the simultaneous presence of a red light and a blue light adds a cue not present when either are presented separately (a purple light), this assumption seems tenuous when the cues are from different sensory modalities. Yet, as Saavedra demonstrates, subjects are capable of effectively predicting an innately salient stimulus even when it is predicted by compound cues configured from different sensory modalities.

A third potential saving hypothesis relies on additional discriminatory power arising from some configuration property of "two-ness." As a slight variation on the notion of cue resonance, it might be reasonable to assume that there is some additional cue corresponding to the number of easily discernible cues. If present, this would allow a feature selection model to account for Woodbury's demonstration of exclusive-or configural learning, for each of the singular cues could accrue strong predictive values while the combination of the two could again have a negative predictiveness. However, Saavedra explicitly constructed her experiment so that both the compounds that preceded the innate stimulus and those that did not incorporated the same number of cues. It does not seem reasonable to assume that subjects are using a feature selection model and relying on emergent cues for effective prediction.

Rather than continue to add tenuous hypothesis upon hypothesis, it may be more effective to consider models that form internal representations in order to assign predictiveness to configurations of cues. Razran favors this approach:

> What seems more warranted is the view that, inasmuch as configures are formed and deformed through learning, their role is much more a function of the organism's conditioned past that its sensory present, and, moreover, that their learning reveals the dynamic essence of their "becoming" if not also of their being.[2]

Recent experimental results seem to support this. Rescorla, Grau, and Durlach (1985)

[2](Razran, 1965, p. 244, fn. 3)

have discovered that configural associations appear to based upon internal representations of the paired cues rather than on some perceptual property arising from their co-occurrence.

Both STAGGER and the learning model proposed by Hampson (1983; Hampson & Kibler, 1983) follow this approach. In the latter, a multiple layer feature selection model is used to learn configural predictors. Initial layers are connected directly to perceptible cues and form implicit conjunctions and disjunctions of these cues. The predictive output of these layers serve as inputs to subsequent layers, allowing the final layer to simply select among the internally configured features. Details of this model and its relationship to STAGGER are discussed more fully in Section 8.5.1.

STAGGER is also a hybrid feature selection and formation model. The weight adjusting method is functionally quite similar to a strict feature formation model, and it is limited in precisely the same manner, as Section 3.5 describes. Its representational limitations are alleviated by the cooperative action of the Boolean chunk formation method which adds new internal representations of its inputs.

7.4.2 Implications

STAGGER's combined weight and chunk methods are capable of acquiring predictive configurations of cues. The Boolean chunking method forms internal representations, or subconfigurations, so that the Bayesian weight method may simply select among them. Figure 5.9 indicates the sufficiency of these combined methods for any configuration of three or fewer cues. This includes the two experimental conditions studied by Woodbury; additional experimentation with STAGGER indicates that it is also capable of acquiring the configurations used by Saavedra (Schlimmer & Granger, 1986b). However, not all of the configurations are equally difficult. Some require considerably more exposure to concept examples than others. This emergent ordering between easy tasks for STAGGER and difficult ones may serve as a detailed set of predictions. In brief, the hypothesis is that human and animal subjects are using a hybrid feature selection and formation method. Therefore, those configural learning tasks that require the formation of internal representations should take qualitatively longer than those that do not. Just as the set of 19 concepts serve to test the operation of the methods across a range of possible situations, so they may be used to test subjects' behavior across a range of configural conditioning experiments.[3] Those concepts that STAGGER finds more difficult than others should also be more difficult for subjects in conditioning experiments.

Undoubtedly there will be disagreement between the relative ease of particular configurations for subjects and for STAGGER. One interpretation of the discrepancy is that Stagger's framework of two cooperating methods is valid, but that the operators for searching the space

[3]This is not a prediction about absolute learning rates for each of the concepts in Figure 5.9, but rather about the relative rates.

of possible sub-configurations or chunks are not. For instance, it may be that instead of the formation of new AND, OR, or NOT chunks, it would be more psychologically consistent for the Boolean method to form new NAND or NOR chunks. Investigating STAGGER's acquisition of the 19 cases with different sets of operators may either yield a close fit with experimental findings and thereby confirm the original hypothesis, or the inability to find a reasonable agreement may indicate that subjects are not using a hybrid feature selection and formation method.

The differences between experimental evidence and STAGGER's performance serve to identify constraints on the future versions of the Boolean chunk learning method. The increased understanding of subjects' learning capabilities will undoubtedly serve to constrain the formulation of forthcoming feature formation methods.

7.5 Overview

In the construction of any procedure there are several alternative techniques that may be used. Frequently, the designer relies on a set of constraints to aid in the selection of appropriate methods. In artificial intelligence, the twin constraints of computational feasibility and psychological plausibility may be married together with more commonplace criteria like simplicity and harmony to limit the choice of technique. This chapter has explored the heuristic role that a pair of psychological results play in narrowing the space of possible methods within STAGGER. First, I recounted that subjects learn to associate a novel cue with an innately salient stimulus even if the two do not always co-occur; this is clearly demonstrated by partial reinforcement and related experiments. In compliance, STAGGER uses two measures that reflect a mathematical characterization of the behavioral data, and in doing so, is effective at learning in the presence of noisy data.

A second set of experimental results indicated that subjects' learning is influenced by their prior experience. This was assessed by administering different types of pre-exposure to the training materials, and then measuring the differences in subsequent acquisition. STAGGER was designed with this data in mind, and its preference for the count based method of computing LS and LN is a direct result. One outworking of this functional decision is demonstrated in Section 6.3 where STAGGER exhibits an implicit heuristic to retain concept descriptions with an effective track record in the face of change.

These two sections focusing on the constraints provided by psychological results were offset by a section describing some implications for the study of human and animal subjects' behavior. After reviewing two experiments studying subjects' ability to form an association between a combination of cues and an innately salient stimulus, I reconsidered the study in Chapters 3 and 5 that applies STAGGER to a set of 19 concepts. Because these concepts are

constructed from binary-valued attributes, the mapping to configural conditioning is clear: each attribute corresponds to a particular cue, and the concept is a test of a subject's ability to form a configural association. As I argue there, the Boolean chunk learning method is a necessary component, for the Bayesian weight method alone is incapable of tackling many of the 19 tasks. While the identification of this task may be interesting simply because it helps to define a space of possible conditioning situations, STAGGER's differential readiness to acquire each concept can serve as a hypothesis about the capability and ease subjects will show when faced with the same task. Though the experiments have yet to be designed and executed, their results will serve as a constraint on reformulations of the Boolean chunking method.

At this point, the methods in STAGGER have come full circle. The constraints that were heuristically used to select among possible mechanisms have themselves been refined by experience with the choices. Further analysis of potential constraints may yield similar fruit as experimental psychology results are incorporated into the formulation of learning methods and those methods are tested.

Chapter 8

Other Dual-Method Learning Models

8.1 Introduction

In a general sense, other researchers have not relaxed their assumptions about the quality of the input for learning from examples. Consequently, many of the prominent learning systems are unable to handle noisy data, unknown values, or concept drift (e.g., Mitchell, 1982). This is not to say that all previous work is inadequate. In fact, as I argue in this chapter, a few of the existing learning methods could conceivably do quite well on this relaxed task. Many of the noteworthy properties of STAGGER did not result from clever design; rather they emerge from the basic organization of the two, cooperative learning methods. To greater and lesser degrees this type of organization is exhibited by other effective learning methods. In the following sections I present representative examples within each of four general learning paradigms, discussing first the main processes and then their implications for concept acquisition. At the close of this chapter, I flesh out the thesis that each of the methods surveyed exhibits a fundamental similarity based on the bipartite division of labor between a simple, correlation-detecting mechanism and a higher level process that adjusts the representational base for the former. In closing, I review four evidence combination criteria presented in Section 2.3.3 and each method's correspondence to them.

8.2 The Iterative Discriminator

Quinlan (1983, 1986, in press) has developed a learning method that is highly tolerant of noisy data and can be shown to converge on a Boolean concept description if one exists. The basis for the method is a set of statistical assessments, each of which guides the construction of a decision tree. The tree serves as the concept description and is used to classify future examples. This method is named Iterative Discriminator (ID3) because the decision tree is

constructed top down, with each successive level corresponding to finer discriminations.

Initially, ID3 begins with an empty decision tree and a set of examples. A statistic is applied to these examples to determine which attribute conveys the most information about their classes. The most discriminant attribute serves as the root of the decision tree, and its values are arcs descending to yet-to-be-constructed subtrees. For each subtree, the examples are partitioned according to their value for the root attribute. In this manner, the set of examples becomes increasingly smaller as the tree is refined, and the task of discriminating between them becomes easier. In the limit, the decision tree is a full tree, with a test for each attribute's value and a label at each leaf corresponding to an example class. However, the root selection measure tends to uncover regularity inherent in the data, resulting in more compact trees.

The specific evaluation measure is drawn from the field of information theory, and it assesses the expected amount of information conveyed by a single attribute about the class of examples. The gain, or expected increase in information is given by:

$$\begin{aligned} gain(A) &= I(p,n) - E(A) \\ I(p,n) &= -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n} \\ E(A) &= \sum_{i=1}^{v}\frac{p_i+n_i}{p+n}I(p_i,n_i) \end{aligned} \tag{8.1}$$

where p is the number of positive examples, n the number of negative, p_i the number of positive examples with value i for attribute A, n_i the number of negative with $A = V_i$, and v is the number of values for attribute A. The quantity $I(p,n)$ represents the expected amount of information contained in a subtree, and $E(A_i)$ represents the expected amount of information communicated by knowing the value of attribute A_i.[1] Like the LS and LN weights used by STAGGER, this measure reflects the notion of statistical dependence; when $gain(A) = 0$, the attribute and example classes are statistically independent (Schlimmer, 1986). The resulting implication is that ID3 should be highly tolerant of noisy examples. This is clearly the case as Quinlan (1983) demonstrates. Though this is in part due to the divisive nature of the decision tree, the *gain* measure effectively identifies relevant attributes to serve as root tests.

A potential limitation of this learning method is that it requires a large number of examples for simultaneous processing. However, with slight modification the method can be adapted to construct its decision tree in an incremental manner (Schlimmer & Fisher, 1986). In this version, dubbed ID4, instead of attempting to drive the tree's construction all at once, each level is built after sufficient statistics have accumulated to make a reasonable choice for the subtree root. No examples need be explicitly retained; after updating the p_i and n_i counts within the existing subtree, the example is discarded. Reasonableness of subtree construction

[1]This measure has been refined somewhat (Quinlan, 1986), for in this form it exhibits an inappropriate preference for attributes with a large number of values.

is assessed by applying a χ^2 measure to determine whether or not the correlations between the attributes and example classes is due to chance or is significant.

$$\begin{aligned} \chi^2(A) &= \sum_{i=1}^{v} \frac{(p_i - p'_i)^2}{p'_i} + \frac{(n_i - n'_i)^2}{n'_i} \\ p'_i &= p \times \frac{p_i + n_i}{p + n} \end{aligned} \tag{8.2}$$

The χ^2 measure is also used in ID3 as a stopping criterion or a way of determining when to quit building further subtrees. Thus it is natural that this measure should be used to inhibit premature subtree construction in ID4.

ID4 learns more slowly than ID3 but converges on the same result as its nonincremental relative. To its credit, the incremental variant never incurs excessive computation for any given example, and thus it is able to continually respond to the input environment.

The ID learning methods share a similar evaluating measure with STAGGER, and both exhibit a preference for simple descriptions by refining them until sufficiently effective. The similarity ends there, for STAGGER relies on a partially overlapping set of descriptive elements as a concept description; on the other hand, ID utilizes a single, highly-compiled description in the form of a decision tree. In matching, the sole description formed by ID completely categorizes new examples, but in STAGGER, all of the descriptions serve to modify expectation of a new example's class. This distinction is further illustrated by considering the methods' behavior when the concept drifts. STAGGER's backtracking and chunk formation processes split previously effective chunks and recombine them into new, predictive combinations. Conversely, the inflexibility of the decision tree representation affords only small changes near the leaves. If concept drift renders the root attribute ineffective, none of the inductive work of building the tree can be retained as the description shifts to meet the new concept.

This inflexibility of the decision tree representation has led Quinlan (in press) to consider methods that transform the tree into a set of production rules. A conjunctive rule is constructed for each path from the root to leaves in the decision tree. Then a separate analysis assesses the effectiveness of each of the rules as is and with various conjuncts eliminated. By first building the decision tree and then converting it into a set of rules, search is efficiently controlled. In the domains tested by Quinlan, this process leads to dramatic simplifications and moderate increases in predictive capability over novel examples. If the methodology could be adapted to the incremental characteristics of ID4, then it might provide an effective approach to the problem of concept drift. Though this representation appears to be somewhat more distributed than the decision tree, prediction of new example classes is largely the result of an individual characterization; even with a set of production rules only one is used at a time in concept prediction.

8.3 Production rule architectures

The nature of the learning methods employed in STAGGER and Anderson's ACT* model (Anderson, 1983) are quite similar; they both utilize a weighted concept description, and they both perform a bidirectional search from simple concepts to more general and more specific ones. In addition to being a model of learning, ACT* is also a model of cognitive performance. It is based on a bipartite division of memory: a *declarative* memory stores factual information while a *procedural* memory stores knowledge about how to do things. The learning methods applied to the procedural memory are the most similar to those employed in STAGGER.

A concept in ACT* is expressed in terms of production rules, which specify an action and the situations for which that action is appropriate. ACT* iterates by first determining which rules are appropriate and then selecting one to fire. The selection process is based on five principles: (a) the degree of match between the production and an example, (b) the strength of the production, (c) uniqueness of match bindings (no part of the example can match in the production in more than one way), (d) specificity of the production, and (e) goal direction. These principles combine to identify the single most appropriate production. Though the concept description is distributed among a number of separate productions, in a given cycle, only one production is used to direct concept expectation.

The productions are subject to two types of modification by the learning component of ACT*: compilation and tuning. The compilation processes compose multiple productions together into longer action sequences and proceduralize productions by replacing variables with specific values.

The tuning processes generalize, specialize, and strengthen existing productions. Generalization (or specialization) introduces a new more general (or more specific) copy of a production into procedural memory. These new productions vie with others on the basis of their strength and specificity. The strength of a production is a number roughly reflecting its predictive accuracy and the frequency of its reinvention. When a production is chosen by the conflict resolution principles, its strength is incremented. If it turns out to be incorrect, its weight is reduced by one fourth. A production's strength is also incremented whenever the learning processes propose an exact copy. This calculation is effective at dampening the potentially spurious effects of generalization and specialization. Langley (1987) has further demonstrated the ability of a similar formula to increase tolerance to noise in a learning production system. Moreover, these strengthening processes have the advantage over those used in STAGGER because only a small part of the concept description is modified. For large descriptions corresponding to complex concepts, STAGGER requires updating the LS and LN weights of each element, and this may be an expensive process.

ACT*'s method for strengthening productions combines the notions of correctness (predictive accuracy) and frequency (number of reinventions). By simplifying out the second and

focusing on the strength's reflection of accuracy, one can derive an estimate of the magnitude of a production's strength. Specifically, a production's strength will be stable when the increases are equivalent to the decreases, or $\Delta S = 0$. Probabilistically, strength increases are proportional to $p(matched)$ for the strength is incremented whenever the production is selected by the conflict resolution principles. Similarly, when the production applies but is incorrect, or $p(matched \wedge \neg example)$ its strength is reduced. If the increments are equivalent to the decrements then the strength is stable. Thus, $\Delta S = c_1 p(matched) - c_2 S \times p(matched \wedge \neg example)$ for $c_1 = 1$ and $c_2 = 1/4$. When the increments are equal to the decrements, the strength will be stable.

$$\begin{aligned} \Delta S = 0 \quad & \Rightarrow \\ c_1 p(matched) \quad &= \quad c_2 S \times p(matched \wedge \neg example) \end{aligned}$$

Dropping c_1 and c_2 (8.3)

$$\begin{aligned} S \quad &= \quad \frac{p(matched)}{p(matched \wedge \neg example)} \\ S \quad &= \quad p(\neg example | matched)^{-1} \end{aligned}$$

Because productions compete with each other, a weight must be interpreted in terms of the strengths of other productions. If a production exists with precisely the opposite semantics, then that production will have zero relative strength when its absolute strength is equal to its opposite's, or:

$$\begin{aligned} S \quad &= \quad \neg S \\ p(\neg example | matched)^{-1} \quad &= \quad p(\neg example | \neg matched)^{-1} \\ p(\neg example | \neg matched) \quad &= \quad p(\neg example | matched) \end{aligned} \tag{8.4}$$

This is an alternative form of the statement of statistical independence, and if ACT* maintains semantic opposites for productions, then it should exhibit reasonable tolerance to noisy data. This assumption may be unwarranted, though, for there does not seem to be any reason to suspect that production complements will be introduced or maintained by the learning components of ACT*.[2]

Langley's discrimination learning method (1985, 1987) has been implemented in a system named SAGE and is similar to Anderson's ACT* in many respects. The learned concepts are expressed as a set of production rules, one of which is mapped onto performance at any given time. Refinement of initially overly general productions occurs via specialization, and these rules are strengthened using an evaluation measure similar to Anderson's.

[2]Schlimmer and Granger (1986a) argued using a similar analysis that the strengths in ACT* were not equivalent to statistical dependence. I feel that this subsequent analysis is clearer and fairer.

In addition to demonstrating this method's ability to tolerate each type of one-directional noise, Langley has also shown that it can track simple changes in a concept definition over time. As the definition of a concept drifts, recently learned productions are weakened while the specialization process proposes new ones. The new productions are strengthened and eventually overwhelm any previous learning. There may be some differential performance when an inherently disjunctive concept changes as opposed to a conjunctive one, though this was not tested. This might arise because production rule descriptions make a modularity assumption, but only with respect to disjunctions. Conjunctive concepts tend to be described with a few specific rules and modularity is abandoned. If a disjunctive concept drifts slightly, then many of the productions can be retained, leading to smooth recovery. However, if a conjunctive definition changes, a completely new rule may be required, necessitating the repetition of the refinement processes. Incorporating a backtracking mechanism similar to the one used in STAGGER (Section 4.4.3) would probably alleviate any imbalance between disjunctive and conjunctive drift recovery.

8.4 The Genetic Algorithm

Holland's (1975, 1986) Genetic Algorithm is an incremental method that utilizes a distributed concept description and is able to tolerate noise as well as track changes in concepts over time. In this framework, the concept description is represented as a set of bit patterns that are subject to natural selection style laws of propagation. These bit patterns (called *genes*) are roughly composed of a matching specification and an output, similar to the left and right-hand sides of a production rule. These genes interact via a *message list* where the patterns specified by matching genes are placed and to which the left-hand sides of others are tested for matching.

The initial concept description is a randomly generated *population* of genes that individually correspond to random conjunctions of attribute-value pairs. Each bit location typically represents a single attribute-value. These genes have an associated strength to reflect the predictiveness of each gene. In matching with a new example, all of the matching genes compete to control prediction. This is accomplished through a *bidding* process. Though Holland prefers to consider this a parallel activation system without any conflict resolution rules, the rule for determining bid size incorporates two of the principles used in ACT*; each gene that matches may make a bid proportional to its strength and specificity. The genes with the winning bid place their output on the message list. They pay out their bid to the previous bid winners, reducing their strength, but they are strengthened in turn if their messages are used by the strongest genes in the next cycle. This process is reminiscent of old-fashioned fireman lines (where water instead of weight strength is passed along chains of participants) and hence is

termed the *bucket brigade.*

After each match and execute cycle, some of the genes are modified in an attempt to uncover more successful bit strings. The strongest genes are retained, and the weakest are discarded. Vacant slots are filled with modified copies of the most successful genes, either by randomly *mutating* particular bits in the gene or by *crossing-over* a pair of genes to make a new pair. In the latter operation, a split point in the gene is randomly selected, and one new gene is composed of the substring prior of the split point in one gene and the substring aft of the split in another. A second new gene is formed from the remaining aft end of the first and the prior portion of the second. These simple operations are the analogue of those assumed to be at work in natural selection, and thus the methods have come to be called the Genetic Algorithm. Of particular note is the fact that these ways of generating new bit patterns do not make use of the generality ordering among concepts nor the individual semantics associated with the underlying bits, yet the methods are robust (Wilson, in press) and perform significantly better than random exploration of the possible bit patterns (Mauldin, 1984).

The bucket brigade process of strengthening some genes and weakening others is similar to the process used by ACT* to weight rules. Some change is made to the strength whenever the gene matches and an opposing change whenever the gene appears to have matched correctly. Here, when a gene matches, its strength is decreased through its bid. When the gene is correct, it is rewarded in due course through the bidding process. Since the gene's bid is proportional to its strength, so is its decrement. Thus, $\Delta S = c_2 p(matched \wedge example) - c_1 Sp(matched)$ for some constants c_1 and c_2. Following the procedure used for analyzing ACT* rule strengths, consider when the changes in a gene's strength are zero, and the decrement is equal to the increment.

$$\begin{aligned} \Delta S = 0 \quad & \Rightarrow \\ c_2 p(matched \wedge example) \quad &= \quad c_1 Sp(matched) \end{aligned}$$

$$\text{Dropping } c_1 \text{ and } c_2 \tag{8.5}$$

$$\begin{aligned} S \quad &= \quad \frac{p(matched \wedge example)}{p(matched)} \\ &= \quad p(example|matched) \end{aligned}$$

Like the ACT* weight measure, this process has the advantage that only a small part of the concept description is updated at any one time, unlike the methods used in STAGGER and connectionist methods where all of the weights are changed at each learning opportunity. However, as with ACT*, in order to accurately reflect the notion of statistical dependence, the Genetic Algorithm would have to retain the opposite of each bit string: something the method is unlikely to do. In fact, the "survival of the fittest" heuristics seem to guarantee that the set

of genes will become dominated by highly similar patterns. Mauldin (1984) reports a similar phenomena in which the genes exhibit premature convergence on a sub-optimal solution for just this reason.

However, this methodology should be quite effective at tolerating concept drift. In fact, since slight variations in the bit strings are continually being introduced into the concept description, small changes in the concept definition can be smoothly incorporated into the description.

8.5 Connectionist learning methods

In contrast to the other learning methods discussed in this chapter, connectionist methods avoid the explicit use of symbolic representations, preferring instead to represent concept descriptions solely in terms of weighted interconnections between input features and example classes. STAGGER's weight learning method is a simple example of this type of model, for without the introduction of Boolean chunks, the concept description is simply a set of weighted predictors. Interpreting these descriptions requires an analysis of the relative magnitudes of individual weights.

There are several attractive properties to this type of representation. It naturally allows for partial matching between examples and the concept description; it graciously tolerates poor quality data both in the form of missing information and noisy examples; it readily handles concept drift; it provides a smooth interpolation for previously unseen examples; and, intuitively, as a representation it appears to be closer to actual neural circuits.

Recent examples of this type of learning method go beyond the weight method used in STAGGER by incorporating several layers of weighted predictions. Instead of a single output node fed by weighted featural inputs, a layered model funnels inputs to intermediate nodes that in turn pass their output on to final output nodes. These intermediate, or *hidden*, nodes serve to expand the representational capability of single layer systems in the same way that the Boolean chunking method increases the representational capability of STAGGER's weight method. The key issue for researchers in this area has been devising methods to properly adjust the weights associated with the hidden nodes. Learning methods for single layer systems are relatively straightforward – STAGGER's weight method is one example. Yet the process of modifying the intermediate node weights involves *structural credit assignment* (Anderson, 1986) – determining which node weights to change and by how much. This has been the primary focus of recent research. As a survey of the available methods, I review two representative methods. The first uses a well known error-correction weight learning rule and makes primarily local changes in the hidden nodes. The second system relies on a distinct but equally rich heritage by using a continuously correcting weight learning rule; moreover, it

modifies the hidden nodes in a much more distributed manner.

8.5.1 Training bi-layer, linear threshold units

Hampson's learning method is a synthesis of the Perceptron (Rosenblatt, 1958) and a localistic strategy for representational learning. The Perceptron is computational node, each of which computes a weighted sum of its inputs called the node's *activation.* If activation is above a threshold, its output is interpreted as positive; negative if below. The Perceptron learning rule (Duda & Hart, 1973) specifies that for the single layer case, the weights are modified whenever the output for a given pattern is inappropriate: increased if output was too low and decreased if too high. The focus of modification is based on the features of the misclassified examples. If a feature is present its weight is modified; if absent, the weight is stable even though the example was misclassified. The change in a weight may be stated as

$$\Delta S = r \times Feature \times \left[\begin{array}{l} (\text{i}) Example \times (1 - Output) \\ + \\ (\text{ii})(1 - Example) \times Output \end{array} \right] \tag{8.6}$$

where r is the learning rate ($0 \leq r \leq 1$), *Feature* is zero if the feature is missing and one if present, *Example* is zero for nonexamples and one for positive examples. Note that if the example is positive (i) and the node's output is one then ΔS is zero. Similarly, if the example is negative (ii) and output is zero, no change is made. Only when the output and the example's identity do not agree are the weights changed.

To assess the relationship between this measure and statistical dependence it is sufficient to note that any misclassified example, including noisy ones, influence the weights equally, and thus tolerance to one-directional noisy situations is unlikely to be significantly different than to two-directional noise.

Hampson refines this weight learning rule by identifying an additional source of focus for weight modification. The simple rule modifies the weights for any feature present on an erroneously identified example, but a more selective heuristic would be to modify only those features that are actually predictive of the example's identity. Hampson suggests assessing predictiveness by comparing the probability of appropriate output given the input feature with the probability of output, or

$$P = p(Example|Feature) > p(Example) \tag{8.7}$$

Like the Bayesian LS and LN weights used by STAGGER, Equation 8.7 reflects statistical dependence. Focusing weight changes in this manner ensures that an appropriate distinction will be made between one- and two-directionally noisy situations, with considerably more

tolerance extended to the first and less to the latter. Features in a one-directionally noisy situation will be identified as predictive, and their corresponding weights will be influential in matching. However, the two-directional noisy features are unpredictive; they will not be allowed to garner strong weights or influence concept expectation.

For the multiple layer network, the weight modification rule is altered somewhat. If the rule for a single layer node were uniformly applied to each of the nodes in the system, then each would attempt to solve the representational task on its own; they would not divide up the representational work. Therefore, the rule for adjusting the weights of intermediate, or hidden nodes explicitly identifies representational shortcomings and attempts to assign nodes to address them. Simply, for all negative examples, all of the hidden nodes should be off, and for all positive examples, at least one of the hidden nodes should be on. Implementing this heuristic for the first case is implemented by training all of the intermediate nodes with the negative example.

Differential training arises in implementing the second part of this heuristic: ensuring that at least one of the hidden nodes is on for any positive example. A rough cut would be to train the node with the highest activation on this positive example and leave the others alone. Hampson implements a generalization of this by varying the learning rate r for hidden nodes in proportion to their (correct) activation. In the limit, if a node's activation is zero, then its weights are not changed. The node with the highest rate of change will be the one that had the highest activation.

$$r_{hidden} = \frac{Activation - Activation_{Min}}{Activation_{Max} - Activation_{Min}} \tag{8.8}$$

This latter portion of the weight modification processes addresses the structural credit assignment problem by proposing minimal changes necessary to account for the erroneously classified input: nodes closest to correctly classifying it are encouraged to do so.

Training an intermediate layer of nodes in this manner, and adjusting the weights of the final output node following the learning method outlined for a single layer node, result in a system that appears to be representationally complete. Though the methods are only formulated for a two-layer system, as Hampson notes, the smaller network is complete over the space of Boolean functions.

This method should be relatively tolerant of noisy data, and it is able to track changes in a concept's definition. Flexibility in the approach naturally arises out of the weighted concept representation; when the concept drifts, the network begins to falter, the weight modification rules are invoked, and appropriate changes in the connection weights are made.

8.5.2 Training multi-layer semilinear units

Another notable approach for training hidden nodes builds on the Widrow-Hoff weight learning rule (Duda & Hart, 1973) and makes distributed changes to the hidden nodes in order to adjust representations appropriately. With the Widrow-Hoff rule, a unit like the Perceptron serves as the performance element. What differs is that the weights are adjusted by comparing the desired activation for the node with its internal activation instead of its output. This rule may be stated simply as:

$$\Delta S = r \times Feature \times (Example - Expectation) \tag{8.9}$$

This equation is similar to Equation 8.6 with the simple substitution of *Expectation*, or the node's internal activation, for *Output*. The result is that the weights are updated after processing each example, not only those that are incorrectly classified. If activation for a positive example is less than maximum weights corresponding to the features are strengthened, even if activation is above threshold. Similarly, correctly predicted negative examples also lead to weight change if activation was greater than the absolute minimum.

Because Equation 8.9 is a generalization of the weight learning rule for the Perceptron (Equation 8.6), the same analysis applies with respect to statistical dependence and noisy data. There does not appear to be any fundamental distinction between weight adjustment in the presence of one type of noisy event (a one-directionally noisy situation) and the presence of both types (two-directional).

Like the Perceptron, the Widrow-Hoff weight learning rule by itself is insufficient for training hidden nodes. Instead, Rumelhart, Hinton, & Williams (1986) rely on the notion of error propagation to assign the appropriate internal weights. Training the first layer is straightforward; the error is defined as the difference between the node's activation and the desired output (as in Widrow-Hoff). However, for the hidden nodes, no such external evaluation is available, so Rumelhart *et al.* suggest a method for computing an error estimate. Using a generalization of the Widrow-Hoff learning rule, the desired activation for each hidden node is computed by measuring the degree to which it causes an output node to attain a suboptimal activation level. A hidden node is greatly in error if it is strongly connected to an output node and incorrectly causes the latter to reach an overly low or overly high activation level. Less of an error is indicated if the connection is weak.

The implementation of this method is called *back-propagation* because the error estimate is pushed back from the output nodes through the hidden nodes. In general the approach can be applied in a recursive manner to many layers of nodes. The specific mechanism for estimating error requires the ability to differentiate node output functions. The linear threshold output function like the one used in (Hampson, 1983; Hampson & Kibler, 1983) is discontinuous and cannot be differentiated, so a *semilinear* unit is instead used that approximates linear

threshold behavior. With this derivative in hand, it is relatively straightforward to compute the desired activation level for any hidden node in the system by pushing an error signal back through the connecting weights from the output nodes.

Like the learning rule used by Hampson's single layer network, this process of adjusting the weights through a propagated error estimate is a type of hill-climbing process. In fact, Rumelhart *et al.* analytically show that back-propagation amounts to the steepest descent through the error space. As such the method is susceptible to the problems associated with getting stuck on locally optimal solutions. One way of minimizing this difficulty is to employ a relatively large number of hidden nodes, and thereby expand the size of the hill-climbing search. With only one hidden node the search is a pure hill-climbing. By increasing the number of nodes, the search takes on more of a beam search flavor, and the likelihood of one of the hidden nodes finding the optimal solution is increased. Of course, many concepts require a number of hidden nodes to represent the inputs for the output nodes, and increasing the supply improves the chance of proper convergence.

One of the key distinguishing features of the back-propagation model is that weight changes are distributed over a large number of nodes as compared to the method used in (Hampson, 1983; Hampson & Kibler, 1983). In the latter, when none of the hidden nodes fire for a desired output, the ones closest to a correct response are modified. Since each change is proportional to the level of activation, the network tends to modify only a few nodes, encouraging a localization of representation. Back-propagation, on the other hand, addresses this structural credit assignment problem by distributing the changes over a large number of nodes. As with the original Widrow-Hoff rule, each node's weights are modified after each example since there is always some difference between the desired and actual activation levels.

One drawback of spreading the modification out over the network is that it can lead to a lack of generality in the concept description. In the extreme, each hidden node will move toward representing an individual input, rather than salient properties of the inputs (Barto, 1987, personal communication). One way to counteract this effect is to restrict the number of hidden nodes, and thereby force the back-propagation method to form relatively local changes and concise, general descriptions. Regrettably, this is incompatible with the technique of increasing the number of hidden nodes as a way of ensuring convergence. This dilemma between too few to reliably capture the concept and too many to retain generality is one of the issues yet to be fully resolved by researchers.

8.5.3 Summary

These two methods for training multi-layer connectionist networks serve as representative samples of recent work that build on the early heritage of adaptive networks. Hampson's learning method extends the Perceptron and therefore utilizes an error-correcting strategy.

In addressing structural credit assignment, or identifying which hidden node weights need to be changed, he adopts a localistic modification rule that adjusts the weights of as few as one node.

In contrast to these techniques, Rumelhart *et al.* base their work on a generalization of the Widrow-Hoff learning rule. Instead of adjusting weights only following a classification error, node weights are updated after each example. This strategy is applied to the hidden nodes as well. All of these weights are changed by the degree to which they contributed to a suboptimal activation level at the output node.

8.6 Overview

8.6.1 Dual learning methods

Each of the learning methods discussed in this chapter relies on a weighting process and a component that learns higher-level representations. In ID3, the weights are used to select a single attribute as a test for the root of a discrimination tree. The process of dividing the examples and iterating the weighting serves to alter the representation base, in effect transforming each remaining attribute into a conjunction with previously tested attribute-values. In ACT* and SAGE, the representational component that forms variations of production rules has received the most attention, but these methods rely on a weighting process to direct matching and evaluate effective production variants. Similarly, in the Genetic Algorithm, the bidding process serves to weight individual genes, while the mutation operators identify new, conjunctive descriptors that allow effective classification. In connectionist methods, the hidden nodes serve to restructure the inputs for output nodes, allowing the simple weighting processes used in the latter to correctly specify complicated functions of the initial inputs. This technique gracefully applies weighting to both the simple correlational and representational learning issues. And, as the previous chapters argue, STAGGER is composed of a weighting method that both relies on and guides a higher level Boolean chunking method for effective prediction of future inputs.

For each of the methods, the weighting component serves to provide noise tolerance and as an evaluation function for the representational learning component. To see the latter, note that ID3 chooses the most informative attribute as the decision tree root, ACT* and SAGE match the strongest productions and in doing so set the stage for the modification of those productions, the Genetic Algorithm mutates the strongest genes, Hampson's method modifies the hidden node closest to correct prediction, the back-propagation method alters weight in proportion to the error, and STAGGER forms new Boolean chunks based in part on the guidance provided by the LS and LN weights. Similarly, in each of the methods, the representational learning component serves to expand the capability of the simple weighting

process. In ID3, without the decision tree, only one test could be made. In ACT* and SAGE, existing production rules could not be modified; learning would be restricted to selection. In Holland's method, without gene mutation, performance could not improve beyond the bounds set by the initial choice of random genes. The connectionist methods and STAGGER would be limited to learning extremely simple concepts; this is the reason for the introduction of multiple layer networks and the Boolean chunking method, respectively.

Despite the strong functional similarities between these models, they differ with respect to their use of the concept representation. At one extreme, ID3 utilizes a unified representation that is consulted as a final authority for classification. At the other extreme, the back-propagation method uses a highly distributed concept representation, in which few hidden nodes play an indispensable role, and all of the nodes participate to some degree in example identification. The unified representation approach has the advantage of being somewhat more clear and therefore easier to verify (Michalski, 1983). On the other hand, the distributed form is more robust, for no individual hidden node is completely responsible for classification performance. This latter method may also be more amenable to efficient implementation, whether in biological or parallel computer hardware.

A finer distinction is possible between the methods that employ a distributed concept representation; some methods rely on distributed properties for learning, but select a small portion for use in classification. The production-rule learning models ACT* and SAGE are prime examples of this strategy, for they maintain numerous variants of effective productions as an explicit part of the learning method. For instance, the strength of a production also reflects the number of times it has been invented. If ineffective productions are not retained, then this process of rewarding reinvention would be fruitless. Though they employ duplicity for learning's sake, these models prefer to select a single production in order to form an expectation of an example's class. This is undoubtedly an artifact of their heritage as condition-action rules, where the right-hand side of the production was conceived as a motor command; selecting one production at a time eliminates attempting mutually exclusive actions. Rules that suggest the same action or operator could be combined. This is the approach followed by the Genetic Algorithm, for it places few constraints on the number of genes that may be allowed to bid and place their messages on the list. In this case, the number of compatible messages is an implicit measure of the degree of confidence in the collective action of the genes. Taking this approach to the limit, the connectionist models and STAGGER explicitly combine all available evidence in an attempt classify examples accurately. As an example of the operational distinction between having a distributed representation and using it for matching, note that for a disjunctive concept, ID3, ACT*, and SAGE would yield no greater expectation of a positive example if one disjunct were present or if all were. Conversely, the latter methods would each exhibit greater confidence in a disjunctive concept as more of the disjuncts are satisfied.

8.6.2 Evidence combination criteria (ii)

The similarity between STAGGER and the other methods discussed in this chapter extends to conformance with the evidence combination criteria suggested by Cohen (1977) presented in Section 2.3.3. The primary constraints are on the use of the concept description in matching it to example features. There are some noticeable differences, allowing a clearer understanding of the close relationship between some learning methods.

First, Cohen notes that when combining evidence, a conjunction of evidence should be stronger than any single component. This intuitive principle is not afforded using a strict Bayesian approach, for multiplying together the necessary conditional probabilities has the inverse effect: adding new evidence to a conjunction only weakens the resulting outcome. However, though the methods surveyed do not all use a distributed concept representation, they do follow this suggestion. In ID3, where the concept description is a decision tree, each of the successive decisions corresponds to a conjunct. As matching proceeds deeper within the tree and more of these conjuncts are satisfied, the confidence associated with classification rises: near the top only a random guess can be made, while a leaf yields a certain assertion. The production rule models and the Genetic Algorithm also accrue stronger expectations as more conjuncts are satisfied. Instead of an increasing level of match, this property arises from the preference for specificity as a criterion for conflict resolution. A rule or gene with more matching conjuncts is preferred. Like the modularity asymmetry between modular disjunctions and conglomerate conjunctions, these methods yield stronger expectation with increasing conjuncts but not with disjuncts. The individual disjuncts of an OR do not have an additive effect. In general connectionist methods sum conjunctive contribution and result in stronger expectation given more evidence. As in STAGGER, each of the matching subcomponents contributes its part to metering expectation.

Second, Cohen observes that jurors do not accept a chain of evidence unless each link is secure. Though Bayesian mathematics is too strict with conjunctive evidence, it is too lenient with transitive evidence. The behavior of three of the methods described here do correspond to Cohen's observation, though one clearly does not. If we constructed a set of decision trees in a staged manner, with the classification of the first serving as a test in the second and so forth, then transitive reasoning in ID3 would require establishing the truth of each intermediate classification; no median state of expectation could be passed along. Similarly, in ACT*, SAGE, and the Genetic Algorithm, chains of productions or genes require the satisfaction of their preconditions before they vie for classification control. In a problem solving situation (as a multi-step classification might be), each intermediate conclusion would have to be certain in order for the subsequent inferences to be drawn. However, the connectionist methods do not exhibit juror-like behavior in transitive reasoning. Instead, intermediate conclusions reached by mid-layers of nodes are passed along to nodes further downstream as a continuous quantity;

if the conclusion is weak it is propagated to other nodes where it may be boosted up into a confident assertion. If these models used discrete outputs, i.e., either zero or one, then they would avoid the brunt of this criterion, yet at least in the case of the back-propagation method, a continuously valued output appears to be central to the learning method's formulation. In STAGGER a chain of concept descriptions interact through a discretely valued attribute for each, and thus it does not allow an argument to be built on weak links (Schlimmer & Granger, 1986a).

The third criterion posed by Cohen asserts that a juror does not assign equal confidence to both an assertion and its negation. In mathematical probabilities, this is typically the case, for $p(\neg A) = 1 - p(A)$. As with transitive evidence, one of the methods behaves inappropriately in this case: ID3 does not conveniently allow for an expectation of a nonexample separate from an example's. The most obvious approach would be to construct a decision tree for each of the possible example classes: one for positive examples and one for nonexamples. However, in the interesting case where the trees disagree (each claiming that the example was in their class), no cohesive answer is available. How could the confidence of one be contrasted with the other? The other three methods fare better, though, and allow competing expectations to have unique levels through multiplicity of the concept description. The production rule methods explicitly allow for the possibility that the right-hand action side of productions are competitive; these methods include conflict resolution principles designed explicitly to avoid making two contradictory assertions.[3] Moreover, the Genetic Algorithm allows for the coexistence of mutually exclusive predictions, for no explicitly syntactic or semantic resolution is imposed on the bidding process. Some resolution is necessary, though it is not clear how it is generally implemented (Holland, 1986). The connectionist methods and STAGGER also allow for separate expectations of potentially competing concepts, in both cases through multiplication of the concept descriptions. Adding a new output node for the negation of a concept yields a competing evaluation that is not necessarily just an inverse. For instance, in training STAGGER to direct a organism in the spatial learning task of Section 3.3, expectation for each of the opposing operators was compared directly, eliminating the need for the explicit threshold used in Equation 2.3.

Finally, Cohen proposed that proof must be established beyond reasonable doubt, as jurors are instructed to do. This relates closely to the notion of a concept description as a set of necessary and sufficient elements. The point here is not that necessary and sufficient features are all that is required for concept description (Smith & Medin, 1981), but that these properties are desirable whether they are explicitly programmed or emergent. Mathematical models of evidence typically require confidence beyond a certain point, neglecting the possible implications of omitted evidences that appear crucial to the outcome. Each method illustrate

[3]These methods do not appear to host complementary left-hand side conditions, as I noted in Section 8.3.

this property, though the case for one of the connectionist methods is somewhat tenuous. ID3 certainly attends to each salient feature of examples, for each point must be met in turn. All of the conjuncts must be true, and all of the negated conditions must be avoided as matching proceeds down the tree. Similarly, the production rule methods and the Genetic Algorithm readily express the necessary and sufficient dependencies between example features and classes. STAGGER, too, represents necessary and sufficient features through its explicit use of the LS and LN weights. Hampson's bi-layer method delegates the detection of sufficient features to the output nodes and necessary feature encoding (if required) to the inner nodes. However, the back-propagation model does not utilize any explicit encoding, relying rather on patterns of activation to correspond to feature patterns. A juror's scan of a checklist of establishing points and possible let-outs does not appear to map closely into this type of process.

8.6.3 Conclusions

In closing, many of the functional properties seen to be effective for STAGGER are exhibited by other learning methods; the contribution of STAGGER is to make these interactions explicit and to identify the source of these noteworthy properties. The approach is not wholly unique, but rather the methods are strongly related to other existing work. One interpretation is that STAGGER represents a synthesis of other methods, combining a connectionist learning method and a symbolic one. The first regulates matching and evaluates the second's contributions, and the second restructures the input so the first can function properly.

Chapter 9

Overview

This thesis is a study of the type of mechanisms that are involved in the abstractive process. Reconstructing the line of reasoning used in the introduction, the environment inherently presents a highly variable picture. To make sense of its world, an intelligent agent must find commonality among events. A strong hypothesis I have embraced here is that the mapping from specific events into general classes of experience may be learned. As evidence, I presented a model for learning from examples that describes its experience as a set of weighted, symbolic components. Modification occurs as the model changes both weights and components. A testable implementation of the model lends credibility to assertions concerning its applicability and robustness.

The second chapter introduced the concept representation used by STAGGER. Each concept is described as a set of elements, or weighted patterns. One weight formalizes the sufficiency of the pattern for expectation of the concept while the other captures the pattern's necessity. When matching the concept description with new examples, one weight from each element contributes through a summation process to yield an overall matching score. This chapter further described how STAGGER modifies the concept description by changing element weights, and in doing so, modifies the outcomes from the matching process. Two simple techniques, similar to single-layer connectionist methods, were presented for altering weights.

In the third chapter, I illustrated the operation of matching and weight modification through a series of four examples. The chapter began by presenting the basic operations of STAGGER, illustrated through its acquisition of a simple, artificial concept. The second example was borrowed from researchers using connectionist methods. This spatial learning task showed the flexibility of the weighting method and fostered intuitions about the method's similarity to connectionist techniques. Third, an example drawn from information retrieval illustrated STAGGER's functioning with tasks involving a large number of concepts. The fourth example closed the chapter by empirically measuring the completeness or coverage of the weight learning method. Performance over a comprehensive set of concept learning tasks

showed the sharp limitations of the weighting process and motivated the need for an auxiliary learning method.

The fourth chapter introduced an auxiliary Boolean learning method that alleviates the deficiencies of the weighting process by forming new AND, OR, or NOT combinations of existing concept elements. These new complex elements are called *chunks*, and they serve to rewrite the input for the weight learning method. The formation of new chunks is driven by matching errors and element weights. A falsely omitted example directs the Boolean method to add a more general chunk that will match the missed example. Conversely, a falsely included example suggests the need for a more specialized chunk that will not match the nonexample. Each formation also attempts to include elements that have strong weights. The weights of component elements establish a testing threshold for the new chunk; if the chunk garners a weight larger than this, it is retained as a part of the concept description. Otherwise it is pruned, and concept description size is limited. By adding new chunks, the Boolean learning method is restructuring the input for the weight learning process, allowing the latter to search a larger space of concepts with its description.

Chapter 5 followed the precedent of Chapter 3 by stepping through a series of four examples. The first example was again simple, this time used to illustrate the cooperative action of the two learning methods. Two subsequent examples were drawn from naturally constrained domains. In addition to providing evidence for the pervasiveness of learning from examples, they raised issues surrounding poor quality data. In closing, I reexamined the comprehensive task first introduced in Chapter 3; adding Boolean chunking to Bayesian weighting improved the coverage of the latter.

In the sixth chapter I explicitly studied the effects of poor quality data on STAGGER's ability to form effective concept descriptions. Individual sections focused on three properties that arose in Chapters 3 and 5: noisy examples, unknown values, and concept change over time. Noisy examples were divided into two distinct classes: those that retain information (one-directional) and those that do not (two-directional). In accord with statistical and psychological results, STAGGER tolerated significantly more of the former noise type than the latter. Unknown values were shown to slow learning but not affect asymptotic quality. Third, STAGGER's methods recovered gracefully from concept drift and displayed several useful heuristics.

Chapter 7 discussed the interplay between psychological results and the development of effective methods. Briefly, experimental results strongly constrain plausible learning mechanisms, and if compliant methods are implemented, the program may itself be tested and lead to new psychological predictions. Two sections of this chapter described clear psychological results and their effects on the formulation of the Bayesian weighting method. Several of the desirable qualities observed in Chapter 6 resulted from conformance to these constraints. A

third section covered the other half of the coin by reinterpreting STAGGER's performance on the 19 comprehensive learning tasks as a detailed hypothesis about humans' abilities in similar situations.

Finally, in Chapter 8 I surveyed related work that addresses the task of learning from examples. Individual sections focused on systems that employ a pair of learning methods: one to adjust weights and another to modify the base representation. Several of the better known research efforts fall naturally into this framework and many exhibit the computational properties discussed here for just this reason.

9.1 Contributions

In summary, I have attempted to fulfill two purposes in writing this thesis. Primarily, this thesis describes the functions and capabilities of the learning methods named STAGGER. Chapters 2 and 3 followed the format of description and illustration – first presenting the Bayesian weighting method and then demonstrating its operation in a number of domains. Chapters 4 and 5 followed the same outline for presenting the Boolean chunking method. A second purpose followed here has been to persuade the reader that the task of learning from examples is a general and useful one. Demonstrations of STAGGER's methods have been drawn from diverse situations, and Chapter 5 examined the methods' range of applicability through a comprehensive assessment.

The work here makes three distinct contributions to machine learning. First, the task of learning from examples has been relaxed to allow for noisy data, unknown values, and concept change over time. STAGGER's methods are sufficient for this task, and because it is more robust than previous work, it will be more widely applicable to realistic learning applications.

Second, the two learning methods (Bayesian weighting and Boolean chunking) exhibit a type of representational learning. The chunking method changes the substrate from which weight learning proceeds. Due to this action the weighting method is able to acquire a larger class of concepts that is otherwise possible. This framework reflects the motivation behind recent interest in multi-layer connectionist systems and generalizes work on bias adjustment.

Third, STAGGER demonstrates a useful cooperation between a numeric learning method and a symbolic one. Each of the two learning methods relies on a distinct heritage and brings advantages to the learning task. The numeric, weighting method naturally accommodates noisy data and adapts to drift. On the other hand, the symbolic, chunking method restructures the concept description language, affording user verification of concept descriptions and allowing the simpler method to capture more concepts.

9.2 Future work

If this thesis has answered questions about concept acquisition, it has raised more. The model presented is effective in many situations, as the previous chapters illustrated, but there are useful heuristics that have not been included. Perhaps principally, STAGGER does not use background information to refine concept descriptions. Vere (1977) refers to this body of information as a set of facts that are potentially relevant to learning problems but are not actually part of example descriptions. For instance, if the task focuses on learning to detect winning chess end-games, information pertaining to the possible moves of the pieces might not be part of the example boards (Quinlan, 1979), yet the learning system could be aided by the information that a Rook moves along ranks and columns, etc., as Vere and others (Flann & Dietterich, 1986; Utgoff & Mitchell, 1982) have demonstrated. It may be that background information could guide the Boolean method's formation of chunks (as in [Utgoff, 1986; Utgoff & Mitchell, 1982]), or perhaps more generally, both the weighting and chunking methods.

A second issue, briefly alluded to in Section 7.4, is the relatively impoverished set of chunking operators used by STAGGER's Boolean learning method. There are many concepts for which AND, OR, and NOT are ineffective at capturing concept regularity. For instance, the BACON system of Langley, Bradshaw, and Simon (1983) learns about planetary motion and circuit resistance; both involving mathematical expressions of example features. Adding this type of capability to a system like STAGGER may be relatively straightforward. Some recent work that dynamically partitions real-valued attributes into nominal values appears promising (Schlimmer, 1987) but is only a small step in this direction.

This work simplifies a third interesting dimension: It assumes that all of the examples are part of predefined classes. By contrast, conceptual clustering does not attempt to identify the class of new examples. Rather, the point of learning is to group the examples into classes. This may aid in understanding the correlation between examples or improve inference processes. Though work in this area has developed efficient, noise-tolerant methods (Michalski & Stepp, 1983; Fisher, 1987), the representations used for concepts is comparatively static. To date, only a few systems have explored the potential of representation change (Langley, Zytkow, Simon, & Bradshaw, 1986).

Despite its shortcomings, this work lays three important foundations: it is a robust method for learning from examples, it frames its methods as cooperative entities interacting to improve each other's performance, and it demonstrates a useful interaction between a symbolic technique and a numeric one. Additional challenges lay ahead that will test these contributions. I hope that by framing the work clearly, it may serve as a stepping stone from which to confront these difficulties, allowing the research community to further its overall goal of understanding learning and general intelligence.

References

Anderson, C. W. (1986). *Learning and problem solving with multilayer connectionist systems.* Unpublished doctoral dissertation, Department of Computer and Information Science, University of Massachusetts, Amherst.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Audubon Society. (1981). *Field Guide to North American Mushrooms.* New York: Alfred A. Knopf.

Baker, A. G. (1976). Learned irrelevance and learned helplessness: Rats learn that stimuli, reinforcers, and responses are uncorrelated. *Journal of Experimental Psychology: Animal Behavior Processes, 2,* 130–141.

Barto, A. G. (1987). Personal communication, May.

Barto, A. G., & Sutton, R. S. (1981). Landmark learning: An illustration of associative search. *Biological Cybernetics, 42,* 1–8.

Bower, G. (1972). Perceptual groups as coding units in immediate memory. *Psychonomic Science, 27,* 217–219.

Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking.* New York: John Wiley & Sons.

Chase, W. G., & Ericsson, K. A. (1981). Skilled memory. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Lawrence Earlbaum.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology, 4,* 55–81.

Cohen, J. L. (1977). *The probable and the provable.* Oxford, England: Oxford University Press.

Congressional Quarterly Inc. (1985). *Congressional Quarterly Almanac, XL* (98th Congress, 2nd session 1984). Washington, DC.

de Kleer, Johan. (1984). Choices without backtracking. *Proceedings of the National Conference on Artificial Intelligence* (pp. 79–85). Austin, TX: Morgan Kaufmann.

Duda, R., Gaschnig, J., & Hart, P. (1979). Model design in the Prospector consultant system for mineral exploration. In D. Michie (Ed.), *Expert systems in the micro electronic age.* Edinburgh, England: Edinburgh University Press.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

Fine, T. L. (1973). *Theories of probability.* New York: Academic Press.

Fisher, D. (1987). Conceptual clustering, learning from examples, and inference. *Proceedings of the Fourth International Machine Learning Workshop* (pp. 38–49). Irvine, CA: Morgan Kaufmann.

Fitzgerald, R. D. (1963). Effects of partial reinforcement with acid on the classically conditioned salivary response in dogs. *Journal of Comparative and Physiological Psychology, 56*, 1056–1060.

Flann, N. S., & Dietterich, T. G. (1986). Selecting appropriate representations for learning from examples. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 460–466). Philadelphia, PA: Morgan Kaufmann.

Gibbon, J., Berryman, R., & Thompson, R. L. (1974). Contingency spaces and measures in classical and instrumental conditioning. *Journal of the Experimental Analysis of Behavior, 21*, 585–605.

Granger, R. H., Jr., & Schlimmer, J. C. (1986). The computation of contingency in classical conditioning. *The Psychology of Learning and Motivation, 20*, 137–192.

Hall, R. J. (1986). Learning by failing to explain. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 568–572). Philadelphia, PA: Morgan Kaufmann.

Hampson, S. E. (1983). *A neural model of adaptive behavior.* Unpublished doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.

Hampson, S., & Kibler, D. (1983). A Boolean complete neural model of adaptive behavior. *Biological Cybernetics, 49*, 9–19.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Kamin, L. J. (1969). "Attention-like" processes in classical conditioning. In B. A. Campbell & R. M. Church (Eds.), *Punishment and Aversive Behavior.* New York: Appleton-Century-Crofts.

Kremer, E. F. (1971). Truly random and traditional control procedures in CER conditioning in the rat. *Journal of Comparative and Physiological Psychology, 76*, 441–448.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning, 1*, 11–46.

Lancaster, F. W. (1979). *Information retrieval systems: Characteristics, testing and evaluation.* New York: John Wiley & Sons.

Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science, 9*, 217–260.

Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development.* Cambridge, MA: MIT Press.

Langley, P., Bradshaw, G. L., & Simon, H. A. (1983). Rediscovering chemistry with the Bacon system. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Langley, P., Zytkow, J. M., Simon, H. A., & Bradshaw, G. L. (1986). In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Mackintosh, N. L. (1975). A theory of attention: Variations in the associability of stimulus with reinforcement. *Psychological Review, 82,* 276–298.

Mauldin, M. L. (1984). Maintaining diversity in genetic search. *Proceedings of the National Conference on Artificial Intelligence* (pp. 247–250). Austin, TX: Morgan Kaufmann.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence, 20,* 111–161.

Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63,* 81–97.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence, 18,* 203–226.

Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), *Expert systems in the micro electronic age.* Edinburgh: Edinburgh University Press.

Quinlan, J. R. (1982). *INFERNO: A cautious approach to uncertain inference* (Tech. Rep. No. N-1898-RC). Santa Monica, CA: The Rand Corporation.

Quinlan, J. R. (1983). Learning from noisy data. *Proceedings of the Second International Machine Learning Workshop* (pp. 58–64). Urbana-Champaign, IL.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1,* 81–106.

Quinlan, J. R. (in press). Simplifying decision trees. *International Journal of Man-Machine Studies.*

Randich, A., & LoLordo, V. M. (1979). Associative and nonassociative theories of the UCS preexposure phenomenon: Implications for Pavlovian conditioning. *Psychological Bulletin, 86,* 523–548.

Razran, G. (1965). Empirical codifications and specific theoretical implications of compound-stimulus conditioning: Perception. In W. F. Prokasy (Ed.), *Classical conditioning.* New York: Appleton-Century-Crofts.

Rescorla, R. A. (1967). Pavlovian conditioning and its proper control procedures. *Psychological Review, 74,* 71–80.

Rescorla, R. A. (1968). Probability of shock in the presence and absence of CS in fear conditioning. *Journal of Comparative and Physiological Psychology, 66,* 1–5.

Rescorla, R. A. (1969). Conditioned inhibition of fear resulting from negative CS-US contingencies. *Journal of Comparative and Physiological Psychology, 67,* 504–509.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. Black & W. F. Prokasy (Eds.), *Classical conditioning II.* New York: Appleton-Century-Crofts.

Rescorla, R. A., Grau, J. W., & Durlach, P. J. (1985). Analysis of the unique cue in configural discriminations. *Journal of Experimental Psychology: Animal Behavior Processes, 11*, 356–366.

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*, 386–408.

Rosenbloom, P. S., & Laird, J. E. (1986). Mapping explanation-based generalization onto Soar. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 561–567). Philadelphia, PA: Morgan Kaufmann.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In J. L. McClelland & D. E. Rumelhart (Eds.), *Parallel distributed processing* (Vol. 1). Cambridge, MA: MIT Press.

Saavedra, M. A. (1975). Pavlovian compound conditioning in the rabbit. *Learning and Motivation, 6*, 314–326.

Schlimmer, J. C. (1986). *A note on correlational measures* (Tech. Rep. No. 86-13). Irvine: University of California, Department of Information and Computer Science.

Schlimmer, J. C., & Fisher, D. (1986). A case study of incremental concept induction. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). Philadelphia, PA: Morgan Kaufmann.

Schlimmer, J. C., & Granger, R. H., Jr. (1986a). Incremental learning from noisy data. *Machine Learning, 1*, 317–354.

Schlimmer, J. C., & Granger, R. H., Jr. (1986b). Simultaneous configural classical conditioning. *Proceedings of the Eight Annual Conference of the Cognitive Science Society* (pp. 65–79). Amherst, MA: Lawrence Erlbaum.

Schlimmer, J. C. (1987). Incremental adjustment of representations for learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 79–90). Irvine, CA: Morgan Kaufmann.

Shafer, G. (1976). *A Mathematical Theory of Evidence.* Princeton, NJ: Princeton University Press.

Shortliffe, E. H. (1976). *MYCIN: Computer-based medical consultations.* New York: Elsevier.

Siegel, S., & Domjan, M. (1971). Backward conditioning as an inhibitory procedure. *Learning and Motivation, 2*, 1–11.

Smith, E. E., & Medin, D. L. (1981). *Categories and concepts.* Cambridge, MA: Harvard University Press.

Stallman, R. M., & Sussman, G. J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided analysis. *Artificial Intelligence, 9*, 135–196.

Utgoff, P. E. (1986). Shift of bias for inductive concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Utgoff, P., & Mitchell, T. M. (1982). Acquisition of appropriate bias for inductive concept learning. *Proceedings of the National Conference on Artificial Intelligence* (pp. 414–417). Pittsburgh, PA: Morgan Kaufmann.

Vere, S. A. (1977). Induction of relational productions in the presence of background information. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 349–355). Cambridge, MA: Morgan Kaufmann.

Williamson, D., Williamson, R., & Lesk, M. E. (1971). The Cornell implementation of the SMART system. In G. Salton (Ed.), *The SMART retrieval system: Experiments in automatic document processing.* Englewood Cliffs, NJ: Prentice-Hall.

Wilson, S. W. (in press). Classifier systems and the animat problem. *Machine Learning.*

Woodbury, C. B. (1943). The learning of stimulus patterns by dogs. *Journal of Comparative & Physiological Psychology, 35,* 29–40.

Appendix A

Deriving LS and LN

Let us consider LS as a measure of the sufficiency of a matched element for expectation of a concept.[1] In terms of logical implication, this amounts to $Matched \Rightarrow Example$. Negating this we have:

$Matched \Rightarrow Example$	Sufficiency
$\neg Matched \vee Example$	Def. of Implication
$Matched \wedge \neg Example$	By negating

In order for a measure to capture sufficiency, it must be sensitive to the likelihood of $Matched \wedge \neg Example$. Therefore, a candidate measure for sufficiency might be:

$$\frac{1}{p(Matched \wedge \neg Example)}$$

However, this measure is insensitive to the likelihood that the element will match. If rarely matched, then the element's pattern is not a useful predictor. One possible refinement is:

$$\frac{p(Matched \wedge Example)}{p(Matched \wedge \neg Example)}$$

Though improved, this formula does not account for the likelihood of positive examples. Since the ratio of the two probabilities is a value between zero and positive infinity, it may be scaled by the odds in favor of a positive example, or:

$$\frac{p(Matched \wedge Example)}{p(Matched \wedge \neg Example)} / odds(Example)$$

Expanding $odds(Example)$ and inverting it yields:

$$\frac{p(Matched \wedge Example)}{p(Matched \wedge \neg Example)} \times \frac{1 - p(Example)}{p(Example)}$$

[1]This development of LS probably does not parallel the original formulation by Duda, Gaschnig, & Hart (1979), but it does attempt to give an accurate and intuitive derivation.

Recombining terms and simplifying to conditional probabilities results in Equation 2.1, the definition for LS.

$$\frac{p(Match|Example)}{p(Match|\neg Example)}$$

The key to this derivation is the inverse sensitivity of LS to $p(Example \wedge \neg Matched)$ (the essence of the implication). Useful, but of lesser importance, are the inclusion of scaling factors to accommodate the likelihood of the element matching and the likelihood of positive examples.

The derivation of LN may be motivated in a similar manner. Its essential component is the quantity $p(\neg Matched \wedge Example)$ which must be small for any necessary element.

Appendix B

Computing LS and LN

Section 2.4.1 states that the LS and LN measures can be computed via simple formulae composed of the matching event types listed in Table 2.4. This appendix shows the derivation by first substituting joint probabilities for each of the conditional probabilities and then using the counts of matching types to estimate the joint probabilities. For convenience, the conditional probability forms of LS and LN are repeated here (a duplication of Equations 2.1 and 2.2)

$$\text{LS} = \frac{p(M|E)}{p(M|\neg E)} \qquad \text{LN} = \frac{p(\neg M|E)}{p(\neg M|\neg E)}$$

The definition of conditional probability states that $p(A|B) = p(A \wedge B)/p(B)$. The above equations may therefore be rewritten as:

$$\text{LS} = \frac{p(M \wedge E)}{p(E)} \times \frac{p(\neg E)}{p(M \wedge \neg E)} \qquad \text{LN} = \frac{p(\neg M \wedge E)}{p(E)} \times \frac{p(\neg E)}{p(\neg M \wedge \neg E)}$$

Noting that the outcome is either a positive or negative example and the feature is either a matched or an unmatched element, the above probabilities may be approximated with counts of the situation types in Table 2.4 as follows:

$$\begin{aligned}
p(M \wedge E) &\approx \text{C}_\text{P}/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N}) \\
p(M \wedge \neg E) &\approx \text{I}_\text{N}/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N}) \\
p(\neg M \wedge E) &\approx \text{I}_\text{P}/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N}) \\
p(\neg M \wedge \neg E) &\approx \text{C}_\text{N}/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N})
\end{aligned}$$

Marginal probabilities of the form $p(A)$ may be similarly approximated by noting that $p(A) = p(A \wedge B) + p(A \wedge \neg B)$.

$$\begin{aligned}
p(M) &\approx (\text{C}_\text{P} + \text{I}_\text{N})/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N}) \\
p(\neg M) &\approx (\text{I}_\text{P} + \text{C}_\text{N})/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N}) \\
p(E) &\approx (\text{C}_\text{P} + \text{I}_\text{P})/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N}) \\
p(\neg E) &\approx (\text{I}_\text{N} + \text{C}_\text{N})/(\text{C}_\text{P} + \text{I}_\text{P} + \text{I}_\text{N} + \text{C}_\text{N})
\end{aligned}$$

Substituting these approximations yields:

$$LS = \frac{\frac{C_P}{C_P + I_P + I_N + C_N} \times \frac{I_N + C_N}{C_P + I_P + I_N + C_N}}{\frac{I_N}{C_P + I_P + I_N + C_N} \times \frac{C_P + I_P}{C_P + I_P + I_N + C_N}}$$

$$LN = \frac{\frac{I_P}{C_P + I_P + I_N + C_N} \times \frac{I_N + C_N}{C_P + I_P + I_N + C_N}}{\frac{C_N}{C_P + I_P + I_N + C_N} \times \frac{C_P + I_P}{C_P + I_P + I_N + C_N}}$$

Simplifying by canceling out $C_P + I_P + I_N + C_N$ results in Equation 2.6 presented in Section 2.4.1.

$$LS = \frac{C_P(I_N + C_N)}{I_N(C_P + I_P)} \qquad LN = \frac{I_P(I_N + C_N)}{C_N(C_P + I_P)}$$

Appendix C

Information Retrieval Data

This appendix contains the document-keyword pairs used in Section 3.4. Each document was drawn from the annual proceedings of the national conference on artificial intelligence and is represented by the author's last names, the year of the proceedings, and the starting page number of the document. Following this identifier are a set of keywords that describe the document, including the name of the system (if any), title words, and terminology introduced in this paper.

Dietterich-1980-218: data-interpretation, decomposition, disjunction, elusis, game, induction, knowledge-base, learning-from-examples, model-fitting, periodic-function, rule-evaluation, segmentation.

Smith-1980-221: application-heuristic, computer-aided-instruction, instructional-strategy, learning-by-teaching, logic, macro-operator, problem-solving, redhot, student-learning, student-modeling, theorem-proving.

Selfridge-1980-224: inference-rule, language-learning, learning-rule, modeling.

Rychener-1980-228: analogy, ips, knowledge-acquisition, problem-space, production-memory, production-system, schema, semantic-network, working-memory.

Reboh-1980-231: control-strategy, expert, goal, knowledge-acquisition, knowledge-base, matching, partial-matching, prospector, search, semantic-network.

Haas-Hendrix-1980-235: expert, klaus, knowledge-acquisition, knowledge-base, knowledge-representation, learning-by-being-told, natural-language, tutors.

Whitehill-1980-240: generalization, isa-generalization, language-learning, learning-from-examples, maximal-common-generalization, production-rule, rule-containment.

Granger-1980-301: arthur, goal, natural-language, parsimony, plan, script, self-correcting-inference, supplanting-inference.

Lebowitz-1980-324: episodic, generalization, ipp, long-term-memory, memory-organization, mop, natural-language, specialization.

Kolodner-1980-331: cyrus, episodic, event-category, generalization, memory-organization, memory-retrieval, mop, natural-language.

Keirsey-1982-99: deduction, expectation-based-understanding, induction, isa-hierarchy, knowledge-base, language-learning, natural-language.

Korf-1982-164: breadth-first-search, game, goal, macro-operator, problem-solving, puzzle.

Carbonell-1982-168: analogy, experiential-learning, induction, learning-from-examples, plan-learning, problem-solving, skill-learning.

Rosenbloom-Newell-1982-255: chunking, power-law-of-practice, psychological-modeling, xaps2.

Abe-Itoh-Tsuji-1982-362: analogy, learning-from-examples, matching, natural-language, object-model-learning, story-understanding, vision.

DeJong-1982-410: explanation, explanatory-schema-acquisition, generalization, knowledge-base, natural-language, problem-solving, schema.

Utgoff-Mitchell-1982-414: analytic, backward-propagation, bias, concept-learning, generalization, induction, learning-from-examples.

Carbonell-1983-64: analogy, derivational-analogy, problem-solving, transfer, transformation.

Kibler-Porter-1983-191: concept-tree, episodic, generalization, macro-operator, operator, perturbation, pet, problem-solving, production-rule.

VanLehn-1983-420: knowledge-communication, knowledge-compilation, knowledge-integration, natural-language, procedural-learning, psychological-modeling, repair, sierra, step-theory.

Benjamin-Harrison-1983-22: expert, generalization, genex, goal-tree, knowledge-representation, learning-from-examples, meta-production, plan-learning, plex, production-system, working-memory.

Korf-1983-206: macro-operator, operator-decomposability, problem-solving, puzzle, serial-decomposability.

Smolensky-1983-378: computational-temperature, connectionist, harmony, perceptual-grouping, schema, schema-selection, stochastic-inference.

Lenat-Seelybrown-1983-236: am, automatic-programming, eurisko, functional-decomposition, heuristic-learning, interestingness, learning-by-discovery.

Winston-Binford-Katz-Lowry-1983-433: analogy, censor, constraint-transfer, functional-definition, learning-from-examples, precedent, vision.

Mostow-1983-279: advice, bar, foo, means-ends-analysis, operationalization, plan, problem-space, problem-solving.

Salzberg-1983-352: causal, explanation, handicapper, hypothesis-generation, prediction.

Keller-1983-182: generalization, operationalization, problem-solving, recognition, specialization.

Scott-1983-359: nature-of-learning, posteriori-knowledge, value-of-learning.

Rendell-1983-343: blackboard, cluster, critic, genetic, penetrance, pls1, pls2, state-space.

Pettit-Swigger-1983-327: genetic, mempop, pattern-tracking, psychological-modeling, random, statistical-predictive.

Hammond-1983-148: case-based, episodic, goal, interaction, memory-indexing, memory-organization, plan, plan-modification, psychological-modeling, retrieval, wok.

Burstein-1983-45: analogy, carl, causal, debug, mapping, psychological-modeling, target.

Lebowitz-1983-232: generalization, knowledge-representation, long-term-memory, memette, natural-language, question-answering, researcher.

Andrea-1984-6: constraint-based, event-generalization, functional-learning, generalization, induction, learning-from-examples, procedural-learning.

Araya-1984-11: experiment, generalization, isaac, problem-solving.

Dietterich-1984-96: eg, iterative-extension, program-systhesis, state-variables.

Laird-Rosenbloom-Newell-1984-188: chunking, game, generalization, goal, macro-operator, memory-organization, problem-solving, problem-space, procedural-learning, puzzle, soar.

Langley-Ohlsson-1984-193: acm, credit-assignment, learning-from-examples, learning-from-solution-paths, operator, problem-reduction, problem-space, production-system, proposer, psychological-modeling, specialization.

Mauldin-1984-247: convergence, genetic, optimization, search.

Minton-1984-251: constraint-based, deduction, game, generalization, learning-from-examples, plan, puzzle.

ORorke-1984-260: dependency-relations, explanation-based, generalization, knowledge-based, macro-operator, problem-solving, schema.

Porter-Kibler-1984-278: episodic, opaque-representation, operator, perturbation, pet, problem-solving, relational-learning, transparent-representation.

Levinson-1984-203: knowledge-base, memory-organization, partial-matching, retrieval.

Alterman-1986-65: adaptive-planning, background-knowledge, causal, generalization, partonomic, plan, plexus, role, situation-matching, specialization, taxonomic.

Christensen-Korf-1986-148: evaluation-function, game, heuristic-learning, problem-solving, regression, search.

Dechter-1986-178: backtracking, constraint-satisfaction, csp, first-order-learning, search, second-order-learning, shallow-learning.

Hammond-1986-267: case-based, chef, credit-assignment, plan, repair.

Ginsberg-1986-436: causal, generalization, knowledge-based, knowledge-representation, meta-knowledge, meta-linguistic, seek2.

Lee-Ray-1986-442: expert, perturbation, prg, probabilistic, refinement, rule.

Wilkins-Buchanan-1986-448: antidote-algorithm, debug, heuristic-inference, reasoning, rule, uncertainty.

Kokar-1986-455: coper, dimensional-analysis, discovery, functional-learning, knowledge-representation, quantitative-law-learning.

Flann-Dietterich-1986-460: bias, envisionment, functional-learning, generalization, induction, knowledge-representation, learning-from-examples, wyl.

Laird-1986-472: generalization, induction, inference, refinement, schema, specialization.

Russell-1986-477: analogy, base-level-induction, bias, determination, enumerative-induction, generalization, induction, projectability, search.

Haussler-1986-485: bias, concept-learning, convergence, disjunction, efficiency, induction.

Cheng-Carbonell-1986-490: algebra, fermi, induction, macro-operator, problem-solving.

Schlimmer-Fisher-1986-496: efficiency, id3, id4, incremental, induction, search.

Schlimmer-Granger-1986-502: concept-drift, conjunction, disjunction, generalization, incremental, matching, negation, noise, search, specialization, stagger.

Nordhausen-1986-508: cluster, opus, relational-learning.

Jones-1986-513: cluster, discovery, experiment, nglauber, quantitative-law-learning.

Subramanian-Feigenbaum-1986-518: discrimination-experiment, experiment, factorization, incremental, vs.

Bain-1986-523: case-based, experience-based, generalization, judge, reasoning, retrieval, subjective-assessment.

Rose-Langley-1986-528: assumption-based, belief-revision, componential-models, discovery, stahlp.

Lebowitz-1986-533: causal, explanation-based, generalization, similarity-based.

Doyle-1986-538: abstraction, causal, causal-learning, domain-theory, experiment, explanation, explanation-based, generalization, inconsistent, refinement.

Pazzani-Dyer-Flowers-1986-545: causal, correlational, event, generalization, memory-organization, occam.

Mooney-Bennett-1986-551: equalities, explanation, explanation-based, explanation-structure, generalization, goal.

Hammond-1986-556: causal, chef, explanation, failure-driven, plan.

Rosenbloom-Laird-1986-561: chunking, concept-formation, explanation-based, generalization, learning-from-examples, soar.

Hall-1986-568: explanation, explanation-based, failure-driven, generalization, precedent, precedent-analysis.

Eshelman-McDermott-1986-950: expert, heuristic-classification, inference-engine, knowledge-acquisition, knowledge-base, mole, problem-solving.

Lathrop-Kirk-1986-1024: constellation, database, design-precedent, knowledge-acquisition, learning-from-examples, structure-learning.

Pazzani-1986-1029: aces, deduction, dependency-directed-backtracking, diagnosis, expert, failure-driven, induction, knowledge-base.

Shimura-Sakurai-1986-1036: arithmetic, laps, natural-language, problem-solving, production-rule.

Michalski-Mozetic-Hong-Lavrac-1986-1041: analogy, aq15, constructive-induction, disjunction, generalization, incremental, induction, knowledge-base, logic, medical, noise, partial-matching, specialization.

Hinton-1986-1149: back-propagation, boltzmann, connectionist, parallel, perceptron.