# MODELING CONCEPT ACQUISITION IN THE CONTEXT OF A UNIFIED THEORY OF COGNITION

by

Craig S. Miller

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
1993

Doctoral Committee:

Associate Professor John E. Laird, Chair
Assistant Professor Steven L. Lytinen
Professor William C. Rounds
Assistant Professor Colleen M. Seifert
Professor Douglas L. Medin, Northwestern University

# ABSTRACT

MODELING CONCEPT ACQUISITION IN THE CONTEXT OF A UNIFIED THEORY OF COGNITION

by
Craig S. Miller

Chair: John E. Laird

This dissertation presents a process model of human learning in the context of supervised concept acquisition. I describe SCA (Symbolic Concept Acquistion), a computational model that acquires and retrieves category prediction rules. SCA is a fully symbolic approach that retrieves prediction rules searching from specific to general. From training examples, it acquires general rules but then incrementally learns more specific ones.

SCA's plausibility as a model of human learning is motivated by its functionality, its ability to replicate human behavior and its constitution within a unified theory of cognition. Functionally, SCA meets task demands by performing incremental, noise-tolerant learning within real time. For replicating human behavior, SCA mirrors psychological data along the dimensions of response time, accuracy and learning rate as a function of various instance and category structures including typicality, linear boundedness, category level and definitional complexity. For its role within a unified theory of cognition, SCA's learning principles are composed from the basic processing mechanisms posited by Soar, a candidate theory.

# ACKNOWLEDGEMENTS

First, I would like to thank my advisor, John Laird. John has allowed me to generate my own ideas and approaches while offering me the appropriate amount of guidance necessary to bring them to fruition. In that he has read and commented on almost every paper I wrote in graduate school, I probably have learned how to write from him more than anyone else. On a wider scale, John's integral involvement with the Soar project has brought me many valuable opportunities including contact with prominent faculty from other universities, discussion forums and speaking opportunities at Soar workshops. On a much more frequent basis, our local Soar group has been particularly helpful in exchanging current ideas.

I would also like to thank the remaining members of my committee, all of whom gave useful comments and suggestions from earlier papers. They include Steve Lytinen, who got me started in research; Bill Rounds, who encouraged me to understand the formal consequences of my work; Doug Medin, who directed me to useful results in psychology; and Colleen Seifert, who introduced me to the 'big issues' in cognitive science during those early seminars.

I would like to thank all of my friends, of whom the closest and the vast majority I met while in graduate school. They've been key in keeping my work and career within the proper perspective. I'd like to thank those that introduced me to ultimate frisbee, which helped me keep my nerves through the last several years. I would especially like to thank my friend Miriam, who, in addition to simply being a great friend, patiently listened to my worries and concerns of all the work I still had to do.

For my financial support, I would like acknowledge and thank the following sources of support: the W.K. Kellogg Foundation through The Presidential Initiative Fund of The University of Michigan, the Nissan Corporation, and NASA Ames through grant NCC2-517.

Finally, I would like to thank my family, who has been the true constant source of encouragement and support throughout graduate school.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# INTRODUCTION

In this thesis, I present a computational model of human category learning. In particular, it is a model of a human performing a supervised learning task in an experimental setting. In a sense, it is a very modest endeavor, especially when one considers the wide variety and complexity of learning that people do in almost every situation they experience. For example, they can learn complex procedural tasks, such as in sports or in the performing arts. They can also learn a language, and with that skill, learn complex declarative knowledge from instruction and reading.

In performing the supervised learning task, the subject receives examples of objects consisting of distinguishing features, (*e.g.* round, blue, and striped), and is told the category for each object. From the examples, the subject learns to predict categories of unclassified examples. Although the task of learning artificial categories of objects made of abstract features certainly pales in comparison to the vast varieties of learning humans perform on a daily basis, people can and do perform the tasks of simple learning experiments, and, despite their simplicity, these tasks require cognitive abilities fundamental to many learning tasks. First of all, subjects must learn incrementally, *i.e.* they often see only one learning example at a time. Secondly, they must be able to recover from mistakes. Finally, they must be sensitive to the frequencies of learning examples. Subjects' behavior, defined in terms of accuracy, learning rate, and response time, are sufficiently complex so as to remain a challenge for comprehensive computational modeling. Many models exist for isolated tasks and types of behavior, but to date, there are no models that cover more than a handful of the results from supervised learning experiments. Furthermore, previously posited models have been developed outside the context of any larger, coherent system which covers the whole range of cognition.

The challenge is to create a model such that it not only covers a broad range of behavior, but also sits within the confines of a more general and comprehensive theory of cognition. The work in this thesis goes after this challenge, by developing and evaluating a process model of supervised category learning. For evaluating the model, I cite three principles that process models must at least obey:

**Principle 1** *The model must be functionally motivated.*

**Principle 2** *The behavior of the model must replicate the behavior of subjects performing the same task.*

**Principle 3** *The model must function within a coherent framework motivated by functionality and general cognitive phenomena.*

The first two principles are well entrenched within cognitive science while the third is now emerging. Despite their acceptance, I make all three explicit and further elaborate on them so that the model can be explicitly evaluated on these terms.

## 1.1 Functionality

The model is expected to be functionally motivated. Assuming human cognition is a product of competitive evolution, any learning strategy must be justified by its functional advantages. There should not be extraneous modules or strategies which provides no conceivable advantage to task performance. One example is an attentional mechanism which focuses on some features of an object at the expense of other features. While the addition of such a mechanism may make the model's behavior correspond better to human behavior, the mechanism's plausibility should be in doubt unless there is reasonable evidence that the mechanism either improves performance or economizes processing resources.

In presenting the model, design decisions are presented in terms of their functional advantage. For my learning task, functionality can be measured as performance along the dimensions of accuracy, response time, learning rate, noise tolerance, and required computational resources. Determining the functional advantage of a specific design decision is currently more of an art rather than a science because of the potential interactions between different components of a model and the diversity of tasks a human must perform. The approach taken here calls for locally motivating each module in explaining its functional role. Shortcomings of this approach are balanced by empirically evaluating the model in terms of performance, showing that it is competitive compared to existing machine learning systems.

## 1.2 Replicating human behavior

Replicating human behavior is probably the most common guiding principle in cognitive science. There are many ways external behavior expresses itself in supervised learning tasks. In this thesis, I focus on three dimensions of behavior: accuracy, response time, and acquisition rate. An example of human behavior on all three dimensions is typicality effects. Humans learn to classify typical examples (*i.e.* examples that are most similar to other examples in the same category) more accurately, with faster response times, and with less training examples.

The claim is that the model covers a significant subset of known human phenomena that no other model does. In particular, it exhibits several response time behaviors which few other models adequately address. Furthermore, these models are limited in breadth and have not addressed the other phenomena discussed in this thesis.

The model is intended as a *process* model and thus offers a *mechanistic theory* of human learning. Furthermore, the model addresses behavior at what Newell calls the *cognitive band* [Newell, 1990]. This band includes behavior occurring between the range of 100 milliseconds to 10 seconds. At this level, the goal of modeling behavior is not only to explain human actions, but also to explain the timing of those actions. The process model gives a strong account of timed behavior since it is formulated in terms of sequences and iterations of well-specified mechanisms. In this case, the process is described in terms of sequences of abstract mechanisms operating at approximately 50 milliseconds. This figure is not an arbitrary parameter. Instead, it has been empirically ascertained in other models [Wiesmeyer and Laird, 1990; John, 1988] and is consistent with timing constraints afforded by the brain's neural hardware [Newell, 1990]. Thus the model offers an account of response times varying on the order of 100 milliseconds or more. This is appropriate for modeling behavior at the cognitive band (100 ms – 10 sec). In the case of category learning, human response times are between 1 and 10 seconds depending on the stimuli, and vary by hundreds of milliseconds as a function of typicality [Rosch *et al.*, 1976b].

## 1.3 Fit within a unified theory of cognition

Finally, an effective learning capability does not act in isolation. Ultimately it interacts with other cognitive capacities in delivering a full interactive autonomous system. It must also be grounded in the computational resources afforded by the human brain. A unified theory of cognition (UTC), as embodied in the form of an architecture, posits what these resources are. They are characterized as

a set of mechanisms that are capable of supporting intelligent behavior. The approach to following this principle thus uses a cognitive architecture to guide the model's design. The architecture of my choice, Soar, is a candidate for a unified theory of cognition and has already proven itself with many diverse tasks [Newell, 1990].

The assumption is that, by having a UTC guide the model's design, the resulting model is an embodiment of a more plausible theory. The extent to which the model is more plausible is determined by the extent to which the properties of the UTC mechanisms are the same properties of the human mind. In the case of Soar, we know that at least three general properties must be the same:

- **Computationally tractable.** Soar's fundamental mechanisms are computationally feasible. Special attention has been given so that the architecture can run in real time for an extensive amount of time and acquire an extensive amount of knowledge [Doorenbos *et al.*, 1992; Tambe and Newell, 1988].
- **Well-defined.** Just as the mind is well-defined in terms of a biological realization in the brain, so is the Soar architecture in terms of a computer implementation.
- **Globally evident in behavior.** Some behavioral phenomena are empirically evident across a wide variety of tasks. Newell (1990, p. 227) presents a list of human behavior that also agree with the characteristics of Soar.

It is also a reasonable assumption that the following properties are generally shared:

- **General-purpose.** Soar's mechanisms are all strongly motivated by their ability to support a wide variety of tasks. Certainly, at least some of the brain's architectural mechanisms are general purpose, as evidenced by the wide variety of human capabilities at the cognitive band. The existence of task-specific mechanisms is also not plausible given the timing disparity between the high speed of task-specific hardware-supported capabilities and the slower speed of behavior at the cognitive band. Finally, it is unlikely that hardware-supported mechanisms exist which, when composed with other mechanisms, only support a limited range of tasks. For example, it is doubtful that humans have special hardware for computing floating point arithmetic which can then only be applied to balancing checkbooks.
- **Integrated.** Soar's mechanisms do not operate in isolation. Rather, they share common data structures and mesh together in performing a variety of tasks. It is likely that the mechanisms afforded by the brain are also well integrated. If not, costly operations for decoding and recoding data structures are required in order to share knowledge across mechanisms.
- **Functional.** Soar's fundamental mechanisms are fully motivated by their functionality. There are no non-functional entities in the architecture. Just as functionality is a useful constraint for cognitive models, so is it for architectural mechanisms.

## 1.4 Organization of dissertation

The remainder of the dissertation focuses primarily on the presentation and evaluation of the model. The dissertation's chapters are structured as follows:

- **Chapter 2** discusses in greater detail the supervised learning task that the model addresses. The chapter places the task in the context of other systems performing the same task and other related tasks.
- **Chapter 3** presents the model. Here I also describe how the UTC (Soar) theory shaped its design and how the model is significantly different from previous approaches.
- **Chapter 4** evaluates the model in terms of functional performance. An empirical comparison is made with two other machine learning approaches.

- **Chapter 5** evaluates the model in terms of replicating human behavior. Qualitative comparisons are made with results from psychological experiments.
- **Chapter 6** presents future research directions. In particular, there is a focus on resolving problems with the current model. Some promising solutions and preliminary evaluations are discussed.
- **Chapter 7** summarizes the contributions and implications of the thesis.
- **Appendix A** describes in greater detail how the model's design arises from the Soar architecture.
- **Appendix B** presents a formal specification of the concepts represented by the model.

# Chapter 2

# THE SUPERVISED LEARNING TASK

My model performs a supervised learning task. The system is presented with training examples,[1] described in terms of attributes and symbolic values, and a category label. The task is then to predict the category for future examples that do not have the label. For example, the following series of training examples may be presented to the system:

```
{shape:spherical, color:blue, texture:smooth, size:small; category:ball}
{shape:oblong, color:red, texture:smooth, size:medium; category:ball}
{shape:spherical, color:blue, texture:smooth, size:large; category:globe}
```

These are training examples since they include both the object description and the category. With these examples, the system learns to predict categories when given only an object description, such as

```
{shape:spherical, color:green, texture:smooth, size:medium}
```

Here the system might respond with the category 'ball'.

Supervised learning from examples has been a popular task for many machine learning systems; these include systems based on discrimination trees [Quinlan, 1986; Breiman *et al.*, 1984; Schlimmer and Fisher, 1986; Utgoff, 1988], logical concept descriptions [Michalski, 1983], artificial neural nets [Rosenblatt, 1962; Rumelhart *et al.*, 1986b], genetic classifiers [Holland and Reitman, 1978], and stored instances [Aha *et al.*, 1991]. Likewise, work in psychology has produced models based on similar representations including discrimination nets [Feigenbaum and Simon, 1984], rules [Anderson *et al.*, 1979], neural nets [Kruschke, 1990; Gluck and Bower, 1988a] and instances [Medin and Schaffer, 1978; Hintzman, 1986].

I further constrain the task so that the learning must be incremental. Informally, this means that the model can perform (*i.e.* predict categories for unlabeled examples) at intermediate stages of learning. This fulfills a psychological behavioral constraint in that humans incrementally learn throughout their lives.

Technically any learning system can be trivially modified to pass as an incremental learner given this informal definition. All that is required is that the system save all training examples and then recompile its prediction procedure whenever new training examples are presented. In order to exclude approaches that require a ridiculous expense of computational resources every time additional training examples are provided, I present a more formal definition.

**Definition 1** *A learning system is incremental if it can perform at any intermediate stage of learning and if the processing of a training example has a constant time complexity,* i.e. *O(1), with respect to the total number of training examples already encountered.*

[1] The terms *example*, *instance*, and *object* are used interchangeably in this dissertation.

Loosely stated, an incremental learning system does not require an increasing amount of computational time as more examples are encountered. This definition excludes approaches that must recompile their prediction procedure upon the introduction of every new training example. In clarifying the definition, I note that the constant time complexity is *with respect to the total number of training examples already encountered.* This does not mean that the processing of a training example will always take the same time. Some examples may require more processing time than others. In these cases, the processing time may be a function of some other factor such as typicality or description size. The definition only excludes the case where processing time grows with the amount of experience.

This is an important functional constraint on behavior since it recognizes the advantage of not having to require more computational resources to learn as more knowledge is acquired. When considering that a human may encounter millions (if not billions) of learning examples, we can safely presume that human learning must at least approach this constraint.

The requirement that the model learns incrementally constitutes a real-time constraint on training examples. Similarly, the task also demands a real-time constraint on performance. The model must be able to respond with a category in a reasonable amount of time. For the psychological experiments with which I compare the model's behavior, subjects typically have only a couple of seconds to respond.

This specific supervised learning task does not include all of the complexities that a human can face in learning. For example, in most real-world learning situations, the object that is serving as a training example must be separated from the background of the total environment. For my model, examples and labels come pre-identified and symbolically encoded. Also, the model's examples are represented by flat symbolic structures, whereas many more complex tasks require numeric and structured object descriptions. However, even with these simplifications, there are no comprehensive models that cover a broad range of human learning for these tasks, and clearly, humans can and do perform these tasks. Many psychological experiments make the same representational assumptions, and despite the simplification, many interesting robust learning phenomena are still observed.

Another consideration is that learning with flat, symbolic structures can still serve in learning with more complicated representations. Indeed, methods exist that convert numeric data to symbolic representations that can then be used with purely symbolic learning approaches [Fayyad, 1991] and approaches to learning structured objects could rely on learning composites of flat representations. Finally, the choice of defining the model within the context of a unified theory helps us understand how a simple, "trivial" task can rely on more general cognitive mechanisms.

## 2.1 Relation to Unsupervised Learning

There exist other tasks which are directly related to supervised learning. In unsupervised learning, training instances are object descriptions with no separate category label. Although a category label may be included in the object description, it does not take on a special status as being the feature which the model must be able to predict. Usually, the immediate goal is to organize the examples into conceptual clusters. Examples of these systems in machine learning include Cluster/2 [Michalski and Stepp, 1983], COBWEB [Fisher, 1987], UNIMEM [Lebowitz, 1987], AutoClass [Cheeseman *et al.*, 1988], CYRUS [Kolodner, 1983], and WITT [Hanson and Bauer, 1989]. Models from psychology include Anderson's rational model [Anderson, 1991] and Ahn and Medin's two-stage model [Ahn and Medin, 1992]. Except for AutoClass and the two-stage model, all of these learn incrementally and thus require occasional dynamic structural re-organizing as new examples are encountered.

For the unsupervised systems, the end-task is usually not to simply cluster examples. Rather, the ultimate goal is to use the conceptual organization to predict the values of features missing from test examples [Fisher, 1987; Anderson, 1991]. The process is simple. When given an object description with a missing feature, the system places the object in the best cluster. Once the object has been categorized, the system checks other examples in the same cluster for their values of the missing feature. The system predicts the prevailing value in the cluster. By presenting training examples whose descriptions include the category name, an unsupervised learning system

can predict categories for examples missing the category name. Thus, unsupervised learning systems can perform the supervised learning task.

From the perspective of the end-task, the only critical difference between supervised learning and unsupervised learning is that a supervised learning system is trained to predict one particular feature (*e.g.* the category name) whereas an unsupervised learning system is trained with the future goal of being able to predict any feature which might be missing from a future example.

The conceptual clustering paradigm is not the only means of learning to predict any feature. For example, auto-associative nets [Rumelhart *et al.*, 1986a] perform the same task without explicitly clustering training examples. Furthermore, any supervised learning strategy can be trivially adapted to perform unsupervised learning, by applying the learning strategy to each feature in the training example. Certainly this strategy is a far less economical means of organizing conceptual information. However, the computational complexity is only increased as a function of the number of features in object descriptions and not in terms of the number of categories or the number of examples.

## 2.2 Relation to other tasks

For this thesis, supervised learning is a naming task. That is, the goal is to name the category (or any other feature that the model is instructed to learn). There are other category-based tasks that the model I present does not address. These include verifying category membership, producing an example of a category, judging the goodness of category membership, and verifying taxonomic relationships. All of these require extensions to the basic model and thus a thorough treatment is not provided. However, Chapter 6 discusses architecturally motivated extensions of the model that address each of these tasks.

## 2.3 Relation to Learning Theory

Recently, the supervised learning paradigm has received some attention in the theoretical learning community. Despite the contributions to defining what is learnable [Valiant, 1984], the role of inductive bias [Mitchell, 1990; Haussler, 1988], and how to quantify bias [Vapnik, 1982], the work is at best tangential to the work of this thesis. First of all, the assumptions of the task differ significantly. The theoretical work assumes a non-incremental learning protocol with training data that is noise-free [Haussler, 1988] whereas the task defined for this thesis requires an *incremental* learning model that can learn from data with at least some noise (as humans clearly can). Secondly, and more importantly, the theoretical contributions do not (and perhaps cannot) help determine whether a particular inductive bias is a good bias. The problem of determining a useful bias is perhaps more of an empirical issue, dependent upon the structure of the world rather than a theoretical result. The work in this thesis addresses this problem, of identifying a good bias, by understanding and modeling learning biases present in human behavior.

# Chapter 3

# DESCRIPTION OF THE MODEL

In this chapter, I describe a rule-based model of supervised category learning. An emphasis is placed on the model's *process*, including how the model retrieves rules for category prediction and how the model acquires new rules to improve prediction accuracy and speed. The model specification is precise enough to realize a computational implementation. However, amidst the implementational details, I focus on a set of cardinal structural properties, which represent long-term design commitments. It is essentially these properties that fit within a unified theory of cognition and account for the model's interesting behavior.

The supervised learning model can be described in two parts. The first part is the basic mechanism that learns and retrieves prediction rules. The prediction rules test for specific attributes and values in examples. A rule that matches an example predicts a category. The second part learns heuristics guiding the selection of which attributes and values should be tested by the prediction rules. A rough analogy of this separation can be made with decision trees [Quinlan, 1986] where the decision tree representation corresponds to the rule learning and retrieval mechanism, and the splitting rule corresponds to feature selection learning. This section separately describes both parts of the model. The rule learning and retrieval mechanism, called SCA (symbolic concept acquisition), is the major novel contribution of this work. However, in order to empirically evaluate SCA, I also present the second mechanism, which is required to functionally complete the model.

For now, I describe SCA outside the context of Soar (the UTC). However, later in the chapter, I discuss how Soar has motivated the design of SCA and how the structural design of SCA differs from previous approaches. Appendix 7 describes the Soar implementation of SCA in more detail.

## 3.1 Rule retrieval and acquisition

### 3.1.1 Representing concepts as rules

In general terms, SCA is a symbolic rule-based system, that incrementally acquires prediction rules as it is trained. At first, SCA learns very general rules that test only a few features (attribute-value pairs) of an example, but as learning progresses, more specific rules are acquired that test more and more features. Thus, there may be many rules of different levels of specificity (and correctness) that predict the same category. In trying to predict the category of an example, SCA attempts to find the most specific prediction rules that match the example.

The examples SCA accepts for prediction are described in terms of symbolic attributes and values. For example, an example may be described as follows:

```
{shape:spherical, color:blue, texture:smooth, size:small}
```

Upon receiving an example, whether it be for prediction or training, SCA internalizes the example description. In order to distinguish between an example presented to SCA and the internal representation of the example, I use brackets in depicting the internal representations:

```
[shape:spherical, color:blue, texture:smooth, size:small]
```

SCA's rules test for features and predict categories. Some are very general (for the following examples, the attribute names are omitted):

```
[spherical] --> predict category:ball
[spherical] --> predict category:globe
```

Others are more specific:

```
[spherical, red] --> predict category:ball
[spherical, blue] --> predict category:globe
[spherical, red, smooth] --> predict category:ball
```

### 3.1.2 Predicting with rules

As would be expected, the more specific prediction rules are more likely to be correct, and thus for prediction, SCA attempts to find the most specific rule that matches the example description. In particular, the process takes the example description and then checks if there are any rules that match all of its features. If none exist, it then removes a feature from the example description and checks if there are any matches on all of the remaining features. In the example, the description might be modified by removing the texture attribute giving:

```
[shape:spherical, color:blue, size:small]
```

This process of removing a feature and then checking for a match continues until either at least one prediction rule matches or until there are no features left. If no rules match, then no prediction can be made until more prediction rules are learned. If a single rule matches, then its prediction is made. In our example, after removing the size attribute, the system would predict `category: globe`. If several competing rules match at the same time, the system arbitrarily guesses from among one of the competing predictions.

Many concept acquisition systems use rules to encode their knowledge about concepts and some have a bias toward picking more specific rules. Independent of learning, how and why is SCA's representation of concepts different?

First, SCA's search from specific to general can be controlled by knowledge which determines which feature to remove next, so that the determination of specificity is not purely syntactic, and is itself subject to learning. Second, even though SCA maintains a large set of prediction rules at various levels of specificity, the computational resources required to match an example at a given level of specificity against all rules is constant. All of the description's features must match exactly for a rule to apply, so that a simple hashing scheme[1] can be used for rule indexation. Thus, this avoids the rule-utility problem [Minton, 1988] where match time grows with the number of rules in memory. The total amount of time taken to perform a prediction is only influenced by the number of attributes that need to be abstracted in order to find a match, and as we shall see, as more rules are learned, the number of abstractions will actually *decrease*, thus decreasing the time required to perform a prediction. This computational efficiency comes at some price, for there is no guarantee that SCA will find the most specific rule in its rule-base that can match the current example. A more specific rule may be overlooked if the order of feature removal is different from when the rule was learned.

Figure 3.1 provides the specification of the performance process written in pseudocode. Here the function *Predict* is passed a set of features serving as the example description. The example description $D$ is incrementally stripped of its features until a match is found. The function *recall* returns the predictions of all rules whose conditions match its argument.

[1] Hashing is a common technique for efficiently accessing a data object. On average, access time is constant with respect to the number of stored objects [Aho *et al.*, 1974].

```
define Predict(D)
   do
      set R = recall(D)
      if size(R) > 0 then
         set Match = true
         set p = choose element from R
      else
         set D = D - Select_Feature(D)
   until Match or size(D) = 0

   return(p)
end Predict
```

Figure 3.1: Performance specification

### 3.1.3 Rule acquisition

When learning rules, SCA accepts an example description *that includes a category label*, and attempts to make a prediction for the example using the procedure described above. In contrast to prediction, during learning the search does not necessarily stop with the first-matched rule. Instead, search continues until a matching-prediction rule makes the *correct* prediction, or until no features are left in the example description. With a match and a correct prediction, the system has thus discovered prior experience that supports the current training example. The training example now serves as new knowledge for adding an additional rule.

For acquiring a new rule which represents a compromise between previously acquired knowledge and the knowledge implicit in the training example, SCA follows a simple strategy. It acquires a new rule whose conditions include all of the features that matched (or no features if no match occurred) *plus the feature that was last removed before the search stopped.* The prediction of the new rule is the correct category given by the training example, which also had been confirmed by the matching rule.

Let us use the training example `ball: {spherical, blue, fuzzy, small}` as an example of how a new rule is acquired. First, the description `[spherical, blue, fuzzy, small]` is processed in search for a category prediction. Given current feature ordering heuristics, 'small' is removed and then 'fuzzy'. The description `[spherical, blue]` matches a prediction rule. However, this rule predicts 'globe'—the wrong category. Search continues by removing 'blue'. Finally, the description `[spherical]` matches a correct rule and search stops. A new rule is constructed and added to memory:

```
[spherical, blue] --> predict category:ball
```

With the acquisition of this new rule, there are now two competing rules with these attributes at this level of specificity. Should both of these rules match during performance, a guess is required in order to make a prediction. Typically, however, the acquisition of this new rule is merely an intermediate step towards the acquisition of still more specific ones. Eventually, with enough training examples, sufficiently specific rules are acquired so that conflicts among them become quite rare.

Figure 3.2 provides the specification of the performance process modified for training. I have added the *store* function which saves new rules whose conditions include the previous example description, *D'*.

Learning using this method is incremental. The computational effort to process a training example does not increase with the number of previously learned rules, but on average, will decrease, as fewer and fewer abstractions need to be performed before a correct prediction can be found and a single new rule is added to memory.

The choice of adding only one rule with only one more feature in the condition represents a compromise between previously obtained knowledge and the knowledge implicit in the newly presented

```
define Train(D,c)
   set D' = D
   do
      set R = recall(D)
      if c in R or size(D) = 0
         set Match = true
         set p = c
         store(D' --> c)
      else
         set D' = D
         set D = D - Select_Feature(D)
   until Match or size(D) = 0

   return(p)
end Train
```

Figure 3.2: Performance specification modified for training

training example. Should the previously obtained knowledge be incomplete, *i.e.* the prediction conditions do not include all relevant features, the addition of a new rule with an extra feature in the condition helps complete the system's knowledge. On the other hand, should the knowledge implicit in the training example be irrelevant (due to spurious correlations) or incorrect (due to noise), the addition of only one more rule with only one additional conditional feature allows room for error recovery as more examples are experienced. Error recovery automatically occurs with the presentation of additional training examples which incrementally lead to rules more specific than the incorrect ones. The number of correct instances required to successfully 'mask' the incorrect rule is at least two: one example to acquire a rule with the same conditions and an additional example to acquire a rule with an additional feature in the conditions. More examples may be required depending on the distribution of feature combinations and the consistency of the feature selection heuristics. By searching through the rule space from the most specific rule to the more general ones, the first matched rule is typically the correct rule.

I have yet to explore intermediate alternatives such as adding a rule with two additional features or adding several different rules each with one additional feature. The performance of these alternate strategies are still open to further research. However, empirical studies have ruled out the extreme alternative, which adds rules of all levels of specificity for each presented training example. Findings indicate that this approach learns at the same rate (with respect to the number of training examples) but with significantly worse accuracy.

## 3.2 Feature selection

As noted earlier, the effectiveness of this approach depends on the order in which features are removed from the example description. Ideally, all the relevant attributes should be in the conditions of the acquired rules. Thus a good heuristic would remove the irrelevant features first so that rules with relevant features in their conditions can be successfully indexed. If relevant attributes are removed from the description before a match occurs, the quality of the category prediction will certainly suffer. In addition, if the selection of attributes during training trials is inconsistent, the learning rate of more specific rules will suffer. Rule retrieval using one feature selection heuristic will fail to retrieve rules learned under a different heuristic. Thus, the ideal search heuristic should be stable across training trials.

Despite SCA's dependency on a reasonable selection heuristic, SCA's performance degrades only in terms of learning rate as the choice of attribute removal suffers. In the case where irrelevant attributes are kept in the example description at the expense of having relevant ones removed, SCA will still be able to make good predictions once specific enough rules have been learned that include

both the irrelevant and the relevant rules. The stability of the order in which attributes are removed is also important. Erratic selection of attributes will slow the progression of learning specific rules. This is because the rule search may overlook a specific rule if the order of feature removal was different when the rule was acquired. However, after enough training examples, specific rules are still acquired.

In considering the qualities of a good feature selection heuristic, I now describe a simple strategy for learning heuristics which interacts with the rule retrieval process. Ultimately, this particular strategy will be used in evaluating SCA as it compares to human behavior. In the next chapter, I will discuss alternate strategies and their functional tradeoffs.

Upon finding a match, the system can seize the opportunity to evaluate the relevance of the matching features. Since a good set of relevant features should minimize the number of conflicting predictions, a simple heuristic can evaluate the quality of the match based on this the number. For example, let us assume the system matches an object description with the following features:

```
shape: spherical  color: blue  texture: smooth
   predicts 'ball' 'globe'
```

Thus, the number of conflicting predictions with the matched values is 2. This 'match evaluation' serves as a means of quantifying the effectiveness of the match. A lower number indicates a better quality match. However, whenever a match occurs, it is not clear which of the features contributed towards the evaluation. For example, of the features that matched, perhaps the shape is the only diagnostic feature. Perhaps shape is always a diagnostic feature. Or, it may only be diagnostic when the shape is spherical, or when the shape is spherical and the color is blue. In general, it is difficult to know which attributes under what context contributed to the result. Since it is not computationally feasible to record statistical counts of match evaluations under all contexts (the number of contexts grows exponentially with the number of features), I resort to a reasonable heuristic.

I propose two possibilities that approximate an attribute's relevance, but at minimal computational cost. The first averages the match evaluations by attribute, independent of context. The second keeps averages by the attribute's values. The first approach captures generalities that exist for an attribute, but fails to capture specific cases where the attribute may only be relevant when it has a specific value. For example, the 'by attribute' approach can learn that color as an attribute is often irrelevant, but fail to learn that certain particular colors are often relevant. In contrast, the 'by value' approach can learn that particular colors are relevant but fail to learn that color in general is irrelevant. Martin and Billman (1991) argue that people generalize across attribute values and that this strategy has functional advantages. I follow this functional route of averaging matching evaluations by attribute. Thus, when SCA must remove an attribute (preferably the least relevant) from an example during its search for a matching rule, the feature with the highest average of conflicting predictions is removed first. *Averaging* the match evaluation across many predictions increases the model's immunity to spurious inconsistencies while also stabilizing the order of feature removal.

Figure 3.3 provides the specification of the 'by attribute' selection heuristic in the context of SCA. This algorithm maintains the average number of conflicting predictions for each attribute, and is the strategy that I will use for modeling human behavior in Chapter 5. After recalling a set of predictions, *Update* is called with the current example description and the number of predictions. For each feature, the number of predictions is averaged into previous updates indexed by the feature's attribute name. For feature selection, the feature with the largest average, again indexed through attribute name, is returned. The process of updating evaluation averages is consistent with the model's ability to learn incrementally since the computational cost of each update is constant with respect to the number of previous updates.

## 3.3 An extended example

Let us now go through an extended training trial where the feature selection order changes. In doing so, I will consider simple object descriptions of three attributes. To start, we will assume that the system guesses that the initial ordering of feature removal (from least relevant to most relevant) is

```
define Train(D,c)
   set D' = D
   do
      set R = recall(D)
      if c in R or size(D) = 0
         set Match = true
         set p = c
         store(D' --> c)
         Update(D,size(R))
      else
         set D' = D
         set D = D - Select_Feature(D)
      until Match or size(D) = 0

      return(p)
end Train

define Update(D,s)
   for each d in D
      Update_Avg(Attribute(d),s)
end Update

define Select_Feature(D)
   set d such that Get_Avg(Attribute(d)) =
                   Max(Get_Avg(Attribute(D)))
   return(d)
end Select_Feature
```

Figure 3.3: Feature selection heuristic in the context of SCA

**texture, color, shape**. With this order, the following training examples are presented to SCA (boldfaced values represent values matched in a previously learned rule):

1. {oblong, red, smooth; ball}
2. {spherical, blue, smooth; globe}
3. {spherical, blue, smooth; ball}
4. {spherical, blue, smooth; globe}
5. {spherical, red, fuzzy; ball}
6. {spherical, blue, smooth; globe}

Presented one at a time, these training examples incrementally create the following rules (the number of the example corresponds to the number of the rule it created):

1. [oblong] ⇒ball
2. [spherical] ⇒globe
3. [spherical] ⇒ball
4. [spherical, blue] ⇒globe
5. [spherical, red] ⇒ball
6. [spherical, blue, smooth] ⇒globe

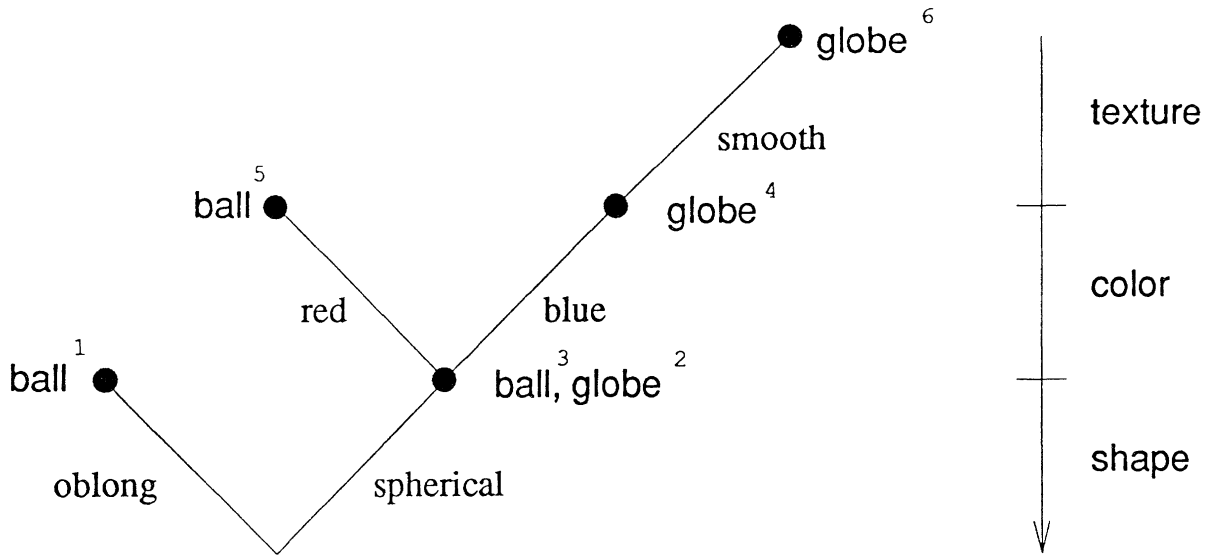


Figure 3.4: Graphical depiction of rules

Figure 3.4 is a graphical depiction of these rules as viewed from the current feature removal order (`texture, color, shape`). Each solid dot represents a prediction rule where the prediction is the category next to the dot and the the rule's conditions are all the attribute values connected below the dot. The superscripts on the predictions correspond to the rule and example numbers. The process searches from top to bottom in finding the most specific rule. For example, with the performance instance of {`spherical, red, smooth`}, rule 5 would first match, after the texture attribute (smooth) is removed from SCA's internal representation of the object.

Figure 3.4 only represents one feature ordering. If after the sixth training example, the feature ordering changes to `texture, shape, color`, then Figure 3.5 depicts the same set of rules. Rules 1, 2, and 3 are not accessible under the new ordering, since SCA's internal representation would never fully match these rules with this order of removal. But rules 4, 5, and 6 are still accessible since their conditions contain both attributes whose order has changed. Thus, reordering attribute removal does not necessarily invalidate all previously learned rules.

With the new ordering, our example continues with two additional training instances:

7. {`oblong, red, fuzzy; ball`}

8. {`oblong, red, smooth; ball`}

leading to the acquisition of these two rules:

7. `[red] ⇒ball`

8. `[red, oblong] ⇒ball`

Figure 3.6 is the graphical depiction of rules learned under the second feature ordering.

## 3.4 SCA in the context of previous work

Though SCA shares a few properties with other models, it is more of a direct consequence of applying the constraints of the Soar architecture. It is primarily these constraints that distinguish SCA from all other models. In this section, I review these constraints and their consequences on SCA's design. This serves the purpose of highlighting SCA's distinguishing characteristics which are contrasted with the design of other models of concept acquisition.

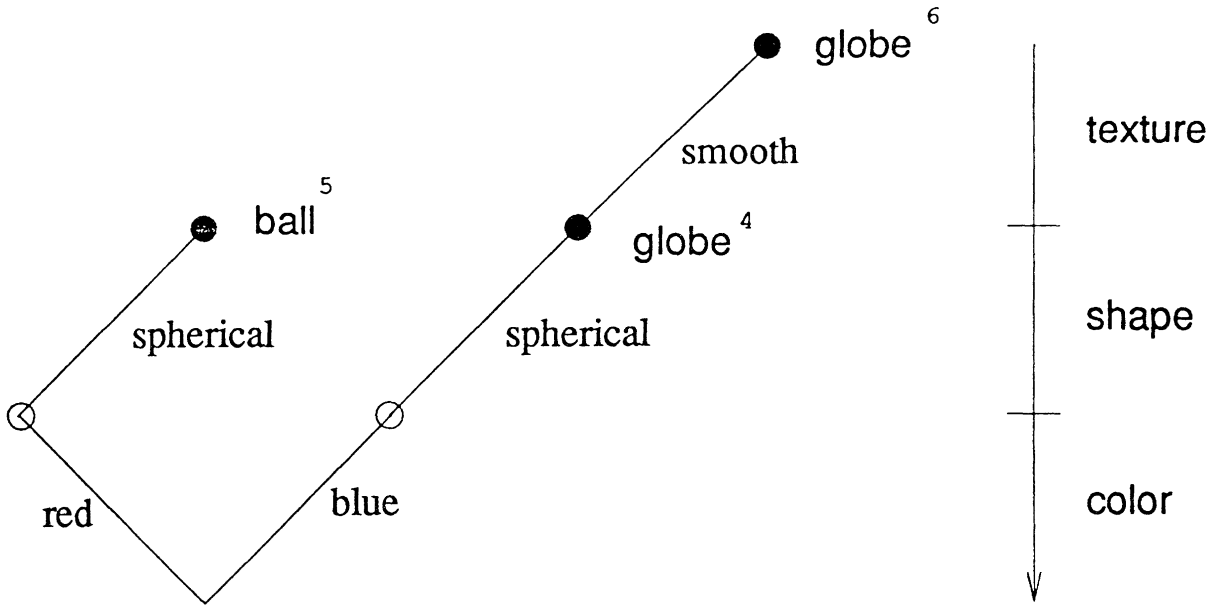


Figure 3.5: Graphical depiction of rules with alternate feature ordering

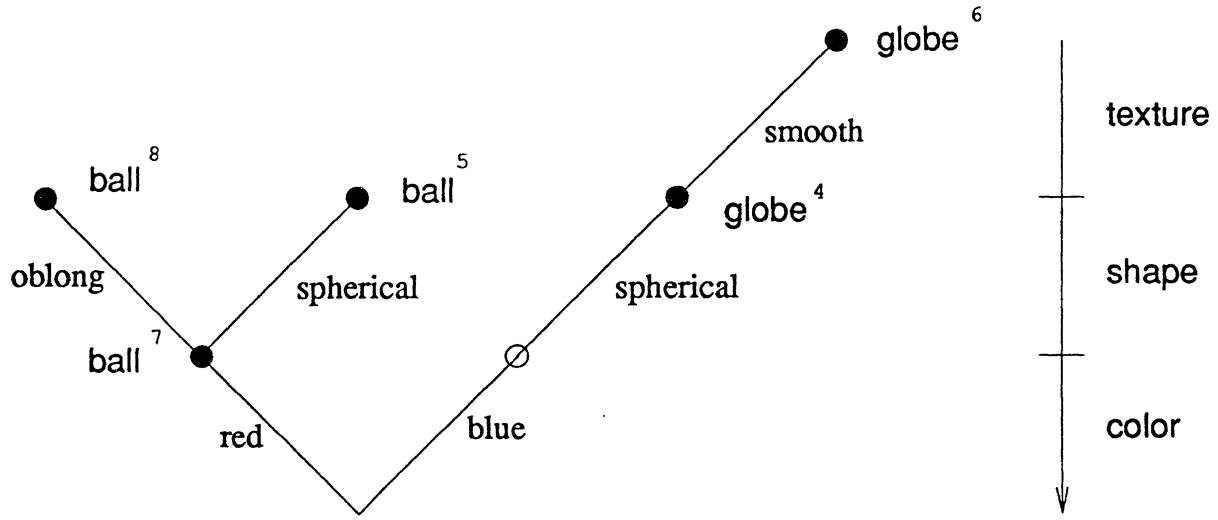


Figure 3.6: Final graphical depiction of rules

The Soar architecture, a candidate unified theory of cognition, is a system of mechanisms that applies knowledge, represented as productions,[2] in creating intelligent behavior. The choice of mechanisms has led to some universal constraints within which a large diverse set of human behavior has been modeled [Lewis *et al.*, 1990; Newell, 1990]. Likewise. these constraints apply to SCA and thus intrinsically shape the structure of the model. The major constraints of the architecture (and therefore the distinguishing characteristics of SCA) can be summarized as follows.

1. Knowledge is encoded as rules.
2. A task is performed by applying a linear sequence of discrete, deliberate operations on a temporary declarative representation.
3. A rule in long-term memory is only accessed by its successful match of working memory

[2] In this dissertation, the terms *production* and *rule* are used interchangeably. In other contexts, the term *production* implies additional processing constraints which I make explicit here.

contents (*i.e.* rules cannot be directly examined).

4. All learning is the result of performance in application of prior knowledge (explanation-base learning).

5. Once a new rule is acquired, it is never lost.

6. Rule matching occurs over a fully symbolic representation.

The second and third constraints distinguish Soar (and SCA) from many other "rule-based" systems. For Soar, problem solving does not have direct access to its rules, as they are not declarative structures whose contents can be examined and modified. A production is only accessed when its conditions match the contents of working memory. In effect, in order to seek out knowledge in long-term memory, the system must deliberately alter its working memory contents "in search" for the knowledge represented *procedurally* as productions. This contrasts with other rule-based systems whose learning mechanisms can directly examine and modify their rules [Dejong and Mooney, 1986; Anderson, 1983; Holland and Reitman, 1978]. This constraint may seem unnecessarily restrictive; however, by limiting the means in which productions are accessed, the architecture can incorporate efficient match strategies whose matching cost do not significantly degrade as more productions are acquired [Forgy, 1982; Doorenbos *et al.*, 1992].

Not only do the above constraints serve in defining SCA's distinguishing characteristics, they also motivate several design decisions. For example, the Soar architecture prohibits the possibility of directly removing incorrect rules as a means of error recovery (Constraint 5). Thus SCA takes an alternate approach by first learning general rules and thereafter incrementally acquiring more specific ones. Since SCA probes for more specific rules first, recovery occurs by eventually learning more accurate specific rules. The older inaccurate ones will be successfully masked from performance.

This same strategy delivers frequency effects without keeping any explicit frequency counts, weights or probabilities (Constraint 6). For example, consider the situation where SCA is presented with a ball but incorrectly told it was a 'can'. SCA would acquire a general rule classifying the object as a 'can'. However, upon receiving several correctly classified examples of balls, it would eventually acquire more specific rules that override the incorrect one. In short, a rule's specificity is an implicit measure of how many examples of one category SCA has encountered.

This approach starkly distinguishes SCA from other production-based models. The ACT model of schema abstraction [Anderson *et al.*, 1979] is similar to SCA in that it encodes conceptual knowledge disjunctively distributed in the form of productions. However, it first acquires specific productions while incrementally generalizing them to create new, less specific ones. Furthermore, it maintains a weight with each production in order to keep track of the production's utility. The weight serves as a guiding factor in whether the production will apply. Like SCA, the specificity of the rule is also important in determining whether a production should apply. However, specificity is explicitly calculated and directly factored into the production's probability of being applied. In contrast, SCA indirectly factors in specificity during performance through a serially ordered search in long-term memory.

The classifier model [Holland and Reitman, 1978] is another production-based model. Their system also uses several parameters in keeping track of a production's utility. The learning of new productions is not directed in a certain way as in SCA (general to specific) or ACT (specific to general). Instead, the set of productions are incrementally modified by genetic operators with the directed goal of optimizing prediction accuracy. In contrast to the third architectural constraint, the genetic operators have direct access to the rule base allowing an unconstrained set of possibilities for rule modification. For performance, a series of productions may apply as determined by the their strengths.

In neither the ACT model nor the classifier model does prediction performance depend on a series of *deliberate choices*. In contrast, SCA requires a series of control decisions that guide which features are relevant to a prediction (Constraint 2). This arises from how problem solving is approached in Soar, where task performance is guided by deliberate decisions, *i.e.* decisions that bring to bear all pertinent knowledge within the system. Even though the feature selection learning mechanism

is not compatible with the sixth constraint, it suggests one type of knowledge useful in guiding feature selection. That deliberation plays a role in induction is consistent with a later observation by Anderson where he suggests, "there is evidence that the generalizations people form from experience are subject to strategic control" [Anderson, 1987, p. 205]. Thus he questions whether induction is truly an automatic process, as implied by the ACT model.

In related work, Billman and Heit's CARI [Billman and Heit, 1988] learns prediction rules whose construction are guided by a technique they call *focused sampling*. Focused sampling is similar to the feature selection heuristic described here. In their application, however, focused sampling was limited to the construction of rules with only one feature in the rule's condition. ALCOVE [Kruschke, 1991] is an example of a connectionist model of human categorization that has applied this heuristic. Here the heuristic is implemented by weighting the focused features proportionally higher.

Perhaps EPAM [Simon and Feigenbaum, 1964; Feigenbaum and Simon, 1984] is most similar to SCA in that it conforms to most of Soar's architectural constraints. EPAM is a fully symbolic discrimination net which discriminates between concepts by focusing on the minimal number of features needed to distinguish between concepts. Like SCA, EPAM incrementally learns more specific discriminations. The critical difference is that EPAM fixes which features the model should discriminate on first. For SCA, feature selection always remains a deliberate choice. Moreover, the knowledge a system brings to bear in making a deliberate choice need not be limited to heuristics directly guided by empirical success. Other types of knowledge include partial domain theories and knowledge arising from verbal instruction. In principle, these types of knowledge, also represented as productions, can apply to the decision making process. Further research should address the issue of how these other types of knowledge arise and how they influence the induction process. Also, it remains an open issue if the feature selection mechanism can be implemented within Soar's constraints. In Chapter 6, I describe a pilot implementation that meets these constraints and present some preliminary results.

Another way to compare systems is to examine the context in which the system learns from a training example. The ACT model and classifier model learn by trying to predict the category of a training example. Learning occurs by adjusting production parameters depending on whether the prediction is correct or not. Similarly, feed-forward connectionist networks (*e.g.* backprop nets [Rumelhart *et al.*, 1986b] and the configural-cue model [Gluck and Bower, 1988a]) learn by adjusting connection weights after making predictions on a training example. SCA also learns by trying to perform on a training example using prior knowledge. However, SCA's learning is *explanation-based* where prediction continues until the correct prediction is made. Learning occurs by summarizing the experience of making the correct prediction (Constraint 4).

Exemplar-based models contrast with the above approaches of learning through performance. Examples of these models are the Medin and Schaffer's Context Model [Medin and Schaffer, 1978], Hintzman's MINERVA models [Hintzman, 1986] and Aha's instance-based learning model [Aha, 1989]. These models learn by storing the training examples. For performance, classification is determined by the classifications of all stored examples, weighted by their similarity to the queried example. Implicit with these models is the assumption that humans can internalize an entire example representation independent of prior learning. In contrast, SCA acquires rules that approach the content of an exemplar only after encountering the same example (or similar ones) several times.

Other models combine several of the above approaches. ALCOVE [Kruschke, 1990] employs a feed-forward connectionist network organized to represent entire examples. One of its learning mechanisms learns by adjusting weights searching to minimize error. A second learns attentional weights for focusing on incoming features. COBWEB [Fisher, 1988] also represents entire examples. The model provides an efficient indexing strategy by organizing examples hierarchically. For both training and performance, the hierarchy, composed of abstract descriptions represented by attribute-value probabilities, is descended, placing the new description so as to maximize feature prediction. For performance, prediction is based off of the best matching concept in the hierarchy. For training, the new example is incorporated into the hierarchy by either updating a previous concept description or by creating a new description. Furthermore, operators may directly modify the hierarchy in order to maximize the ability to make correct feature predictions.

In this section, I have claimed that the Soar architecture has placed constraints on SCA's structural design. It is these constraints and their consequences that distinguish SCA's design from other models. SCA's learning of incrementally more specific rules is one such consequence. Another consequence is that instead of using explicit frequency counts associated with prediction rules, SCA relies on search procedures that manifest many of the desired effects captured by frequency counts.

# Chapter 4

# SYSTEM BEHAVIOR AND EMPIRICAL EVALUATION

## 4.1 Introduction

In this chapter, I explore the learning behavior and performance of SCA as an incremental supervised learning algorithm. In the previous chapter, the presentation is oriented as a model of human category acquisition designed within the context of Soar. Here, however, I present the SCA algorithm separate from the Soar context, as a novel algorithm with interesting learning behaviors that distinguish it from previous approaches within the machine learning paradigm. Furthermore, I demonstrate SCA's promise as a powerful learning approach in terms of accuracy, noise tolerance and learning rate.

Empirically evaluating SCA's performance also motivates its plausibility as a cognitive model. In the last chapter, SCA's design was explained in terms of how the model fufills its learning task. In other words, the functionality of the design components were *locally* motivated. Here, the model's functionality is evaluated *globally* by comparing its learning characteristics with machine learning systems primarily engineered for performance.

Traditionally, supervised learning algorithms in machine learning have taken three approaches to representing concepts. The first approach seeks a compact symbolic representation that covers the concept's positive examples while excluding negative examples. Searching for a parsimonious concept representation as an approach is defended by Blumer *et al.* (1987) and underlies the working principles of many supervised learning systems. These include decision trees [Quinlan, 1986], which search for the smallest tree that covers the training examples, and the STAR methodology [Michalski, 1983], which seeks the simplest logical formula as represented in disjunctive normal form.

The second approach saves classification examples instead of maintaining a compact explicit concept definition. In categorizing a new example, the algorithm seeks a similar, previously classified example. This approach places a burden on the system to correctly index the appropriate preclassified example in memory. Example systems include a nearest neighbor method [Aha, 1989], and COBWEB [Fisher, 1987], which indexes instances through a concept hierarchy constructed by an unsupervised clustering strategy. Also, in contrast to many examples using the first approach, both of these systems use probabilities or weights which quantify a matching metric by which search for the best exemplar is guided.

The third approach also does not keep an explicit concept definition. It differs from the second group, however, in that it does not maintain the integrity of individual training examples. The primary examples of this approach are the artificial neural network models [Rosenblatt, 1962; Rumelhart *et al.*, 1986b; Kohonen, 1982]. Concept representations are distributed across a network of excitory and inhibitory connections. Learning occurs by incrementally adjusting weights in order to minimize the prediction error on training examples. Evolutionary systems similarly alter their conceptual representation as they search for the representation that minimizes prediction error

[Booker *et al.*, 1989; Holland and Reitman, 1978].

The approach described here does not conform to any of these three approaches. SCA is a symbolic rule-based system, but it does not try to maintain a compact explicit concept definition. Neither does it keep whole training instances for future classification. And, rather than explicitly adjusting the concept representation to minimize error, it incrementally acquires new prediction rules by applying previous rules in *explaining* the training example's classification. SCA represents each concept with many rules. At first, the rules are very general, but as learning progresses, more specific rules are acquired. For category prediction, SCA searches for the most specific prediction rules first before accessing more general ones.

In the next section, I discuss some of SCA's learning behaviors that result from its design. These behaviors are general consequences independent of the feature selection strategy. In the following section, I describe alternate feature selection strategies and their consequences. Finally, with a different, more computationally intensive feature selection strategy, I make some empirical comparisons with other established concept acquisition systems in order to demonstrate SCA's effectiveness as an approach to concept acquisition.

## 4.2 Behavior

Here I discuss the behavioral consequences of SCA's design. Because the specificity of SCA's rules has important consequences on response time and accuracy, I first review how SCA learns specific rules and their resulting impact.

As SCA learns, it gradually acquires more specific rules. Figure 4.1 shows how more specific rules are matched through the course of learning. These results were obtained by running SCA on the congressional voting data set.[1] This dataset contains the voting records of congress representatives on a selected set of issues. Each issue serves as a feature attribute with the possible values of yes, no or unknown (presumably including absences and abstentions). The task is to learn to predict the party affiliation of a representative when given only the voting record. In estimating the dataset's difficulty, Paxton (1990) reports of two supervised learning systems, MARTIAN [Paxton, 1990] and STAGGER [Schlimmer, 1987], which, at asymptotic performance, were both able to successfully predict party affiliation 90% of the time.

In Figure 4.1, the x-axis specifies the number of encountered training instances. After intervals of twenty training instances, the system made predictions on a set of test instances. The y-axis specifies the average specificity, in terms of features matched, of the indexed prediction rules for the test set. Clearly, as more instances are encountered, the specificity of the match increases. This is to be expected, since SCA always learns a new rule whose conditions include one more feature than the rule that matched the training instance. Figure 4.2 shows the relationship between rule specificity and prediction correctness. Using data from the same run of Figure 4.1, the prediction accuracy was calculated according to rule specificity. The x-axis specifies the specificity of the indexed prediction rules, which is the number of features in the matching rule. The y-axis specifies the accuracy, *i.e.* the percentage that the rule of this specificity predicted correctly. Roughly, the more specific the rule is, the more likely the prediction will be correct. This observation helps us understand SCA's external properties which I now explain.

SCA has three behavioral consequences that are independent of the attribute selection heuristic choice. They are 1) learning often improves over multiple exposures of the same training data, 2) computational expense lessens through the course of learning and 3) the algorithm has a built-in tolerance for noise.

Often one pass through a training set is sufficient for SCA to acquire rules specific enough to make reasonably accurate predictions. Typically, however, running it over the same dataset continues to improve performance. Figure 4.3, shows the effect on accuracy by exposing a small training dataset multiple times. This dataset was a set of stimuli used in an experiment discussed in the next chapter (see Table 5.3 for the list of instances), but the phenomena exemplified here is typical of all

[1] All of the datasets used in functionally evaluating SCA can be obtained from the University of California-Irvine dataset depository.

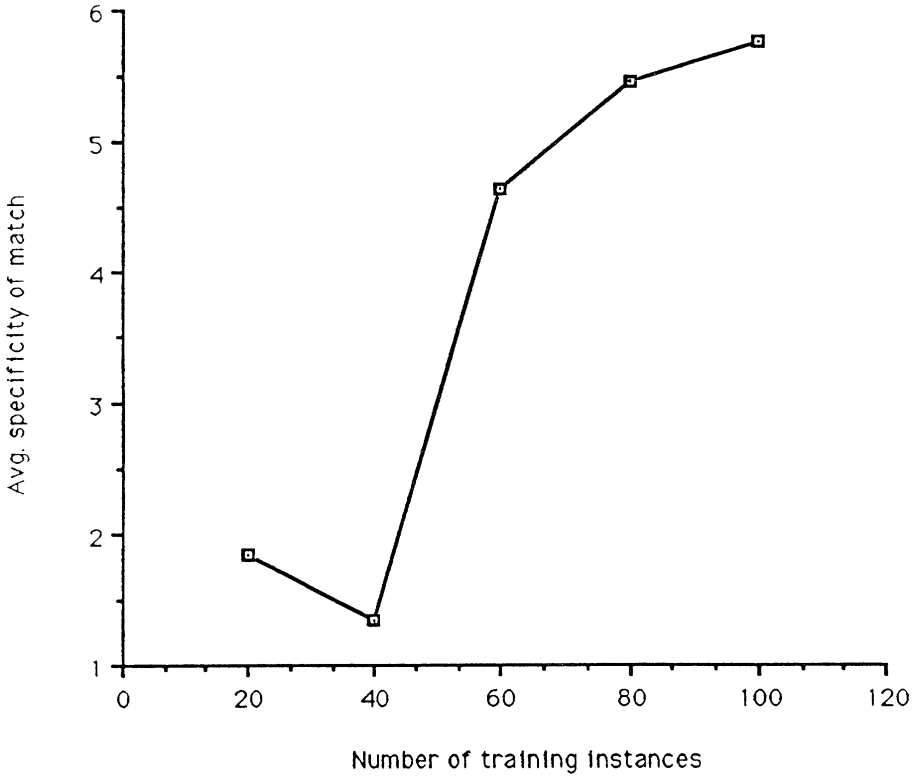


Figure 4.1: Relationship of Avg. rule specificity to number of training instances (cong. voting dataset).

datasets, provided they are not too easy. The x-axis specifies the number of times the training set was presented to SCA. The right side of the y-axis (graphed as the gray line) specifies the prediction accuracy on a test set. The graph shows how accuracy improved over multiple exposures to the same dataset. The improvement is due to the system's having learned some more specific rules containing relevant features not previously included from the first training cycle.

The same figure also shows SCA's computational runtime in relation to the number of exposures. The left side of the y-axis (graphed as the solid line) specifies the average runtime per testing instance in terms of a machine-internal time unit. Clearly runtime decreases with learning. Recall that SCA searches for specific rules first before it tries to match a more general rule. Runtime is thus inversely proportional to the specificity of the matched rule. If specific rules can be found in memory, response time is faster. After many training trials, more specific rules have been acquired, causing runtime to be faster.

Instrinsic to its design, SCA tolerates noise. In understanding its noise resilience, SCA can be thought of as a competition model where conflicting rules compete in making the category prediction. Because SCA searches for the most specific rule first, it is the most specific rule which wins the competition. Presumably, during training, the system encounters more correct instances than incorrect ones. The correct rules thus contribute to the learning of rules that are more specific than prediction rules formed from incorrect instances. Typically, correct predictions result despite there being many incorrect rules (rendered benign by their lack of specificity) in memory. Later in this chapter, empirical comparisons are made with other systems demonstrating SCA's noise tolerance.

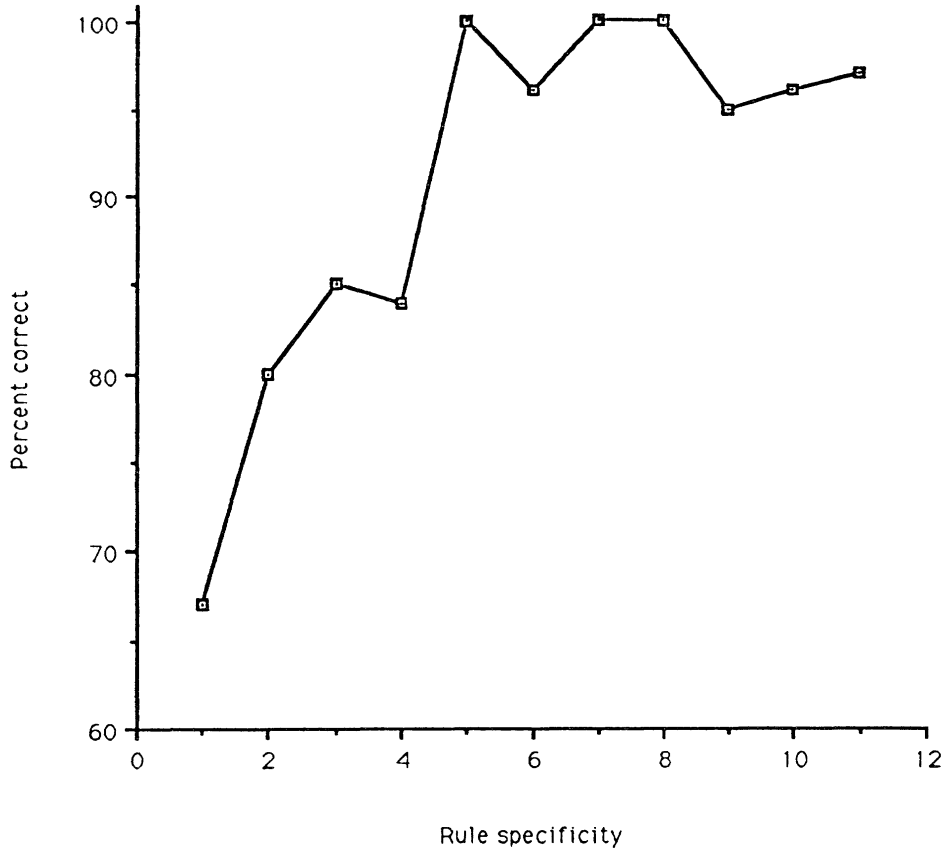


Figure 4.2: Relationship of accuracy to rule specificity (cong. voting dataset).

## 4.3 Search control

In the last chapter, I described a strategy for learning the order in which features should be abstracted out of the object description first. The selection decision is described as a deliberate process upon which many knowledge sources can be brought to bear. The selection strategy of the previous chapter is but one such knowledge source, based upon the number of conflicting predictions. It is a *data-driven* knowledge source since it is ultimately derived from knowledge acquired from the training data.

The power of the method described in the last chapter is limited since it must rely on a sporadic, low resolution evaluation stemming from prediction performance. However, this chapter emphasizes the performance of SCA when used with a powerful feature selection strategy. The goal is to motivate SCA's functionality when given one or several selection strategies. To demonstrate this, I compare SCA's performance with other machine learning systems. These systems are primarily designed for performance, outside the context of any architectural or cognitive constraints. Without such constraints, they are free to invest more computational resources to the storing and processing of category examples. Likewise, in comparing SCA to these systems, I present an alternative selection strategy that performs better, but at the cost of requiring more computational resources.

What constitutes an effective method of feature selection? As noted earlier, the ideal strategy removes the most irrelevant features from the object description first, thus leaving the more diagnostic features for prediction. Also, it is important that the method is consistent with the selection. By consistent, I mean that the order of feature removal makes only relatively small changes in feature removal order from trial to trial. If the order changes often, rules generated by a previous ordering become inaccessible to retrievals controlled by a radically different ordering. Small, local changes minimize the problem of accessibility if changes are made between features that are already included

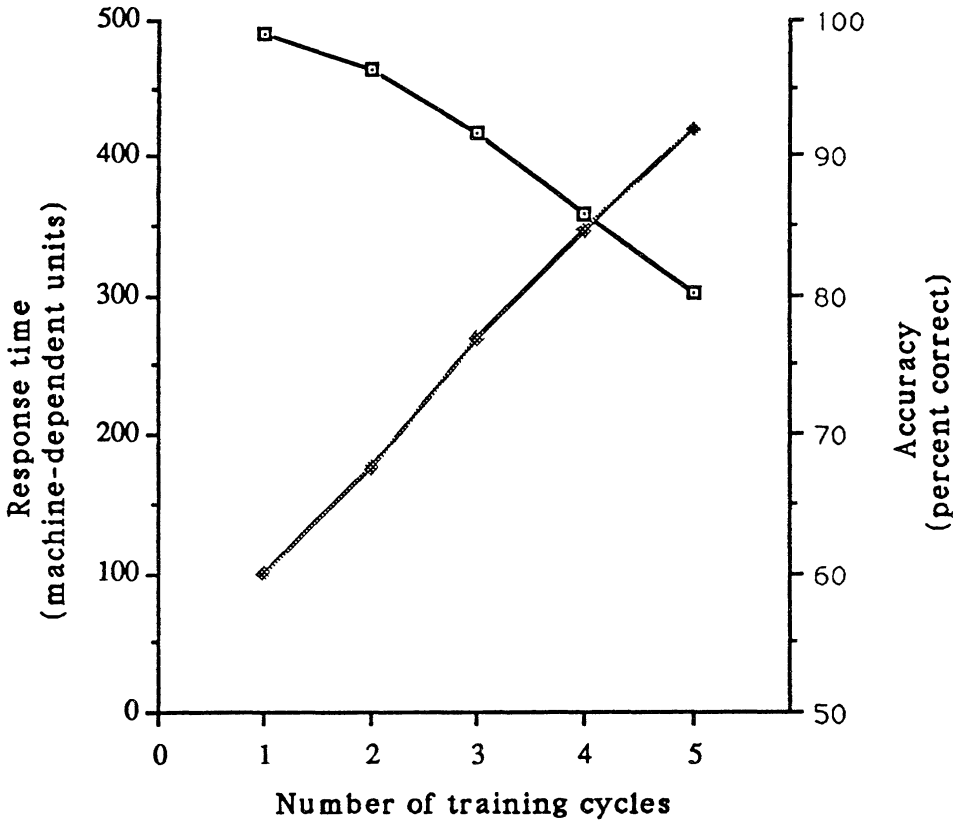


Figure 4.3: Performance and runtime improvement over multiple trials.

in the rules' conditions or if they are between features that are all excluded from the rules' conditions. Furthermore, a stable feature ordering quickly acquires specific rules since more specific rules are acquired in the context of accessing previously learned rules.

To illustrate the consequences of feature selection consistency, let us consider an example. To start, let us assume that the respective order of feature removal is rigidity, texture, color, then shape. The following rules could plausibly be acquired after encountering several examples:

```
[spherical] --> predict category:ball
[spherical] --> predict category:globe
[spherical, red] --> predict category:ball
[spherical, blue] --> predict category:globe
[spherical, red, smooth] --> predict category:ball
```

Given the object [spherical, red, smooth, soft], the removal order would successful match the last rule in the list and thus retrieve the prediction for ball. Furthermore, if the two neighboring features shape and color were changed in the ordering (resulting in the order rigidity, texture, shape, color), the same rule would match immediately after rigidity is removed from the object description. In contrast, if shape was swapped with rigidity—a radical change with a result ordering of shape, texture, color, rigidity; none of the rules could match the object description. This is because as soon as shape is removed from the object description, none of the rules' conditions could match.

With the consequences of feature removal orderings in mind, let us now look at the performance of different order strategies. Figure 4.4 reports some empirical results comparing different feature selection strategies. From a dataset of 335 examples, 235 examples were randomly chosen and ordered for training and the 100 remaining were randomly ordered for testing. This procedure was repeated 50 times, each time with differently selected and ordered training and testing sets. The graph presents accuracy means inside a 95% confidence interval.

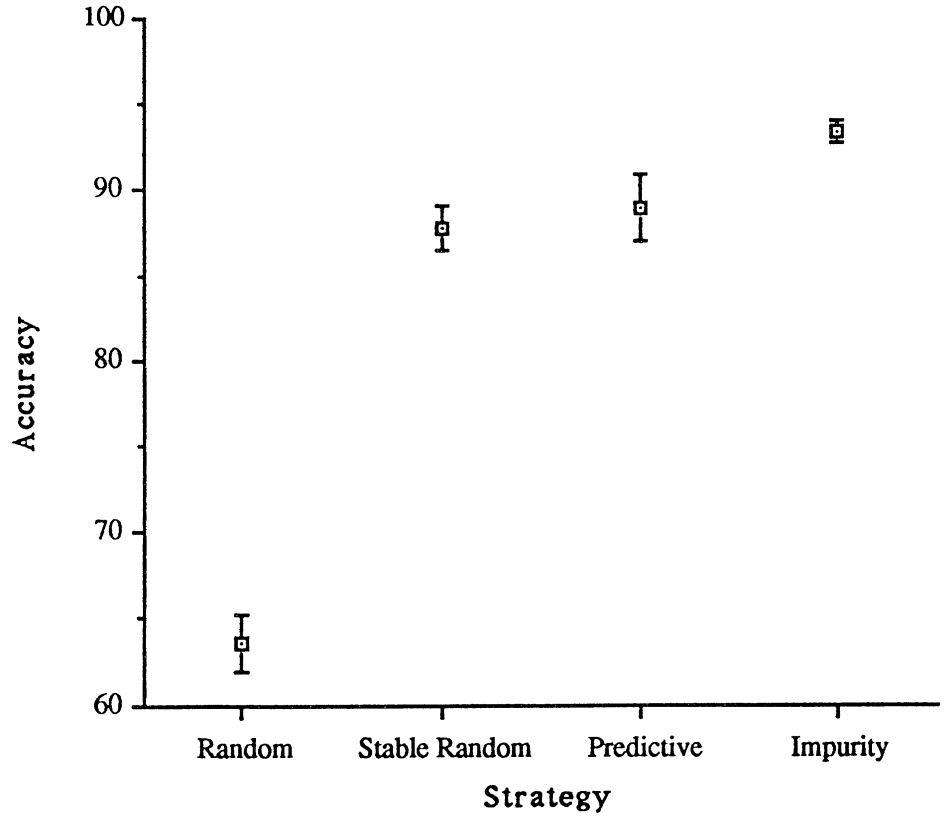


Figure 4.4: Performance as a function of feature selection strategy (cong. voting dataset).

For now, let us consider the results of the first two strategies in the figure. The simplest strategy is choosing features at random. For each example, a feature ordering is randomly and dynamically assembled. The feature removal order for one example could be radically different for the next. The second strategy also uses a random ordering. However, the same random ordering is kept through the entire data set. This is the *stable random* strategy since the same ordering is consistently used from one example to the next. The first strategy does poorly since it slowly acquires more specific rules. Furthermore, it does not necessarily access the most specific that it has required. In contrast, the stable random strategy is guaranteed to access the most specific applicable rule since the ordering is the same as when the rules were learned.

The third strategy is the prediction-based 'by attribute' evaluation method proposed in the last chapter. This method used the number of conflicting predictions as feedback in evaluating the relevance of each matched attribute. In this case, it only performs insignificantly better than the stable random strategy. Eventually, this method finds a good ordering of features but with a cost. In order to find a good ordering, the strategy must experiment with different features with which it receives a crude evaluation based upon the number of conflicting predictions. A good evaluation may be the result of having one good feature in the conditions, of having several good features in the conditions, or of simply not having yet encountered an example of a contrasting category stored with these features. After many trials and much experimentation, evaluation averages become significantly useful in ordering the features. The inconsistency of exploration initially hurts performance in the same way that the random strategy does. In the long term, however, the acquired ordering would aid in the learning of new categories.

The fourth strategy independently calculates an impurity measure for each attribute. These measures capture the amount of information gain for category prediction when given the value of an attribute. In particular, the entropy gain used with ID3 [Quinlan, 1986] was chosen. Pilot

experimentation did not reveal significantly different results using other impurity measures.

Equation 4.1 is the formal specification of the measure used with the fourth strategy. $N$ is the number of categories. With $V$ values for attribute $A$, $k_{ji}$ is the number of examples in the $j$th category with the $i$th value for attribute $A$. For this formula, the measure is inversely correlated with the attribute's relevance. Thus, SCA abstracts out the attributes with the highest measure first.

$$E(A) = -\sum_{i=1}^{V} \frac{S_i}{S} \sum_{j=1}^{N} \frac{k_{ji}}{S_i} \log \frac{k_{ji}}{S_i} \tag{4.1}$$

Because SCA is an incremental learner, counts for each variable must be updated with each new training example. Furthermore, the measure must be recalculated for an attribute every time one of its variables is updated. The maintaining of the impurity measure thus comes at an additional computational expense, where the additional expense is proportional to the number of all values associated with every attribute in the object description. Figure 4.4 shows that there is a benefit with the expense. The use of an impurity measure produces significantly better performance than the other strategies.

## 4.4 Empirical comparisons

SCA's performance was compared with two machine learning systems, ID3[2] and COBWEB.[3] The implementations of both of these systems contain no explicit mechanisms for handling noise. While implementations for handling noise exist [Quinlan, 1986; Fisher, 1989], these are not used since the goal is to understand learning properties inherent to the model's design. It is also important to realize that both COBWEB and SCA are incremental learners, whereas ID3 is not. For all of these studies, ID3 had access to all training examples throughout the construction of its conceptual representation.

The results were obtained using the congressional voting dataset[4] with the same methodology described in the last section (randomly choosing 235 training instances and 100 testing instances). In order to illustrate the systems' tolerance to noise, the procedure was also repeated with the introduction of noise in the training data. In particular, 10% of the training examples were randomly selected and then corrupted with the wrong categorization.

Figure 4.5 shows the results of the comparison. For the uncorrupted training set, little if any significant difference exists between the three systems. For the noisy training set, however, SCA performs better than the other systems. Since the error bars of SCA and ID3 overlap, a two sample, lower-tail $z$ test was performed revealing a significant difference with $\alpha = 0.01$ ($z = -2.509$).

Both ID3 and COBWEB suffer a 10% decrement in performance. The reason is that both systems tend to fit the training data to the extent that wrongly classified examples are incorporated into their concept descriptions. In other words, they *overfit* the data. ID3 continues to learn new discrimination conditions as long as it fails to accurately classify all of its training examples. In the case of noisy examples, it must use irrelevant conditions in order to fit the noisy examples. COBWEB hierarchically organizes its concept structure with training examples represented at the leaves of the hierarchy. If 10% of the training examples are incorrectly classified, then 10% of the leaf nodes will have object descriptions with the wrong classification. As a result, roughly an additional 10% of the test examples will be incorrectly classified.

Unlike ID3, SCA does not update its concept representation in order to classify correctly all training instances. Rather, with each new training example for which the most specific applicable rule does not correctly classify it, SCA retrieves the most specific rule that does correctly classify

[2] I used a version of ID3 that built a binary tree, implemented by John Paxton [Paxton, 1990]. Fayyad (1991) reports that a binary tree is superior to the standard implementation.

[3] The tested implementation was COBWEB/3, obtained from NASA Ames Research Center.

[4] In addition to the congressional dataset, pilot experimentation revealed all three systems easily learning the mushroom dataset and COBWEB performing marginally better than ID3 and SCA on the soybean dataset [Miller and Laird, 1992].

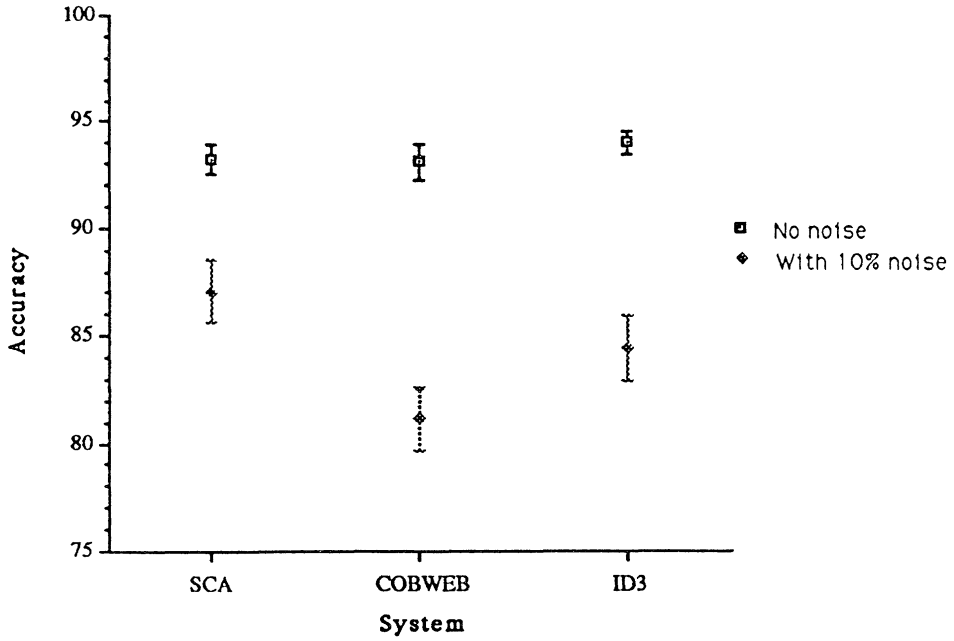


Figure 4.5: Prediction accuracy.

it and builds a new rule that is slightly more specific (*i.e.* one feature more specific in the current implementation).

SCA's tolerance for noise comes at a price. Its learning rate is significantly slower than ID3 and COBWEB. The learning strategies of these systems aggressively fit the training data and thus can learn much faster. Figure 4.6 reports the results of an example run showing the incremental performance at several stages of learning. Both ID3 and COBWEB achieve near asymptotic performance with only a small number of training examples. In constrast, SCA requires many training examples before its rules are specific enough to achieve asymptotic performance. Furthermore, initial performance is particularly poor, as it is still learning a good feature selection order.

The trade-off between noise tolerance and learning rate has significance outside of this work. Other empirical studies [Shavlik *et al.*, 1991; Fisher and KcKusick, 1989; Mooney *et al.*, 1989] have shown that the backpropagation algorithm as applied to an artificial neural net tolerates noise well in comparison to ID3. However, backpropagation requires considerable more time and training examples before asymptotic performance is achieved. In some cases, the total required training time can be several orders of magnitude greater than that of ID3. Perhaps SCA is a compromise between these two learning paradigms. Based on Figure 4.6, it appears that SCA requires at most one order of magnitude more training examples than ID3 while also showing a better tolerance for noise, but perhaps not to the extent that backpropagation can.

## 4.5 Summary

In this chapter, I highlighted SCA's key behavioral characteristics. An emphasis was given to functional properties in order motivate SCA as a model of human learning. Chief among these properties are 1) learning often improves over multiple exposures of the same training data, 2) computational expense lessens through the course of learning and 3) the algorithm has a built-in tolerance for noise. All of these are consequences of SCA's design independent of any particular feature selection strategy.

SCA's functionality is further supported by comparing it to other machine learning systems which have been specifically engineered for performance. The goal was not to show that it is superior to

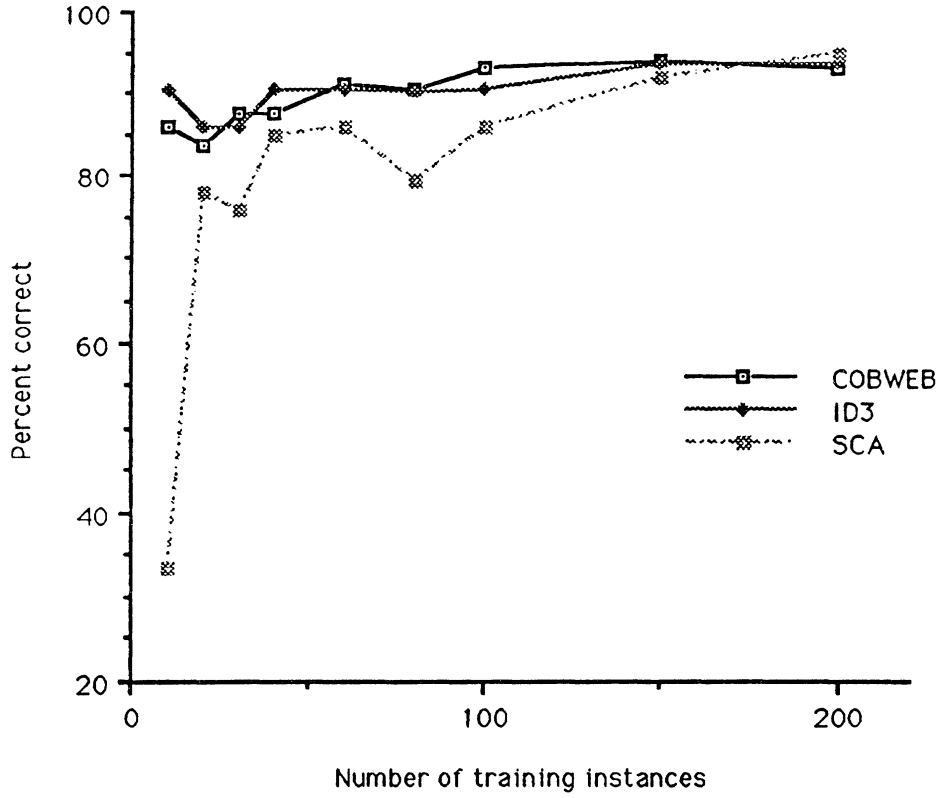


Figure 4.6: Learning rate (cong. voting dataset).

other systems. Rather, I simply emphasized that the design, when coupled with a reasonably powerful feature selection technique, is competitive. Nevertheless, in the case of noise tolerance, this goal was surpassed. SCA, without the benefit of probabilistic matching or statistical pruning techniques, was found to perform significantly better than the other systems.

# Chapter 5

# REPLICATING HUMAN BEHAVIOR

In this chapter, I describe some phenomena manifested by humans in learning categories and evaluate how well the model replicates this behavior. Much of the model's behavior can be attributed to SCA, the rule acquisition and retrieval component of the model. Indeed, previous work [Miller and Laird, 1991] reports how SCA with hand-coded feature selection heuristics models some typicality effects and exhibits a reasonable distribution of extension errors. In this thesis, the results are generated from SCA working with the 'by attribute' selection heuristic described in Chapter 3 and cover a much broader set of phenomena. However, much of the model's behavior is still explained through SCA's learning properties.

The aim is a broad covering of major robust phenomena found within the psychological literature. Coverage is limited to qualitative fits. Quantitative fits make stronger assumptions about the input to the model (*e.g.* number of features, amount of noise, structure of features) whereas qualitative predictions are preserved provided that the qualitative relationships in the model's input are the same received by the actual human process. Previous approaches to quantitative fitting may have addressed these problems by adjusting parameters in order to obtain the best fit. Some examples are connectionist models, which have parameters specifying learning rate [Kruschke, 1990] and exemplar-based models, which often have parameters controlling similarity calculations [Medin and Schaffer, 1978]. However, the use of parameters can be a difficult enterprise when it comes to showing that fits to human data are an intrinsic property of the model and not a product of having the right parameters and settings. By presenting SCA as a parameter-free model, qualitative behavioral distinctions clearly come from the model. Future work could address quantitative fits through expansion of the model's scope and use of well-motivated parameters.

Using qualitative distinctions, SCA addresses a range of phenomena along the dimensions of response time, accuracy, and learning rates. Because SCA requires varying amounts of time in searching for category predictions, a natural target is phenomena manifested by response time, *i.e.* the time required for the model to make a prediction given an instance. This explanation of response time differences distinguishes SCA, as a process model, from other approaches. For other phenomena, I choose those that have been useful in distinguishing among previous theories. For example, ALCOVE [Kruschke, 1991], a hybrid of connectionism and exemplar-based learning, targets behavior for gating categories versus condensation categories. This phenomenon distinguishes it from previous exemplar-based approaches and parallel connectionist approaches. Likewise, exemplar-based models target behavior for linearly versus non-linearly separable categories in distinguishing them from prototype-based approaches. I also present some findings on basic-level superiority, a phenomenon that SCA does not adequately address. Certainly, many more diagnostic phenomena exist. However, the research achieves the primary goal of illustrating how my approach exhibits some reaction time effects which others do not while also presenting results modeling other diagnostic effects.

## 5.1 Response time

The amount of time a person takes to predict an instance's category can vary as a function of several factors (*e.g.* typicality). Here the claim is that many category learning strategies do not require varying amounts of time and thus the human data on response time is quite revealing of the process underlying performance. Furthermore, I assert that SCA does conform with some of the interesting data while also making a novel prediction.

SCA, the rule acquisition portion of the model, does not use weights, similarity measures, frequency counts or probabilities. Instead it relies on deliberate search that orders access to prediction rules of varying degrees of accuracy. As a consequence, the time required for performance is not constant. Furthermore, because SCA learns by performing on training instances, the time required to process a training instance covaries with the time it would take to process it as a test instance. Despite the varying response time, SCA lies within the definition of an incremental learner. This definition requires that the learning system take at most a maximum amount of time which does not increase with the number of training examples. In conforming to the definition, we see that response time for SCA is bounded by a constant proportional to the number of features in an instance description. However, this is only an upper bound and is independent of the number of instances encountered.

SCA is unique in making many response time predictions because the algorithm is fully defined thereby specifying which processes occur in parallel (*e.g.* production matching) and which are performed serially (*e.g.* deliberate search). Furthermore, these procedural design decisions have been motivated through functional and architectural constraints and not as an *ad hoc* need to fit the human data.

In contrast, the many learning theories that use explicit numerical representations do not commit to an *algorithmic procedure* describing how the quantitative data is processed. For example, the context model [Medin and Schaffer, 1978] mathematically defines the probability a stored instance will be retrieved in making a category prediction. While the probability calculation is dependent on the similarity of all instances in both contrasting categories, the model does not commit to how this information is gathered and processed. Similarly, ALCOVE [Kruschke, 1990] mathematically defines probabilistic, causal relationships between connections in its artificial neural net, without committing to a process that implements them.

The process specification is critical in addressing response time issues such as whether one type of instance is processed faster than another type or whether process time decreases as a category is mastered. For analysis I can nevertheless speculate on a reasonable implementation and analyze its implications. Moreover, after carrying out these speculations, I maintain the claim that there is no obvious reason why models that maintain explicit numerical information should deliver varying response times.

My contention is that models using quantitative representations suggest implementations which require the same amount of performance time for each instance. Consider Figure 5.1. In many ways, this is a generic characterization of performance for many of the theories cited in Chapter 3. In the preliminary stage, individual pieces of evidence are computed in parallel. Ultimately, however, the pieces of evidence must be brought together to a decision process where they are summed or filtered in order to produce one unique prediction. For exemplar-based models, similarity measures of stored examples constitute individual pieces of evidence. These measures are summed in order to produce a prediction. For feed-forward connectionist nets, the generic model is re-applied for each layer in the net, with the number of layers fixed in advance. A simple production-based model has productions matching in parallel and the central mechanism filters out the production with the highest strength or best match. For all cases, it is most plausible that the individual pieces of evidence are computed in parallel. If this were not the case, performance would seriously degrade as more knowledge is acquired (*e.g.* as an exemplar-based model stores more examples).

How long should it take the decision process to sum up or filter out the evidence? The most naive implementation would serially run through each computed piece of evidence either summing them or searching for that with the highest weight. This strategy would require a time in proportion to the size of the evidence, where the size of the evidence is the number of contributing factors

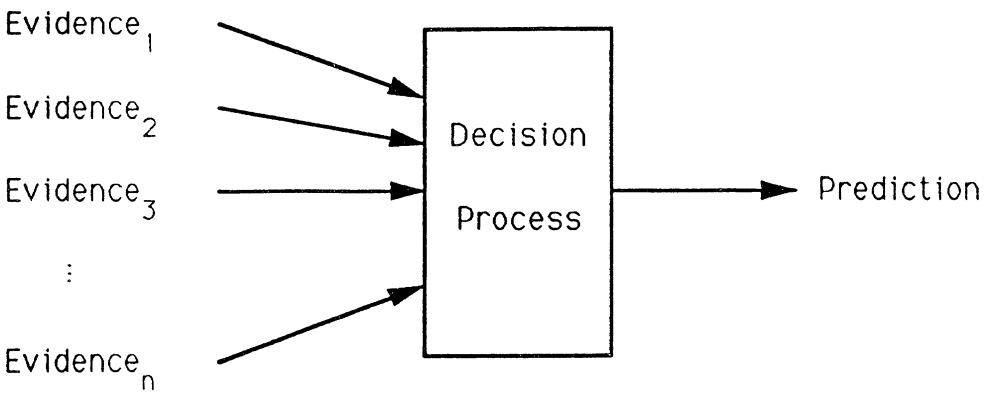


Figure 5.1: Generic performance model

(*e.g.* number of total instances stored, number of matching productions, or the number of input connections). A more sophisticated strategy could perform local decisions and using these results in producing more global decisions. This strategy would require a time in proportion to the log of the size of the evidence. Finally, there are conceivable mechanisms that could perform the decision in a constant amount of time. For all of these cases, however, the amount of required time is constant ***with respect to the type of instance being classified.*** Moreover, it may be the case that, for even the naive strategy, the process is so simple and therefore fast enough that differences in externally measured response times go undetected.

The above argument is by no means a formal proof that these models cannot have implementations that predict varied response times as a function of instance type. This generic model is perhaps an oversimplification and may miss some additional sophisticated features of some models. For example, the classifier model may require several serial production matches which vary on factors other than the production set size. Furthermore, I have only examined the most obvious implementations for the generic model.

More sophisticated ones probably exist, perhaps by only examining a subset of the total evidence where the size of the subset would vary for different instances. For example, random walk models incrementally accumulate evidence and make a decision once a satisfactory threshold of certainty is achieved [Ratcliff and Murdock Jr., 1976; Ratcliff, 1978]. However, a full account would need to develop the following:

- Describe mechanisms that calculate and report pieces of evidence at varied times.
- Account for the mechanisms' plausibility.
- Show how the decision procedure working with incremental evidence accumulation (as constrained by the evidence calculating mechanisms) is consistent with human response times.
- Show how the decision procedure *remains* consistent with other human data (*e.g.* effects on accuracy and learning rates).

Thus, a random walk implementation could conceivably account for response time data. Nevertheless, further research is still required, filling in the necessary details, before such an account can be seriously evaluated.

SCA, on the other hand, is a radical departure from the generic model. Rather than considering all evidence at one critical juncture, SCA serially and systematically probes production memory. The sequence of operations guiding search are of varying length, lending to a varied response time. Other models deviating from the generic model are the classifier model, COBWEB, discrimination nets and 'settling' nets [Schyns, 1991] which I discuss later in the following sections.

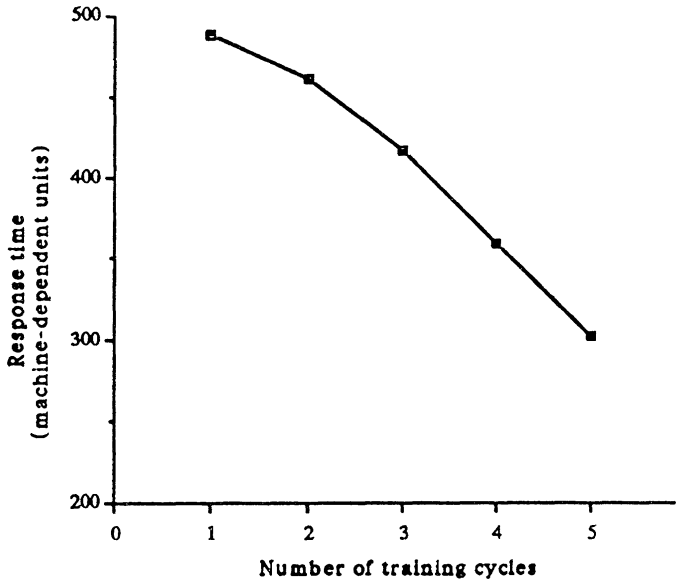


Figure 5.2: Practice effect in terms of response time

### 5.1.1 Practice effect

As discussed in the Chapter 4, performance time decreases as SCA masters category prediction. Using the same results from that chapter, Figure 5.2 shows SCA's computational runtime in relation to the number of exposures. The y-axis specifies the average runtime per test instance in terms of a machine-internal time unit.

Of the reviewed models, only Holland and Reitman's model predict a similar improvement in response time performance through the course of learning [Holland and Reitman, 1978]. They report that over the course of learning, less production applications are required before predicting the category. This is the result of rules receiving greater feedback when less applications are required. Greater feedback comes with less rule applications because the overall feedback is shared with less rules.

Models that conform to the generic model do not experience this improvement of performance. Moreover, discrimination nets and COBWEB do not either, as learning for both of these models increases the depth of their conceptual structures, thus requiring more time to descend them. While Schyn's (1991) application of a Kohonen net does not report an increase in response time performance, it is conceivable that the model would exhibit a speed-up over the course of learning.

### 5.1.2 Typicality effects

People have a general understanding that some objects are more typical instances of categories than other objects. These instances have particular properties as well as particular behavioral effects relative to less typical members. A good example of the general phenomenon is the comparison of a robin and a penguin inside the bird category. Generally, the robin is considered a more typical bird. While this does not mean that a robin is more of a bird than a penguin is, it appears that typicality plays a role in how categories are processed. Several behavioral effects occur which vary as a function of an instance's typicality [Rosch, 1978]. These are 1) typical instances are generally processed faster than less typical ones, 2) typical instances lead to fewer errors in category prediction and 3) typical instances are frequently given as an example to a category. These results have led scientists to believe that categories do not have sharp, strongly defined boundaries. Instead, category membership is thought to lie on a continuum.

Several approaches have addressed this apparent 'fuzziness' of category boundaries. These include fuzzy sets [Zadeh, 1965] or frequency distributions [Fisher, 1987; Lebowitz, 1987], but, as Bergadano

Table 5.1: Training and testing data for typicality effects

| Category | Attributes D1 | D2 | D3 | D4 | D5 | Similarity Score | Typicality Group |
|---|---|---|---|---|---|---|---|
| A | 1 | 0 | 0 | 1 | 1 | 12 | Low |
| A | 1 | 1 | 0 | 0 | 0 | 12 | Low |
| A | 0 | 1 | 0 | 0 | 1 | 14 | Mid |
| A | 0 | 0 | 0 | 1 | 0 | 14 | Mid |
| A | 0 | 0 | 0 | 0 | 1 | 16 | High |
| A | 0 | 0 | 0 | 0 | 0 | 16 | High |
| B | 0 | 1 | 1 | 0 | 0 | 12 | Low |
| B | 0 | 0 | 1 | 1 | 1 | 12 | Low |
| B | 1 | 0 | 1 | 1 | 0 | 14 | Mid |
| B | 1 | 1 | 1 | 0 | 1 | 14 | Mid |
| B | 1 | 1 | 1 | 1 | 0 | 16 | High |
| B | 1 | 1 | 1 | 1 | 1 | 16 | High |

*et al.* (1992) point out, once a system has explicitly defined a measure for category membership, category membership once again has a "fixed, well defined meaning." Rather than speculate on whether SCA's approach captures the spirit of fuzzy category membership, a more objective position can be taken by focusing on how well the external behavior of the model fits the external behavior of humans.

This differs from many previous approaches that make response time predictions based on an internal matching metric. For example, COBWEB shows a correlation between typicality (as determined by response time) and the degree of match to one of the system's internally represented concepts [Fisher, 1988]. COBWEB does not address whether there exists a process that would implement the model while also producing varied response times in accordance with the match metric. Moreover, none of the models conforming to the generic model implementation could model these response times.

One exception is the application of a Kohonen net [Schyns, 1991] where the response times are strongly modeled in that the system actually requires different amounts of time to process instances of different degrees of typicality. The varied response time results from a 'settling' process, which is a search for a stable state influenced by bottom-up constraints of the unsupervised Kohonen net and top-down constraints of a supervised learning component.

Because the supervised learning task only requires the category name to be predicted from the training example, the third effect (giving a typical instance as an example) does not apply. However, response times and error rates do apply to the task.

To test the model's behavior on instances with different degrees of typicality, it is presented with an artificial dataset (*i.e.* a set of artificial stimuli). Typicality can be defined by determining an instance's similarity to the other instances in the same category. Rosch, Simpson and Miller [Rosch *et al.*, 1976b] show in several experiments how response times and errors vary in accordance to this metric. In particular they report that humans categorize the more typical instances with faster response times and less errors. Table 5.1 shows the stimuli used for testing typicality effects. For this data, there are two categories: A and B. For each category there are six instances, each with five attributes. Each of the attributes can have only two values: 0 or 1. These values serve as symbolic representations of features (*e.g.* color, shape, size, *etc.*) that humans perceive when undergoing a categorization experiment. A given instance has a similarity score that is the sum of how many features the instance shares with the other instances in the same category. This is the same definition of typicality as in the Rosch *et al.* study. Based on this score, the typicality is rated as low, middle, or high.

In testing the model, I presented the data for five training cycles while interleaving performance

Table 5.2: The effect of typicality on error rate and response time

| Typicality group | Error rate | Response time |
|---|---|---|
| Low | 12.0% | 92.1 |
| Mid | 8.2% | 76.5 |
| High | 5.6% | 65.7 |

trials (predicting the category name) after each training cycle. This process was done fifty times[1] and the results were then averaged. Table 5.2 shows the results. The response time is a machine-dependent measure that allows comparisons across typicality groups. Even though error rate as a function of typicality is not very diagnostic of many categorization models, it has been included for the sake of completeness. Here the error rate is the average percent of incorrect category guesses. The results, in accordance with human behavior, indicate a faster response time and a lower error rate for the more typical instances.

The model's typicality effects are a result of how SCA incrementally acquires new prediction rules. New prediction rules result by first matching a training instance with an already existing prediction rule and then adding a new feature to the matching rule's conditions. Training instances with a high typicality are more likely to match more features, thus creating more specific rules. An instance will match a more specific rule faster because SCA searches for the most specific prediction rules first. A specific rule match is also more likely to be correct because it has a higher probability of including more relevant features in its conditions.

The typicality effects that the model exhibits should be further qualified. For the results presented here, typicality is defined as the total similarity of an instance with other instances in the same category. In addition to this intra-category typicality, Rosch and Mervis (1975) report inter-category typicality where typicality is defined by the degree of contrast of competing categories. The model would have difficulty in predicting faster response times for instances with high inter-category typicality. Whether an instance is close to a contrasting category makes no difference in SCA's response time. This is because SCA, in searching from the most specific to less specific, stops searching as soon as it finds a prediction; having a slightly less specific rule predicting a contrasting category has no bearing on the response time.

The design choice of having SCA perform a simple search from specific to general is perhaps at fault in its failure to account for inter-typicality resonse time. This design decision is also the least motivated. It is not the consequence of any architectural constraints, nor is it functionally motivated. However, a search from general to specific is not a plausible strategy either. This approach would not successfully predict the practice effect nor the intra-category typicality effect, as its behavior would be similar to a discrimination net. Instead, I am currently exploring more sophisticated approaches that would start with specific rules, but would continue search with different feature orders if conflicting rules are first retrieved. Chapter 6 describes a preliminary implementation that does account for inter-category typicality.

### 5.1.3 Response time for training

With SCA, process time not only varies for performance; it also varies in the time required to process a training example. Furthermore, because the model learns new productions by correctly performing on a training instance, the same response time differences hold for training instances as they do for performance. To my knowledge, no experimental study confirms or refutes this result, and thus it serves as a novel prediction for human learning.

[1] Unless otherwise indicated, averaging the results over fifty trials was more than sufficient for achieving the significance necessary for all of the qualitative comparisons in this chapter.

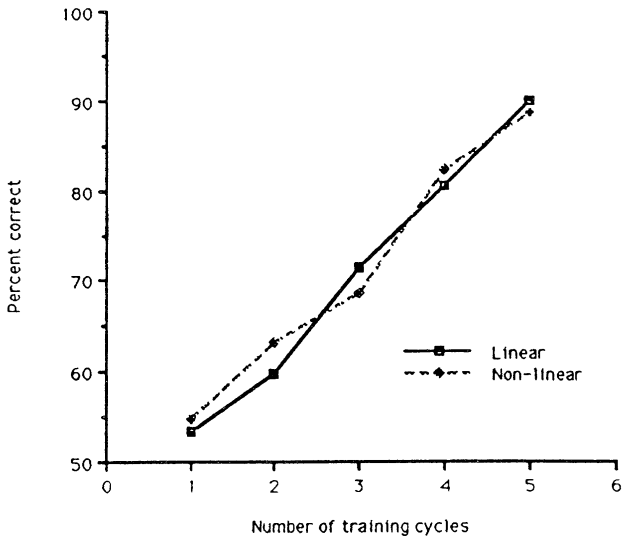


Figure 5.3: Linear vs. non-linear learning rates

### 5.1.4 Asymptotic performance

One aspect of the model is its asymptotic performance on all of these datasets. Eventually, given enough repetitions, the prediction rules' conditions will include all of the features found in the instance. At this point, the distinction in response time behavior as a function of typicality is lost. In practice, however, if real instances contain many features, this asymptote in performance may never be reached leaving typicality distinctions intact. Chapter 6 discusses other possible solutions to this problem.

## 5.2 Linear separability

In previous studies, the relative difficulty of learning linear and non-linear category structures has been used to diagnose the underlying human process [Medin and Schwanenflugel, 1981; Kruschke, 1991]. Since SCA represents categories in terms of many prediction rules, its concept representation power is ultimately equivalent to that of exemplar-based models. Thus, unlike prototype models and single-layer nets, it is not limited to linearly bound concepts. Furthermore, because the feature selection heuristic evaluates feature relevance in conjunction with other features included in the match, the learning rate of non-linear categories is often the same as a linear category set.

Medin and Schwanenflugel (1981) report a set of experiments where subjects learned linear and non-linear category sets at roughly equal rates. For comparison, I have tested my model with the stimuli from that paper (from the first experiment). The model was executed with five training cycles interleaved with performance runs after each training cycle. This process was repeated fifty times with the results averaged by cycle number. Figure 5.3 shows the learning rates of both sets. The x-axis indicates the number of times the training set was presented to the model. The y-axis indicates the accuracy (in terms of percent correct) of the model performing on the dataset after the given number of training cycles. The graph indicates the model learning the two category types at roughly the same rate, thereby replicating the Medin and Schwanenflugel result.

## 5.3 Learning relevant feature dimensions

Recall that SCA learns general rules first and then gradually acquires more specific rules. This learning strategy makes a strong prediction that in general SCA will learn categories defined by a

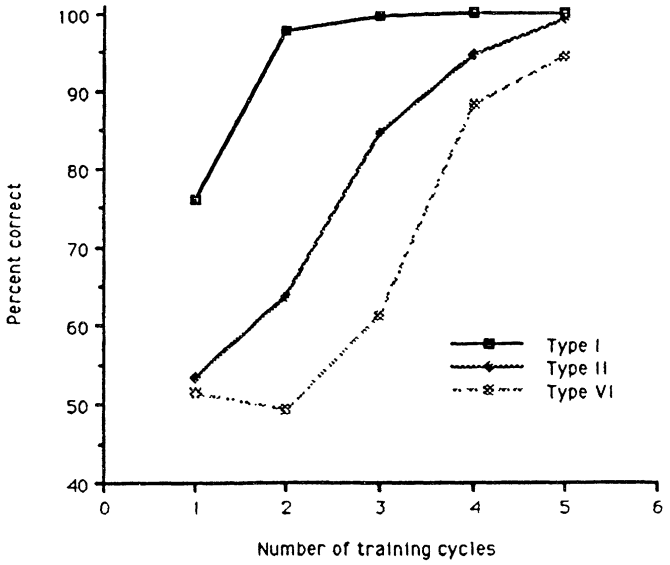


Figure 5.4: Learning rates as a function of category complexity

small number of features faster than categories defined by many features.

Shepard, Hovland and Jenkins (1961) report several experiments that reveal the effect the number of defining features has on human learning rates. They tested humans on six different category types. Of the types, Type I, Type II and Type VI were tested extensively.[2] The Type I categories were simply defined by one feature. Type II categories were defined with an exclusive-or logic using two features. Type VI categories required three features in order to correctly guess the category. Of these three types, Shepard *et al.* report that subjects learned Type I the fastest, Type II next, and Type VI the slowest.

Using the same stimuli from these experiments, I compare the model's performance with the experimental results. Again, the model was executed with five training cycles interleaved with performance runs after each training cycle. This process was also repeated fifty times with the results averaged. Figure 5.4 shows the model with the equivalent behavior.

A more general version of this phenomenon is discussed in Kruschke (1991). Learning a category defined by one feature is referred to as *gating* whereas learning a category defined by several features is called *condensation.* Kruschke reports that gating is an easier task than condensation and claims this is a robust phenomenon for human category learning. Furthermore, he includes results showing that the back propagation model [Rumelhart *et al.*, 1986b] and the configural-cue model [Gluck and Bower, 1988a] do not make predictions favoring the gating task. Models which learn and evaluate features in parallel would have extreme difficulty favoring the gating task.

Kruschke's ALCOVE [Kruschke, 1990] successfully models the gating versus condensation distinction by incorporating an additional module that selectively learns to attend to relevant features. In contrast, SCA's bias towards gating is a direct consequence of initially learning general rules. While SCA also has a module for guiding feature selection, it is inherent to the rule learning performance system. Models which do not learn to attend to selective features, such as the other models presented in this paper, may have difficulty accounting for this distinction in learning rates.

In addition to previously replicated phenomena, SCA makes one novel prediction with the Shepard *et al.* categories. Let us consider the learning curve of Type VI categories in Figure 5.4. Even after two training cycles, SCA is still performing at chance, *i.e.* at essentially 50%. This behavior is a consequence of the model's learning properties interacting with the difficulty of Type VI categories. In order to guess these categories at an accuracy rate greater than 50%, all features must be consid-

[2]Because Types III, IV and V were tested with fewer subjects with quite erratic results, learning curves for each of these types were not presented in their report.

ered in making the prediction. Thus, for SCA to perform above chance, it must have acquired some rules whose conditions include all three features. However, the acquisition of such rules is delayed by two factors. Firstly, SCA incrementally acquires more specific rules through multiple exposures to the stimuli. Secondly, the acquisition of more specific rules is retarded through inconsistent feature selection. Learning Type VI categories invariably leads to inconsistent feature selection since the prediction feedback for rules of one or two features is always poor. Thus, SCA is continually selecting alternate feature orderings in an attempt to achieve better feedback.

This prediction distinguishes SCA from essentially all other models. While most models learn Type VI categories with the most difficulty, they do not predict performance to be at chance for the first several training cycles. The reason is they all allow the possibility (perhaps the probability is very low) of factoring in all three features into the concept definition, even during the earliest stages of learning. For example, ALCOVE, while it has attentional focusing governed by weights, still factors in all features. The attentional weights are never zero; otherwise the model could not learn to optimize the weights. Thus, even though some features are more weakly represented than others, enough conceptual information has been acquired to predict correctly at a rate greater than 50%.

There is some evidence that SCA's prediction is correct. While Shepard *et al.* (1961) do not report error percentages, Anderson (1991) reports some unpublished human data [Nosofsky and Gluck, 1989] that shows performance relative to chance. Here subjects learning Type VI categories perform at less than or equal to chance through the first two blocks (two epochs per block) of training. Unfortunately the data is sketchy and there is no confidence interval for the error rate. Thus the evidence is only suggestive and the prediction still remains to be confirmed.

## 5.4 Role of strategies

Medin and Smith (1981) report some experimental results that can simultaneously test several structural properties of SCA. In particular, the results report the effect of some individual stimuli on response time and error, and the effect of instructed strategy on response time and error. Subjects, divided in three groups, were all trained on the stimuli presented in Table 5.3. Furthermore they were all told they need to learn how to categorize each stimuli into one of the two categories. Subjects in the first group were not given any particular strategy or hint on how to learn the categories. Subjects in the second group were instructed to pay particular attention to one attribute (referred to as D3 in Table 5.3). Furthermore they were informed that this feature is particularly diagnostic for determining the correct category, but that they would also have to learn exceptions. Subjects in the third group were instructed to learn the categories by garnering a "general impression" of both categories. For all three groups, training proceeded by making subjects first guess the category and then receiving feedback as to whether the guess was correct. Training continued until either a subject successfully completed a pass where all categories were correctly named or the subject completed 32 passes through the stimuli.

After training, the subjects were required to fulfill several performance tasks, of which one task tested errors rates and response times (the subjects were told their responses were being timed) on the original stimuli. Table 5.4 presents human response times (in seconds) and error rates for subjects' performance according to learning strategy. For this task, several interesting results were statistically significant:

- **Overall, stimulus 7 was easier to classify than stimulus 4 in terms of error rate and response times.** This result is interesting since stimulus 4 is closer to the central tendency of category A (all 1's) than stimulus 7. On the other hand, there are two other stimuli (15 and 4) in A which differ from 7 by only one feature whereas there is only one stimuli (7) that is closely similar to 4.

- **The strategy impacted the subject's average performance.** On average, subjects in the prototype group performed the worst in terms of response time and accuracy. Subjects

Table 5.3: Stimuli learned with different strategies

| Category | No. | Attributes D1 | D2 | D3 | D4 |
|---|---|---|---|---|---|
| A | 4 | 1 | 1 | 1 | 0 |
| A | 7 | 1 | 0 | 1 | 0 |
| A | 15 | 1 | 0 | 1 | 1 |
| A | 13 | 1 | 1 | 0 | 1 |
| A | 5 | 0 | 1 | 1 | 1 |
| B | 12 | 1 | 1 | 0 | 0 |
| B | 2 | 0 | 1 | 1 | 0 |
| B | 14 | 0 | 0 | 0 | 1 |
| B | 10 | 0 | 0 | 0 | 0 |

Table 5.4: Effects of stimulus type and strategies on human subjects

| Stimulus | Standard | | Rule-plus-Exception | | Prototype | |
|---|---|---|---|---|---|---|
| Number | RT | ER | RT | ER | RT | ER |
| 4 | 1.11 | .05 | 1.27 | .03 | 1.92 | .07 |
| 5 | 1.34 | .14 | 1.61 | .11 | 2.13 | .18 |
| 7 | 1.08 | .03 | 1.21 | .01 | 1.69 | .04 |
| 13 | 1.27 | .09 | 1.87 | .15 | 2.12 | .14 |
| 15 | 1.07 | .02 | 1.31 | .01 | 1.54 | .07 |
| 2 | 1.30 | .12 | 1.97 | .20 | 1.91 | .12 |
| 10 | 1.08 | .03 | 1.42 | .02 | 1.64 | .03 |
| 12 | 1.37 | .19 | 1.58 | .10 | 2.29 | .16 |
| 14 | 1.13 | .06 | 1.34 | .04 | 1.85 | .06 |
| M | 1.19 | .08 | 1.51 | .07 | 1.90 | .09 |

in the default group had the fastest response times while subjects in the rule-plus-exception group had the least errors.[3]

- **The strategy impacted the relative difficulty of some stimuli.** Stimuli 13 and 2 are exceptions to the hint given in the second group. These were the hardest to learn for subjects in this group. For subjects in the other groups, stimuli 5 and 12 were at least as hard to learn as stimuli 13 and 2.

These three major results from the timed performance task are also consistent (in terms of the relative difficulty of the stimuli) with error rates that occurred during learning and during an untimed transfer task performed immediately after learning.

Medin and Smith were able to fit the context model [Medin and Schaffer, 1978] to account for these results. One aspect of this model affords the assignment of attentional weights corresponding to each feature. By assigning a different set of weights for each strategy, the model can account for the differences in performance for each strategy. The weights are thus parameters which the researchers select in order to minimize the performance difference between the model and human subjects.

The human data on timed categorization can also be compared with SCA's behavior. The task is particularly appropriate for several reasons. First, it was a timed task where subjects generally

[3] Medin and Smith do not report whether these pairwise differences were significant.

Table 5.5: Effects of stimulus type and strategies on SCA

| Stimulus Number | Standard | | Rule-plus-Exception | | Prototype | |
|---|---|---|---|---|---|---|
| | RT | ER | RT | ER | RT | ER |
| 4 | 0.16 | .04 | 0.09 | .01 | 0.78 | .10 |
| 5 | 0.61 | .11 | 0.19 | .02 | 1.14 | .15 |
| 7 | 0.13 | .01 | 0.06 | .00 | 0.77 | .08 |
| 13 | 0.63 | .11 | 0.62 | .11 | 1.11 | .14 |
| 15 | 0.15 | .00 | 0.09 | .00 | 0.82 | .07 |
| 2 | 0.73 | .21 | 0.61 | .13 | 1.25 | .22 |
| 10 | 0.18 | .00 | 0.13 | .01 | 0.95 | .09 |
| 12 | 0.73 | .21 | 0.27 | .03 | 1.23 | .20 |
| 14 | 0.48 | .04 | 0.18 | .01 | 1.20 | .14 |
| M | 0.42 | .08 | 0.25 | .04 | 1.03 | .13 |

responded within 1–3 seconds, approximately corresponding to SCA's time band. Second, the task was a category naming task, as opposed to a category verification task. Furthermore, SCA, as defined within the context of a UTC, is committed to how strategies can impact its performance. For the most part, SCA is an automatic process which acquires new rules as it guesses what category an object belongs to. Deliberate control is only granted to feature selection, and thus feature selection is the only process that is penetrable by additional knowledge, including strategic advice. Therefore, it is reasonable to assume that the strategic advice directly affects how features are selected.

When considering SCA within the context of Soar, each advised strategy suggests a particular feature selection implementation for SCA:

- **Default strategy.** In this case, no strategic advice was given to subjects. This suggests the standard feature selection strategy (using averaged feedback based on conflicting rules) that is used in all other simulations in this chapter.
- **Rule-plus-exception strategy.** Here the subjects were instructed to focus on the third feature and then learn exceptions. SCA can follow this advice to the extent that it always primarily focuses on the third feature, *i.e.* it always removes other features from the internal representation first. However, the model does not afford the possibility of deliberately learning exceptions. Rather, it must learn exceptions by learning more specific rules. The simplest approach is to focus randomly on features in learning more rules.
- **Prototype strategy.** For this strategy, subjects were asked to garner a general impression of the category. Ideally, this would mean that the learner would focus equally on all features at the same time. However, this is not possible for SCA; initially, until rules with multiple features are learned, focus on only one feature is possible. In order to comply best with the strategic advice, SCA must choose to focus randomly in learning rules.

Each of these versions of SCA was trained on the stimuli presented in Table 5.3. After training through five passes of the stimuli set,[4] the model completed the performance task. The results in Table 5.5 are averages of 10,000 runs. The error figure is the fraction of times the model guessed wrong in categorization. The response time is the average number of iterations of feature removal before a prediction rule matched. The number of runs was sufficient in attaining a 95% confidence interval of $\pm 0.01$ for both error and response time.

SCA's behavior is consistent with the statistically significant results reported in Medin and Smith. I will discuss these points one by one. But first, let us review how accuracy and response time can

[4] At this point, error rates were roughly the same as the human data.

vary. Accurate performance is the result of matching rules whose conditions 1) have discriminating features and 2) have many features (more specific rules). Feature selection strategies that focus on discriminating features produce rules with discriminating conditions. Strategies that consistently focus on features in the same order (*i.e.* stable strategies) acquire and access more specific rules. Furthermore, stimuli that share combinations of features with stimuli in the same category also lead to the acquisition and access of more specific rules. Fast response time is the result of matching specific rules. Discriminating features do not play a direct role in response time for SCA.

With these general performance characteristics, the specific results and their reasons are:

- **Overall, stimulus 7 was easier to classify than stimulus 4 in terms of error rate and response times.** Stimulus 7 shares more combinations of features with other stimuli in the same category than 4. Thus, more specific rules match the stimulus, resulting in fewer errors and faster response times.

- **The strategy impacted the subject's average performance.** The prototype strategy (SCA randomly chooses features) is an inconsistent strategy that does not always focus on discriminating features. Thus, this strategy results in poor performance, in terms of both accuracy and response time. The other two strategies are both moderately consistent in selecting features. The rule-plus-exception strategy always keeps the same feature in the object description, with the rest chosen randomly. The default strategy experiments with feature selection orderings, but then somewhat stabilizes as good ones are found. Similarly, both strategies generally choose discriminating features.

- **The strategy impacted the relative difficulty of some stimuli.** The most problematic stimuli for the rule-plus-exception strategy are those which are exceptions to the rule (nos. 13 and 2), whereas, for the default model, these are just as difficult as two other stimuli (nos. 5 and 12). SCA's rule-plus-exception strategy has difficulty with the exceptions because it is primarily focusing on the least diagnostic feature for these stimuli.

In further evaluating the models' fits to human performance, the degree of difficulty (according to error rate) was ranked, by strategy, for both the human data and the models' data. Using a Pearson correlation coefficient, a quantitative measure of correlation ranging from 1 (perfectly correlated) to -1 (perfectly inversely correlated), the ranking of the human data was compared to the ranking of SCA's data. The coefficients for the default, rule-plus-exception, and prototype strategies were respectively .941, .966, and .790. As a point of reference, the coefficient between human data with the default strategy and human data with the rule-plus-exception strategy was .836. Thus, SCA's rule-plus-exception model was a better predictor of human behavior for this strategy than was human behavior under the default strategy. On the other hand, the coefficient of human data between the default and prototype strategies was .912 (compare to .790 for SCA), suggesting the pure random strategy does not strongly account for aggregate human data. Without analyzing individual performances, the reason is not clear. It may be the case that some of the subjects disregard the advice, thus producing an aggregate result lying between the behaviors of the two versions of the model. Or possibly, subjects are mixing strategies during learning and performance. The extended response of the prototype suggests that this hypothesis is plausible. In any case, despite the pure random strategy failure to account strongly for the difficulty ranking, it nevertheless suggests why the prototype strategy is inferior in performance (in terms of both error rate and response time) than the other strategies.

Another point of reference for comparing the ranking correlations between SCA and human data is the ranking correlations between the human data performing the speeded task and human data performing unspeeded task with the same stimuli. In addition to the speeded classification task, Medin and Smith had subjects perform classification where the responses were not timed. The tested stimuli were the same used in the speeded task, but also included 7 additional stimuli which the subjects did not see during training. Taking the 9 that were seen during training, their ranks were compared with those for the stimuli's ranks in the speeded task. The correlation coefficients (by respective strategy) were .727, .916, and .929 (as compared to .941, .966, and .790 for SCA).

Thus, for the default and rule-plus-exception strategies, SCA served as a better predictor on speeded classification than human data from the otherwise identical task of untimed classification.

For further evaluation, SCA's behavior was compared with the subjects' performance on the untimed classification task. As previously mentioned, the task included stimuli not present during training. The learning experiment was repeated for all three strategies. After 4 training cycles,[5] SCA was tested in its ability to predict the categories for all 16 stimuli (9 seen during training, plus 7 novel stimuli). Averages of error rates were taken from 10,000 runs for each strategy. The resulting ranking correlation coefficients were respectively .841, .867, and .644 for the default, rule-plus-exception, and prototype strategies. As a benchmark, Medin and Smith were able to fit the context model in achieving respective coefficients of .90, .98, and .96.

The apparent success of the context model relative to SCA is misleading. By choosing appropriate feature attention weights, Medin and Smith demonstrate how the context model is *consistent* with human learning for all three learning strategies. The use of these parameters played a critical role in the context model's fit. Medin and Smith report, "The parameter constraints are fairly tight in that values more than a few percentage points away yield substantially poorer fits for both [the context and prototype] models."

SCA takes a further step by having *a priori* commitments to where and how strategic advice should alter the acquisition process. In particular, the model, as it is part of a larger comprehensive theory, is more constrained in how different strategies could be implemented. It thus makes stronger predictions since it does not have as many degrees of freedom afforded by parameters.

Another consideration is that SCA is not as well suited to model untimed responses as timed responses. The timed responses took approximately 1 to 3 seconds. This falls well within the time constraints of SCA's process in the context of the Soar architecture. Presumably, for the untimed task, subjects deliberated longer with their response. The extended deliberation time allows people more flexibility in the strategies they use, and thus allows them to deviate more from the default process shaped by architectural constraints [Huffman *et al.*, 1993].

## 5.5 Basic level superiority

Some categories are more general than others and often are supersets of other categories. For example, the furniture category is a superset of the chair category. Furthermore, the chair category is a superset of the rocking-chair category. As we can see, words can have different levels of specificity. The categories at the basic level usually occur somewhere in the middle of the different levels of specificity. The category referred to by 'chair' is a good example. Categories at the basic level have been identified as having the highest level category inclusion where the objects in the category still share many features [Rosch, 1978].

In this section I review the model's learning behavior as a function of the level of the category being learned. Two reported phenomena are relevant to the supervised learning task. First, children learn basic level categories before learning categories at other levels [Rosch *et al.*, 1976a]. Secondly, queries for basic level categories have a faster response time [Rosch *et al.*, 1976a; Murphy and Smith, 1982].

In testing the model, I ran it on the data (from experiment 1) used in Murphy and Smith (1982). As they point out, it is not possible to evenly compare learning rates of categories at different levels of specificity. If the number of training instances per level are held constant, then categories at the superordinate level receive more training instances per category. On the other hand, if the number of training instances per category are held constant, then the subordinate group of categories have an advantage in that they receive more instances across their level.

Murphy and Smith ran a experiment favoring the superordinate group (holding the number of instances per level constant). Despite the advantage that the superordinate set has over the basic set, they reported faster response times and lower error rates at the basic level (the difference in error rates, however, were not significant). Giving the same advantage to the superordinate group,

[5] As before, training stopped when error rates approximated those of the human subjects. It is not clear why subjects performing the untimed task did worse than when they were (knowingly) being timed.

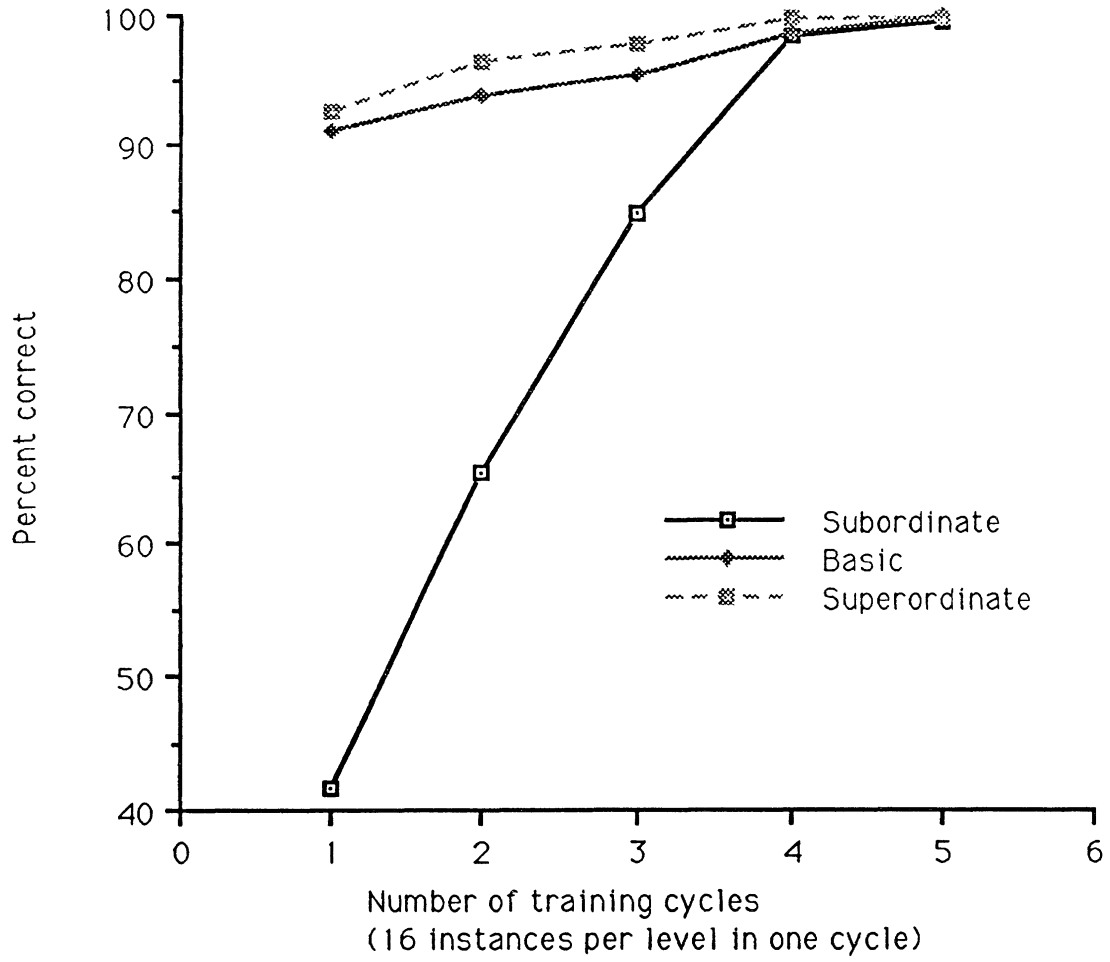


Figure 5.5: Learning rates as a function of category level with equal number of training instances per level

I ran the model on their data. Figure 5.5 shows the learning rates at the different levels. Faster response times closely correlated with the faster learning rates.

The model's predictions differ from the Murphy and Smith results in that the superordinate level has slightly fewer errors than the basic level. The discrepancy may originate in the difference of tasks that the human subjects are performing versus the model's task. In their experiment, the subjects are given the category name and then the testing instance. The subjects must then indicate whether the instance belongs to the category by answering true or false. The model, on the other hand, is given the instance and must predict the category. Thus, the model confronts a more difficult task for categories at lower levels of generality because it must guess from more possibilities. There remains the challenge of extending the model to perform the recognition task.

For completion, I also present the same results however evening out the number of training instances per category (Figure 5.6). In this case, the basic level has, on average, the highest accuracy. I cannot claim that normalizing the results in this manner necessarily compensates for the discrepancy in tasks. However, I do claim that the model predicts a basic-level superiority under qualified conditions.

The basic-level superiority the model exhibits stems from the model's tendency to favor categories with a higher degree of intra-category similarity. Recall from the typicality results that similarity

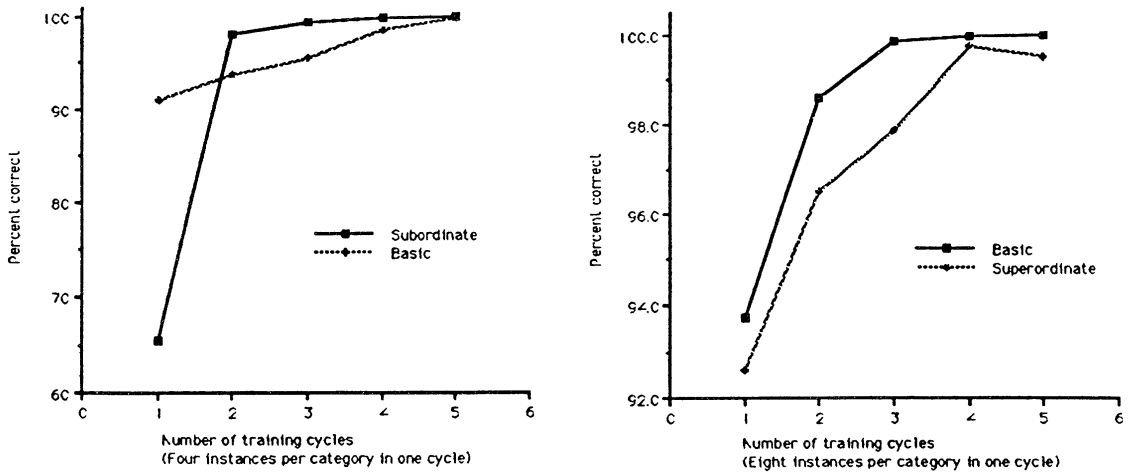


Figure 5.6: Learning rates as a function of category level with equal number of training instances per category

between instances leads to faster construction of more specific rules and thus prediction is faster and more accurate. The favoring of categories with a high amount of intra-category similarity puts superordinate categories at a disadvantage in relation to the more specific category levels. However, this disadvantage is not enough to compensate for the superordinate advantage of receiving twice the number of training instances per category as reported in Figure 5.5. When the number of instances per category are held constant, the basic level has the superior learning rate. It still holds an immediate advantage over the subordinate categories. This result is due to the fact that subordinate categories require more predictive features in the prediction rules in order to accurately differentiate between the many specific categories.

The model's explanation of basic-level superiority is consistent with Gluck and Corter's (1985). They claim that categories at the basic level maximizes the metric *category utility*. Category utility can be understood as a tradeoff between cue validity (feature predictiveness) and category validity (intra-category overlap of features). Of the models compared in this paper, only COBWEB addresses basic-level superiority. COBWEB [Fisher and Langley, 1990] explicitly uses this metric in forming its internal concept representation. This approach enables COBWEB to make correct response time predictions based on its matching metric. As with the typicality effects, COBWEB depends on an unspecified implementation in order for this internal metric to manifest itself as response time. On the other hand, SCA does not explicitly calculate cue and category validity. However, these factors indirectly influence SCA's varying ability to learn categories at different levels and with different response times.

## 5.6 Practice effect

The model also manifests a practice effect for accuracy that is rarely explicitly described in the psychology literature. In examining the various learning rate figures of the last several sections (see also Figure 4.3 and its explanation), we see a steady increase in performance as the model is repeatedly trained on the same data set. The primary reason for this is SCA's conservative incremental learning of more specific rules, one feature at a time. Also, the feature selection heuristic plays a role as its selection order stabilizes.

The practice effect is an important distinction between the model and other symbolic approaches. For example, exemplar-based models that acquire and keep whole examples for classification cannot show a practice effect by repeating the same training examples.[6] To what extent the practice effect

[6] Exemplar-based models with a device that probabilistically stores example descriptions or even partial descriptions

exhibited by the model mirrors behavior in people is still open for more experimental research, both with human subjects and the model.

## 5.7 Underextensions in early language acquisition

The extent to which the learning process used by people in controlled experiments is the same process applied in everyday situations remains an open issue. Thus, there is the possibility that a model which successfully explains behavior in an experimental setting may not apply in a more natural setting. To a certain degree, I have guarded against this possibility by focusing on behavioral phenomena that is evident across a wide range of experimental paradigms. Typicality effects are one example, which have been observed in several tasks (*e.g.* category naming, membership verification, taxonomic relationships) using a variety of stimuli (*e.g.* faces, alphanumeric characters, stick figures, geometric objects). Another means of securing external validity is to observe learning in more natural settings. Here I apply some general principles obtained by observing children acquiring word categories.

One early theory of lexical acquisition is the *semantic feature hypothesis* [Clark, 1973], which Carey (1982) later described as *component-by-component* acquisition. This theory hypothesized that children learn word categories by initially acquiring incomplete definitions with only one or a few defining features. Over time, the word definition is completed by incrementally adding the remaining required features. While the theory made strong predictions and seemed to explain overextensions in early language acquisition, the theory was criticized on the grounds that it was limited to learning 'classical' concepts, thus failing to capture family resemblance concepts [Carey, 1982], and could not account for underextensions present in early language learning [Reich, 1976; Dromi, 1987].

An underextension is the failure to apply a known word to a subset of the word's referents. In the case of word production, underextension occurs when the child fails to utter the word even though it is known that the child has produced the word before but in a more limited context. The semantic feature hypothesis does not predict underextensions since, by having only a subset of the required defining features, the concept definition is overly general in the early stages of learning. Thus, under this hypothesis, initial definitions would overextend words as opposed to underextending them.

SCA is similar to the semantic feature hypothesis since it incrementally acquires new rules that incorporate more features into their conditions. However, it does not share the same pitfalls. Because many rules represent a concept, it can extensively represent concepts learnable by instance-based models, including family resemblance categories. The SCA model does not uniquely predict overextensions in the early stages of learning. Indeed, underextensions are frequently common. Even if rules are overly general, there is no guarantee that they will be accessed. Rules that were learned under a certain feature selection ordering are not necessarily accessed by retrieval using a different feature selection ordering.

SCA can account for underextensions in another way. The supervised task as described so far is essentially a forced-choice task. That is, in order to maximize performance, the best strategy is to always pick the prediction that is most likely to be correct. In the case where several conflicting prediction rules match, SCA has no means of deciding which prediction is best. Yet, guessing is a better strategy than predicting nothing at all. Thus, SCA randomly chooses among one of the conflicting predictions. Learning categories may not always be a forced-choice task. For example, as children learn their first words, rather than always guessing a word of whose category they are unsure, they may prefer to say nothing at all.

In demonstrating how SCA can underextend categories in the initial stages of learning, I report an experiment [Miller and Laird, 1991] which used a version of SCA that, instead of always guessing, makes no prediction whenever conflicting rules matched. In this experiment, we can trace extensional errors by testing performance on one category of objects. Here training instances drawn from several

do show practice effects. Non-functional devices such as these may be plausible since they could account for possible computational limitations of the architectural hardware. However, my research methodology of only using functional devices places an emphasis on explaining human shortcomings in terms of performance tradeoffs (*e.g.* noise tolerance vs. learning rate) within the constraints of the computational resources afforded by a unified theory of cognition.

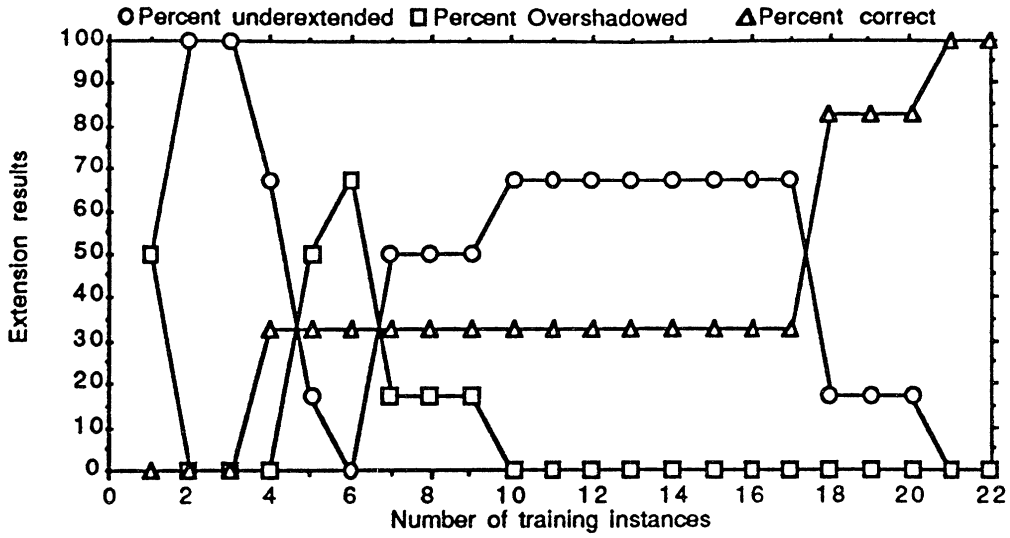


Figure 5.7: Trace of extensional errors

Table 5.6: Models and the phenomena they address

| | Phenomena | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | Gating bias | Typicality (RT) | Typicality (Error) | Practice (RT) | Practice (Error) | Non-linear learning | Strategy effects | Basic-level effects |
| SCA | X | X | X | X | X | X | X | |
| ACT | | | X | | | | | |
| ALCOVE | X | | | | X | X | | |
| Classifier | | | | X | X | | | |
| COBWEB | | X | X | | | | | X |
| Configural-cue | | | | | | | | |
| Context | | | X | | X | X | X | |
| Schyns' | | X | X | | | | | X |

categories are given one at a time. Between each training instance, a set of testing instances, all from the same category, are run on the system. Figure 5.7 shows the results of the trace. Here we see an initial onslaught of underextensions and overextensions before correct predictions dominate. The category is considered *underextended* whenever the system fails to make a prediction. The category is *overshadowed* by another category whenever a wrong response is given for the category we are testing. These are examples of overextensions of the other learned categories which are not being tested during performance runs. The particular conclusion that can be drawn from this figure is that several initial underextensions result from the model even though the system starts by learning more general productions first. This is the result of several general conflicting rules matching at the same time.

## 5.8 Summary of models and known phenomena

Table 5.6 reviews the models and documented phenomena discussed in this chapter. The concluding chapter reviews all phenomena that SCA has addressed including novel predictions.

Throughout I have attempted to speculate as to whether a particular model would either predict or be consistent with a particular phenomenon. This is always a difficult endeavor. First of all relevant process details may be missing in the model's description. Secondly, fitting the model to the data may require searching for the optimal set of parameters. Finally, there remains the possibility that I have misunderstood or been unaware of certain subtleties in the model's design and behavior.

In Table 5.6, I refrain from any speculation. Rather, I simply note with an 'X' as to whether the model has explicitly addressed the phenomenon in a previous publication. In many cases, it is conceivable, and in some cases probable, that an untested model would adequately address a phenomenon, either as it is, or with minor modifications. For example, probably all of these models exhibit typicality effects in terms of error rate.

It is also the case that some of these models address some human phenomena which I do not discuss in this dissertation. While I have focussed on effects that result as a function of instance and category structure, others have addressed effects that arise from varying the frequency and order of training examples. For example, the ACT model has addressed the effect of blocking similar examples together during training [Elio and Anderson, 1984]. Also, the Configural-cue model [Gluck and Bower, 1988b] and ALCOVE [Kruschke, 1990] have successfully modeled some base-rate effects as determined by the frequency of certain training examples.

# Chapter 6

# EXTENSIONS AND MODIFICATIONS

In this chapter, I discuss SCA's limitations while offering possible directions in which they may be overcome. Some of these limitations are direct consequences of applying the constraints of Soar. In these cases, I discuss these consequences and suggest how performance could be improved if the constraints were relaxed. Other limitations are the result of design decisions motivated by their simplicity. For these latter cases, I discuss more sophisticated approaches that may overcome the limitations of the original design.

Generally, the model's limitations can be organized into three kinds: performance limitations, behavior replication limitations and task limitations. Performance limitations address degraded performance in terms of response time, learning rate and prediction error. Behavior replication limitations address phenomena which the model fails to replicate. Task limitations address the narrowness of supervised learning and difficulties in integrating the model with other tasks.

These types of limitations correspond well to the three motivating principles, namely and respectively, functionality, fit to human data and fit to architectural constraints. While the correspondence to the first two principles are transparent, the last requires further explanation. If the architecture supports a unified theory, its constraints may provide directions to extending the theory to new tasks. Moreover, a task implementation that is well integrated with the architecture's assumptions provides links with other implementations constrained by the same assumptions.

Amidst all of the proposed research directions in this chapter, one preliminary design seems the most promising. This new model, introduced as SCA-2, has a more sophisticated search strategy while also employing a fully symbolic feature selection learning mechanism. Its design can be motivated in terms of performance, human data, and architectural constraints.

## 6.1 Improving performance

The Soar architecture has placed constraints on the computational resources that can be used by the model. Without the addition of other knowledge sources already compiled for immediate retrieval, it is doubtful if performance can be much increased when given the time frame of 1-2 seconds for each example. However, in some cases, it may be possible to improve performance on some dimensions at the cost of performance along other dimensions. In this section, I propose alternate design routes that yield some functional advantages at the expense of other performance dimensions.

First, I explore the possibility of a fully symbolic feature selection mechanism. I will demonstrate that this approach has performance advantages for simple concept structures whose objects are described with only a few features. However, for larger object descriptions, the problem may be intractable with the same mechanism. Second, I question the design decision of always learning a new rule with every new training instance. Learning new, more specific rules that mask already adequate rules may be a waste of memory. On the other hand, knowing when to stop learning could have a

high overhead cost that mitigates the advantage of the saved memory space. Third, I suggest the possibility of maintaining more prediction information such as the use of explicit frequency counts, thus allowing a more informed prediction. However, keeping additional information has added time and storage cost and may require the relaxation of some architectural constraints. Finally, I describe how other knowledge sources could be integrated into SCA, but with the added expense of searching and compiling the knowledge sources.

### 6.1.1 Stabilizing feature selection order

Pilot work in understanding feature selection techniques indicates a trade-off between two extreme strategies (see Section 4.3). The *random* strategy randomly determines a feature order which is then consistently used through all training and performance trials. Because of its stable ordering of features, more specific rules are quickly learned, thus producing a quick learning rate in the short term. The *experimental* strategy (*i.e.* the one presented with the model) experimentally chooses different orderings during training in search of a good feature ordering. At first, because of its initial instability, many general rules with different feature orderings are acquired. Furthermore, during performance, the search may miss the most specific rule because the ordering changed after the rule was acquired. Nevertheless, the experimental strategy ultimately leads to a better ordering than what the random strategy yields on average. Therefore, in the long-term, the experimental strategy has a better prediction accuracy. Moreover, the ordering learned with the experimental strategy transfers to the learning of other categories, for which there would be no instability and it would clearly dominate the random strategy.

Is there a means of combining the advantages of both strategies while minimizing the disadvantages? One promising route of exploration constrains ordering experimentation to only local reorderings, thus minimizing the instability. Upon finding that one abstraction ordering led to conflicting predictions, this strategy would modify the ordering, so that the last removed feature is swapped with the feature that would have been removed next. For example, consider the case where attributes are ordered for abstraction as follows: **color, size, texture, shape**. After abstracting out color and size from the object description, let us assume that two conflicting rules match the object's texture and shape. Using the conflict as a heuristic that the ordering is wrong, size and texture are swapped in the abstraction ordering, producing the new ordering as follows: **color, texture, size, shape**. At most, only the order of two neighboring attributes is changed with one training example.

The swapping heuristic can be further improved by immediately probing what predictions are made with the new ordering. If the inclusion of the new feature (and the exclusion of the last matched feature) also retrieves conflicting predictions, then the new proposed ordering has a dubious benefit. In this case, the old ordering is kept.

On the other hand, if the probe retrieves no predictions, there is not enough evidence to decide if the new ordering is better. Without good evidence that the alternate ordering is better, the best design would keep the current ordering in order to preserve stability. However, since the alternate ordering may be potentially better, saving the rule learned under the alternate ordering may provide the necessary favoring evidence with a future example.

Finally, if the additional probe retrieves one unique and correct prediction, this suggests that the alternate ordering is superior to the current one, and the ordering is thus changed. Whether a more specific rule should be learned with the new ordering needs to be investigated further.

The probing strategy can also be applied to performance. If the initial ordering leads to a conflicting prediction, the strategy temporarily swaps the ordering, further probing for a non-conflicting prediction. The 'swap and probe' strategy can be viewed as deviation from the 'specific to general search' strategy: once a match is found, the strategy undergoes a limited hill-climbing search.

In obtaining a preliminary assessment of this local swapping strategy, several modified versions were tested against the original experimental strategy. Data from the six category types [Shepard *et al.*, 1961] were used in the evaluation. Three different swapping strategies were used. The SCA-Swap strategy applied the swapping strategy to the training trials. The SCA-Swap' strategy applied swapping to training and performance. Both the SCA-Swap and SCA-Swap' learned a more specific

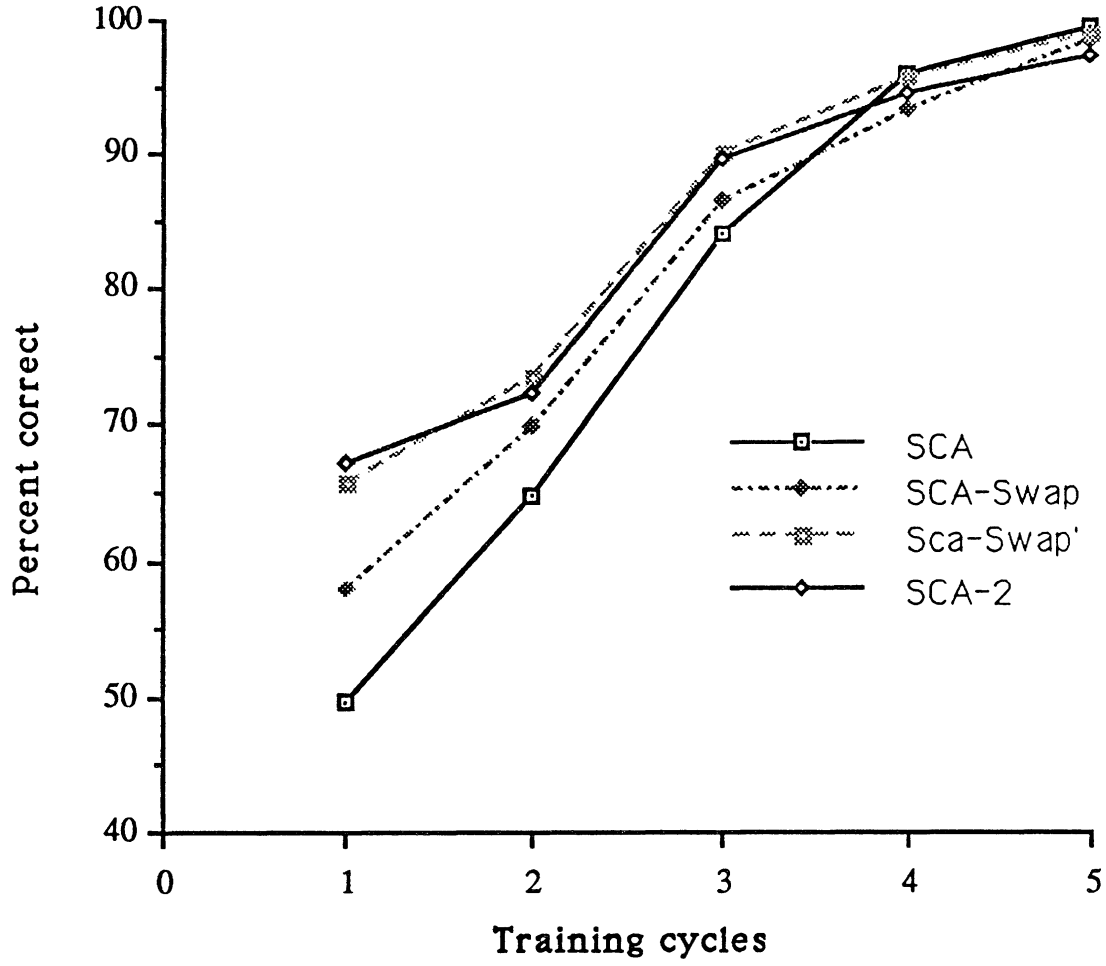


Figure 6.1: Performance comparison of swapping strategies on Type II categories

rule for every training example. The SCA-2 version swapped for training and performance, but it did not take the additional time to learn a more specific rule when a permanent order change was made.

Depending on the category type, the swapping strategy performed either the same or significantly better than the experimental strategy. Type II was one category structure with the new proposed strategies performing significantly better than the original experimental strategy. Figure 6.1 presents the results, within a 95% confidence internal of ±2%, for this type. All three modified versions performed much better than the original. In addition, the swapping during performance clearly aided in the superior performance for SCA-Swap' and SCA-2. The strategy of not acquiring a new, more specific rule whenever a permanent swap is made, did not seem to hurt the performance of SCA-2 in comparison to SCA-Swap'.

These results require a cautionary note. They are only preliminary and the functional advantage may not be evident in a broader range of applications, including transfer to novel instances and especially instance descriptions with many features. Nevertheless, these findings demonstrate a promising direction in which further inquiry is certainly warranted. They also demonstrate that a purely symbolic strategy may be sufficiently effective in learning a category's relevant features.

### 6.1.2 Imposing a learning criterion

By default, it has been assumed that, with each training example, SCA acquires a new rule, whose conditions are one feature more specific then the previously matched rule, until all possible features are included in the rule's conditions.

Extreme variants on this assumption are not reasonable. On one end, SCA could only learn more specific rules when it cannot discriminate the training example from contrasting categories. This essentially constitutes the strategy of a discrimination net. However, this approach fails to capture implicit frequency information in the rules since the mapping of one learned rule to every training instance is lost. We have also seen in Section 4.4 that this approach is less tolerant of noise than SCA's current implementation. On the other extreme, it is unreasonable for SCA to learn the maximally specific rule immediately. This strategy would carry no predictive transfer to other similar examples during performance.

Still, less extreme variants on the current implementation present interesting directions for future research. For example, there may be times when one training instance should lead to the acquisition of several rules. Another possibility is that, at some point, a sufficient sampling of training instances has been encountered, and further learning is not needed. The imposition of a learning criterion for when learning should cease might have functional advantages, as it would save storage space while still preserving the frequency information implicit in the rules. Future work would have to address what the criterion should be and if and how performance could be efficiently monitored in order to determine when the criterion is reached. It may be the case that the overhead of monitoring success, while also estimating future success, may exceed the benefit of imposing such a criterion.

### 6.1.3 Maintaining frequency counts

Whenever SCA reached conflicting predictions, it either has to guess among them or return no response at all. For example, when given the instance {`blue, spherical, fuzzy`} search may lead to the two prediction rules, [blue, spherical] ⇒'ball' and [blue, spherical] ⇒'globe'. Which should be the predicted category, ball or globe? It probably is the case that one of the categories has been associated with blue, spherical objects more than the other. For example, perhaps it has encountered the instance {`blue, spherical, smooth`} three times for 'globe' and only once for 'ball'. However, based on the information that was stored, the only thing we know for sure is that both categories have at least once been associated with blue, spherical objects, and twice with either blue or spherical objects, depending on the feature selection.

Basing the prediction off of stored frequency counts is an alternative to random guessing. Ideally, the model would keep counts of how many blue, spherical objects classified as a globe and classified as a ball that have been encountered. As stands, this strategy is unreasonable since it requires that all pertinent rules are sought out and updated. A more reasonable approach only updates the rule that would be accessed anyway, *i.e.* the rule that verifies the prediction.

This approach of counting the times a rule is accessed is easily implemented in an unconstrained programming language with only a constantly bounded time cost and linearly bounded storage cost with respect to the number of rules, and thus does not add to the system's overall complexity. Nevertheless, Soar's architectural constraints rule out any feasible implementation. Since the strategy must continually revise the contents of rules in long-term memory, the model would have to continually perform an error recovery routine that would disable the old rule and then enact a new one with an updated access count. While Soar has no native quantitative processing mechanisms, there exist feasible implementations of symbolically encoding rule frequencies. However, the time cost of error recovery would prohibit a Soar implementation that would behave within the 1–2 second time frame as required by the task.

### 6.1.4 Integrating other knowledge sources

For SCA, feature selection is a deliberate decision, meaning that any source of knowledge is potentially relevant to the selection. In the evaluation of SCA, the knowledge sources for feature selection

were data-driven, that is, they were derived from the training examples. For performance evaluation, a statistical entropy measure was calculated directly from the examples, independent of SCA's application of rules. For evaluating the fit to human data, statistical averages of prediction conflicts for each attribute were calculated. In both cases, the quantitative information ordered attribute relevance.

In this section, I suggest alternatives to data-driven heuristics. In these cases heuristic knowledge is derived from other sources such as advice from a tutor or from domain knowledge. In addition to the functional advantage, it is well documented that humans apply these kinds of knowledge in constraining concept definitions [Lakoff, 1987; Rips, 1989; Schank *et al.*, 1986; Murphy and Medin, 1985]. However, for both of these examples, knowledge is often represented qualitatively. In the case of advice, the tutor might point out certain features, suggesting that they are somehow relevant to the category's conceptual definition. For domain knowledge, the learning agent may have some causal knowledge connecting features to common actions and results. A critical problem with integrating different knowledge sources is their representations can be incompatible. Even when knowledge representations have quantitative components (*e.g.* fuzzy logic [Zadeh, 1965] and certainty factors [Shortliffe, 1976]), the quantitative knowledge is not compatible with the representation acquired through data-driven sources.

In order to integrate knowledge sources within the context of SCA, decision preferences from the different sources must be commensurable. The data-driven knowledge source applied to SCA ranks feature preferences with an averaged quantitative measure, thereby producing a linear ordering. In contrast, advice-driven or theory-driven knowledge sources produce a partial ordering. When preference conflicts exist, how should they be resolved? Which knowledge source should take priority? The data-driven knowledge may be overly general, or it may have been inaccurately biased to spurious correlations. Perhaps the data-driven knowledge should then take the least priority. On the other hand, the tutor's advice may be wrong, or there may be errors in the domain knowledge.

Resolving the incompatibilities of different knowledge sources remains an open research issue. One direction, suggested in the previous section, has the data-driven knowledge only make conservative changes to an existing feature selection order. Conservative, infrequent changes in ordering are made on the occurrence of conflicting predictions. While this policy provides a stable ordering, it may require large amounts of training to correct a poor initial ordering.

One way to remedy the problem of having a poor initial ordering is to apply ordering constraints from other knowledge sources before learning from any examples. The data-driven policy then makes conservative changes to the ordering only when the heuristic suggests that the current ordering is inadequate. In the next sections, I propose how knowledge from advice and theory is accessed and applied to the selection process.

### Induction constrained by advice

The SCA model suggests that people do not directly learn prediction rules from verbal advice. Instead, the advice influences feature selection which indirectly guides which predictions rules will be acquired and accessed. In Section 5.4, SCA's rule-plus-exception model made use of a heuristic based upon advice. However, this simulation did not address the issue of how this procedural heuristic is acquired from a declarative representation obtained through advice.

SCA, within the context of Soar, makes a deliberate decision by allowing relevant knowledge (represented as productions) to apply. If the knowledge that can immediately apply is not sufficient to make a decision, Soar automatically creates a subgoal for resolving the decision impasse (see Appendix A). At this point, additional search for knowledge takes place in order to resolve the impasse.

Figure 6.2 presents an example of selecting a feature biased by advice. We presume a tutor has instructed to focus on the object's shape and that this knowledge can be retrieved so that it is declaratively represented in working memory. By retrieving this knowledge within a feature selection subgoal, its application decides to keep the attribute of shape in the object description and therefore prefers abstracting out the other features. Unless other knowledge is available to constrain the decision, one of the remaining features would be chosen at random. Issues for further research

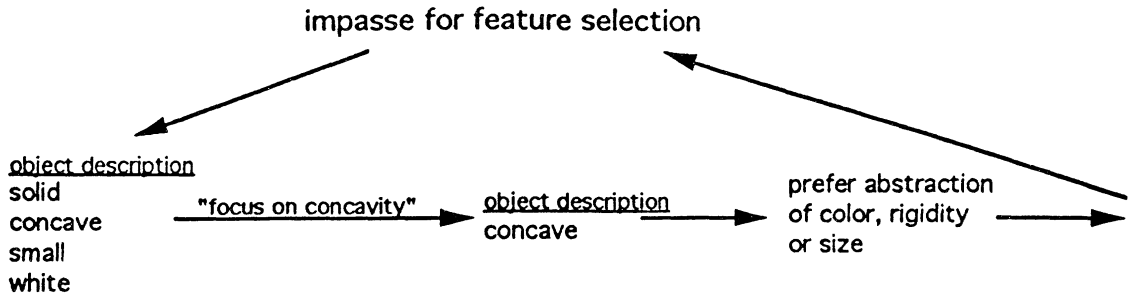


Figure 6.2: Advice–driven feature selection

include integrating natural language understanding with the construction and retrieval of knowledge preferring certain features over others.

In a sense, the above procedure is trivial. However, its 'triviality' depends on the model's *a priori* commitment to deliberate feature selection. Augmenting any other model with a similar strategy would be an arbitrary extension unless it also fixed *a priori* constraints on what decisions could be deliberately controlled.

### Induction constrained by domain knowledge

The second strategy for acquiring control knowledge for abstraction operators takes an explanation-based approach, requiring the use of goals, theories and beliefs. As with applying knowledge obtained through advice, causal knowledge can have a similar impact in deciding the relevance of features. Features that have immediate consequences on whether the agent's goals can be achieved may also be considered the most relevant in forming categories. Usually, the causal knowledge, as represented as productions in Soar, is not immediately accessible in declarative form [Rosenbloom and Aasman, 1990] and may require extensive search through the additional creation of impasses and subgoals. Further research needs to develop the processes in which causal knowledge is represented, retrieved, and compiled for future use.

Figure 6.3 provides an example of this last strategy at work. We have presented a description of a cup with the goal of learning the word. So far, no rules have matched the current object description. Because of the impasse at recognition, an abstraction operator must be chosen. Let us assume that the system already considers shape and texture to be relevant (either an innate bias or learned from previous experience). This leaves a choice between color and size. In this example, the color attribute is removed from the object description. The modified object description is then processed through the domain theory. Included with the object description is the context and the goal of drinking. Because color is an irrelevant attribute for fulfilling the goal, the domain theory is still able to explain how the object fulfills the goal. On the other hand, the size attribute affects the goal's fulfillment and consequently the domain theory cannot explain how the object is used for drinking since the object description lacks this feature. This comparison reveals that the color attribute should be abstracted out of the object description.

## 6.2 Expanding coverage of human behavior

In the presentation of the model, several inconsistencies with human data were noted. First, the model fails to predicts typicality effects at asymptotic performance. Second, the model does not predict response time effects as a function of inter-category typicality. In the following subsections, I address these two problems. The goal is to establish research directions that resolve these issues without compromising the model's functionality nor its fit within Soar.

Finally, the model does not predict accuracy or response time effects as a function of categorical level (basic-level effect). In Section 5.5, I suggest the discrepancy between the relative rate in which

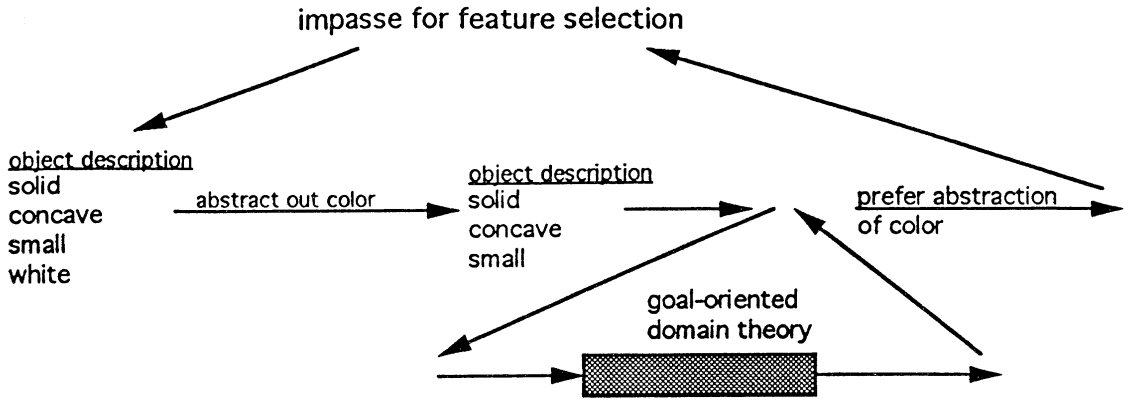


Figure 6.3: Theory–driven feature selection

SCA learns superordinate categories and that in which people learn them could possibly be accounted by the difference in tasks. For the Murphy and Smith (1982) experiments people were performing a category verification task whereas SCA performs a category naming task. By extending SCA to perform category verification, we can test this theory. At the end of this chapter, I propose research directions in which SCA could be extended to perform this task.

### 6.2.1 Asymptotic response time and typicality

As pointed out earlier in the thesis, eventually, given enough repetitions, the prediction rules' conditions will include all of the features found in the instance. At this point, the distinction in response time behavior as a function of typicality is lost. Whether this also occurs in humans has not been thoroughly investigated. However, Murphy and Smith (1982) report differences in asymptotic response times as a function of category level. Presumably, the situation is the same for typicality.

For improving performance, I suggested that the imposition of a learning criterion might have some functional advantages. If such an implementation is functionally feasible, it would leave asymptotic response times intact. Once the learning criterion is reached, the specificity of the system's rules become fixed. Thus less specific rules would still match less typical instances.

### 6.2.2 Inter-category typicality

Another problem with SCA is its inability to explain response time differences as a function of inter-category typicality. As explained earlier, this is because SCA, in searching from the most specific to less specific, stops searching as soon as it finds a prediction; having a slightly less specific rule predicting a contrasting category has no bearing on the response time.

The key to explaining response time differences lies in the varying likelihoods of conflicting predictions as a function of inter-category typicality. An instance which shares many features with instances from a contrasting category is more likely to retrieve conflicting category predictions. However, this observation, by itself, does not account for response time differences. Rather, an account of varying response times must motivate the need for additional computational resources for the low typicality cases.

Investing in additional computational resources for resolving conflicting prediction can be motivated in two ways. First of all, this situation presents the opportunity of refining feature selection knowledge. Secondly, for performance, further search may retrieve a non-conflicting prediction, thus increasing accuracy. Both of these ideas have been incorporated in SCA-2, already presented as a means of improving performance. SCA-2 uses the opportunity of a conflicting prediction to explore better feature orderings. For performance, upon retrieving conflicting predictions, it further probes for a non-conflicting prediction.

Table 6.1: Training and testing data for inter-category typicality

| Category | Attributes D1 | D2 | D3 | D4 | D5 | Overlap Score | Typicality Group |
|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 0 | 13 | Low |
| A | 0 | 0 | 0 | 0 | 1 | 13 | Low |
| A | 0 | 1 | 1 | 1 | 0 | 9 | Mid |
| A | 0 | 1 | 1 | 1 | 1 | 9 | Mid |
| A | 0 | 2 | 2 | 2 | 0 | 5 | High |
| A | 0 | 2 | 2 | 2 | 1 | 5 | High |
| B | 1 | 0 | 0 | 0 | 0 | – | – |
| B | 1 | 0 | 0 | 0 | 1 | – | – |
| B | 1 | 0 | 1 | 1 | 0 | – | – |
| B | 1 | 0 | 1 | 1 | 1 | – | – |
| B | 1 | 2 | 0 | 1 | 0 | – | – |
| B | 1 | 2 | 0 | 1 | 1 | – | – |

Table 6.2: Inter-category typicality effects

| Model | Accuracy Low | Mid | High | Steps Low | Mid | High | Swaps Low | Mid | High |
|---|---|---|---|---|---|---|---|---|---|
| SCA | 89% | 91% | 92% | 2.55 | 2.56 | 2.48 | – | – | – |
| SCA-2 | 72% | 93% | 99% | 1.52 | 1.58 | 1.53 | 12% | 10% | 3.9% |

In order to obtain a preliminary assessment of the impact of varying levels of inter-category typicality on SCA-2's performance, both SCA and SCA-2 were tested on a data-set where all instances had the same intra-category typicality but different levels of inter-category typicality. Table 6.1 describes the data-set. Category A consists of examples of different levels of inter-category typicality. The overlap score is the number of features the instance shares with all of the instances in the contrast category. The overlap score is the inverse of inter-category typicality. Category B serves as the contrast category.

Table 6.2 presents results comparing SCA and SCA-2. Both models are compatible with human behavior for accuracy, where accuracy is better for higher typicality. However, SCA-2 makes a larger distinction between the typicality levels. The average number of steps is also included. This is the number of feature abstractions required before a prediction rule matches. This figure does not count the possible swap. Neither model has any significant differences between the different typicality levels. The swap statistic is the percentage of times an additional probe is made by swapping features. Here there is a significant difference between the category levels. If backtracking takes a significant amount of time, SCA-2 reaction times would be consistent with human behavior.

## 6.3 Extending to other tasks

The supervised learning, as described here, consists of predicting the category name given an object description. Despite the narrowness of the task, supervised learning is a powerful faculty since it can learn to predict any missing feature provided that it was explicitly trained to do so. Nevertheless, humans routinely perform other category-based tasks which utilize knowledge acquired through supervised learning. These include verifying if a instance is classified correctly, expressing the goodness of category membership, producing an example of a category, and verifying taxonomic relationships.

Here, I address these tasks and offer directions on how they may be realized given SCA and

its instantiation in the Soar architecture. For all of these tasks, I suggest how the performance of these tasks arise out of the design structures and representations that already exist within SCA and Soar. As a result, the proposed theories often contrast with more traditional approaches that build explicit declarative data structures for fulfilling the task. While none of the forthcoming theories are specified to the extent that they can be readily implemented, their theoretical content suffices to make some empirically testable predictions.

Finally I address feature selection. Even though I have specified and used an implementation that performs this aspect of the supervised learning task, this implementation does not conform to Soar's constraints. In the last subsection, I discuss how a Soar implementation could be extended to perform data-driven feature selection.

### 6.3.1 Concept verification

I have described SCA in the context of a naming task. That is, for performance, the goal is to name the correct category given an unclassified object description. In contrast, the concept verification task requires the agent to verify the correctness of a classification. Thus, when given an object description and a category, the agent determines whether the object belongs to the category. For example, when presented with the object {`spherical, blue, smooth, medium`} and the category 'globe', the agent must indicate if the classification is correct, perhaps by answering 'true' or 'false'.

How could a SCA-consistent model perform this task? The most straight-forward model would simply perform the naming task and then compare the prediction with the given category. If the prediction and the given category are the same, then the model responds affirmatively, otherwise negatively.

Applying the naming task to the verification task is not the most functionally motivated method. Consider the case where multiple predictions are made. For naming, SCA must arbitrarily choose one of them. For verification, the arbitrarily chosen prediction is compared with the given category, ignoring all predictions that were equally consistent with the object description. A more sophisticated approach would compare the given category to see if it is consistent with *any* of the predictions.

Applying SCA to the verification task using the consistency approach makes a prediction on how humans would perform the task. In particular, it suggests that humans, when given an object description and category, are more likely to affirmatively verify its membership than they would name the same category, given the same object description unclassified. For example, when shown a small spherical object, people are more likely to confirm that it is a ball than they would freely name it as such.

### 6.3.2 Expressing goodness of membership

Not only are typical instances classified faster and with less errors, people tend to consider them good examples. For example, when presented with the object {`spherical, blue, smooth, medium`} and the category 'globe', people can verbally express, perhaps on a scale from 1 to 10, how well the object exemplifies the concept of 'globe', and, if this object is fairly similar to many other objects previously classified as 'globe', they will highly rate this example as a good member of the concept.

Given people's ability to verbally express an example's typicality, the verbal process must have some access to the model's typicality rating. In SCA typicality is the degree of match (*i.e.* the number of features that matched the prediction rule) which is directly available in working memory. Verbal processes can have direct access to this knowledge. The procedure seems trivial. However, as with advice-driven learning, its 'triviality' depends on the model's *a priori* commitment to explicitly controlling which features are matched.

### 6.3.3 Producing an example of a category

When given only a category name, people can describe an example belonging to the category [Rosch, 1978]. For example, when given the category 'globe', people may describe the object {`spherical,`

blue, smooth, medium}, particularly if this object is fairly similar to many other objects previously classified as a globe. The critical question is whether the knowledge acquired by SCA can be applied to producing category examples. The answer is 'yes' but with difficulty. However, with practice, the task becomes easier.

The knowledge acquired by SCA takes the form of prediction rules whose conditions are partial object descriptions and whose actions are category predictions. In the context of Soar, these rules are productions to which problem solving does not have direct access. They must be accessed through a constructive generate and test strategy. For example, let us suppose that the following rules have been acquired by SCA through supervised learning:

1. [oblong] ⇒ball
2. [spherical] ⇒globe
3. [spherical] ⇒ball
4. [spherical, blue] ⇒globe
5. [spherical, red] ⇒ball
6. [spherical, blue, smooth] ⇒globe
7. [red] ⇒ball
8. [red, oblong] ⇒ball

SCA can not examine these rules directly. Instead, it must generate object descriptions and test if any rules match that predict 'globe'. This scheme thus requires knowledge that can be used to construct object descriptions composed from a basic set of primitives. However, the search for an object description need not be random. It can start by guessing only one feature. In this case, feature selection heuristics may suggest starting with shape. It may first guess oblong and cubical, before generating spherical, at which point Rule 2 matches. Next, the strategy may try to add color to the object description. It may guess several colors before generating blue, which causes Rule 4 to match. Eventually, it generates and adds smooth causing Rule 6 to match. It may continue generating additional features and adding it to the object description, or it might backtrack by deleting some features and adding others. Further research would have to address when to backtrack and when the generated object description is sufficiently specific that search should stop.

Despite the use of general rules in guiding the construction process, the generate and test strategy is expensive. Without backtracking, the cost is proportional to the number of possible values for each attribute. If extensive backtracking is used, the problem becomes intractable. Two learning devices could reduce the computational expense. First, the process can learn which attribute values are most common and generate those first. Second, the process of generating object descriptions can be compiled into new rules for future object recall. By generating descriptions within subgoals created by Soar, the architecture can automatically learn new recall rules through its native learning mechanism, chunking. Thus the generate and test process becomes easier with practice.

### 6.3.4 Taxonomic relations between concepts

Given two categories, humans can usually say whether one is a subclass of another. For example, given the categories airplane and jet, most people would be able to say that a jet is a type of airplane. Traditional models of this task organize concepts as explicit hierarchies [Collins and Quillian, 1969; Conrad, 1972]. In this case, the relationship between the concepts of jet and airplane is represented as an explicit directional link indicating that a jet is an airplane.

In developing SCA, no provisions have been made for organizing concepts in explicit taxonomic hierarchies. The claim is that explicit relationships do not necessarily exist by virtue of knowing the concepts. Rather, explicit hierarchies only arise through deliberate problem solving where the task requires explicit connections to be made. Thus, in some circumstances, SCA may competently

process the concepts of jet and airplane in isolation without explicitly representing the former as being a subclass of the latter.

Nevertheless, using knowledge acquired by SCA, Soar could verify taxonomic relationships as humans can do. A simple strategy has Soar creating a typical example from the subclass concept. The example is then checked for membership with the superclass concept. As with learning to produce an example of a category, the task is difficult at first. But with practice, Soar chunks over the process, thereby creating an explicit connection between concepts.

Because SCA does not perforce organize concepts hierarchically, it is not constrained to representing concepts in a tree. Instead, the implicit conceptual structure is a dynamic process depending on task goals and their focusing influence on different features of the object description. This suggests that the role of different concepts for a certain instance is not strictly limited to describing the instance at different levels of abstraction. Different concepts for the one instance can also describe different aspects of the instance even if the concepts do not have any direct taxonomic relationship.

In returning to the airplane example, SCA suggests that the concept of an airplane is not simply an abstract description of a jet. Rather, both the concepts of jet and airplane place a different focus on a instance's representation. An airplane focuses on the vehicle's ability to fly and carry passengers, whereas a jet primarily focuses on the vehicle's speed and its role with commercial airlines. The context provides the focus and thus the appropriate feature selection. With different feature selection orderings that depend on the context, SCA can consistently and flexibly represent both concepts.

Another example is the so-called ambiguous classification of a tomato as either a fruit or a vegetable. Citations of this example have been used to motivate the use of probabilities in order to capture the different ways in which the concept lies within a taxonomic hierarchy [Anderson and Murphy, 1986]. Rather than having a tomato being probabilistically categorized as either a fruit or a vegetables, the SCA framework implies that the taxonomic relationship is context-dependent. Feature selection is a deliberate process and can thus depend on the context. The botanical context emphasizes features such as seeds, their location, and the type of plant from which it comes. In this context, the tomato is unambiguously a fruit. In the context of food, features such as taste and nutrition are emphasized. Here, the tomato is clearly a vegetable. Models that impose a hierarchical structure have difficulty with these kinds of relations and thus must resort to probabilistic representations. The SCA framework dynamically constructs the taxonomic relation, affording the flexibility of context-dependence. As a consequence, SCA predicts that people should have little difficulty verifying similar relations whenever they are strongly situated within a disambiguating context.

### 6.3.5 Feature selection

The two data-driven feature selection learning strategies with which SCA was evaluated required the continuous updating of conflicting prediction averages indexed by attribute name. These types of learning strategies are not compatible with Soar, being a fully symbolic architecture with no direct access to long-term memory.

One solution is to keep feature learning knowledge in working memory instead of trying to continually update the knowledge in long-term memory. This approach would predict disastrous consequences if the learning process was abruptly interrupted—here the feature selection knowledge would be lost, thus requiring a trial and error process of recovering the knowledge by probing prediction rules in long-term memory. This approach also fails to address how people learn categories whose training instances are presented intermittently over a long period of time.

Perhaps an alternate is SCA-2. SCA-2 uses a fully symbolic feature ordering which is only modified approximately ten percent of the time (see swaps in Table 6.2). Critical remaining issues concern how the feature ordering is represented in Soar's long-term memory and how the ordering knowledge is modified.

# Chapter 7

# CONCLUSION

There are no perfect models. At best one can only approximate the process that underlies a set of natural phenomena. To the extent that the model is a good approximation, it provides insight into the natural processes' structural properties, their relationship in a larger system, and their behavioral consequences. The model also serves as a foothold for future research. Improvements to the model emerge out of additions and modifications to the existing version. Model development is thus an evolutionary path, eventually leading to better approximations of natural processes.

I have described SCA, a model for human category learning. Amidst the implementational details, SCA is distinguished by a particular handful of structural properties, listed in Figure 7.1. These are essential properties on which the successful evaluation of SCA depends. In a sense, these lay out a class of models, of which I have constructed and evaluated one implementation. More sophisticated models in this class probably exist, possibly including models which can represent structured object representations instead of the flat feature structures described in this dissertation. Thus the essential properties depict a set of least commitments constraining the construction of future models.

With SCA's essential properties, I have shown how they extend from the constraints of a unified theory (Figure 7.2), how they contribute to the model's functionality (Figure 7.3), and how they yield behavior consistent with a diverse set of human phenomena (Figure 7.4—the last two are novel predictions). Furthermore, in Chapter 6, I have laid out directions for extending additional capabilities within the UTC, for increasing the model's performance, and for expanding the model's coverage of documented human phenomena.

- **Distributed, rule-based representation.** A concept is represented by a distributed set of rules (productions). *Section 3.1.1.*
- **Full symbolic matching of rules with parallel retrieval.** Rules fully matching the internal symbolic object description are retrieved in parallel, within constant time. *Section 3.1.2.*
- **Serial search.** Rules that do not fully match the internal representation are retrieved with a serial search. *Section 3.1.2.*
- **Search from specific to general.** The serial search pattern tries to retrieve specific rules first. *Section 3.1.2.*
- **Deliberate feature selection controlling search.** All relevant knowledge is potentially brought to bear in guiding the serial search. *Sections 3.4 and 4.3.*
- **Learning specific rules from general rules.** New, more specific rules are acquired from applying more general rules. *Section 3.1.3.*

Figure 7.1: SCA's essential structural properties

- Distributed, rule-based knowledge.
- Symbolic retrieval (full-matching).
- Serial application of discrete deliberate operations.
- Non-examinable rules.
- Learning determined by performance in application of prior knowledge (explanation-based learning).
- Monotonic learning.

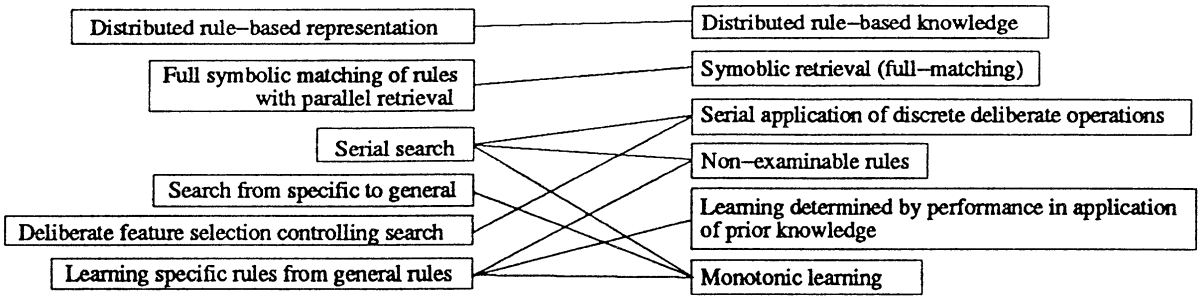


Figure 7.2: Architectural constraints and their dependencies. *Section 3.4.*

Constructing the model out of mechanisms posited by a unified theory of cognition is also a major contribution of the thesis. Not only has the UTC been useful in constructing SCA, but this methodology also contributes to our understanding of Soar as a UTC. SCA provides a window as to how Soar's mechanisms, operating at the order of 10 milliseconds, can be composed to explain behavior operating on the order of 1 second. For example, we have seen how constant-time fully symbolic rules can be applied in producing flexible, noise-tolerant behavior with varied error rates and response times conforming to human behavior.

By keeping SCA's essential properties, improvements can be sought by experimenting with alternate implementations. These properties serve as a set of constraints for the future construction of models. As long the main properties remain intact, modifications will not adversely affect our accomplishments. Also the structural properties need not be lumped into one monolithic set. Certain properties or combinations of properties have particular consequences or fit particular constraints (see dependencies in Figures 7.2, 7.3, and 7.4). For example, as depicted in Figure 7.4, response time effects are consequences of a serial search, the order of the search, and the order in which rules are learned. By understanding these dependencies, we can predict the consequences when modifying key properties. When constructing a new model with some new properties, we know that, for the remaining previous properties, the accomplishments that depended on them will be preserved in the new model. Thus we can usefully bias the search for newer and better models.

- **Concept expressiveness.** SCA can represent a diverse set of concept structures including non-linear categories and family resemblance categories. *Sections 4.4, 5.2, and 5.7.*
- **Incremental learning.** The time required to process a training instance has a constant bound with respect to number of previous encountered training examples. *Definition 1 and Sections 3.1.1 and 5.1.*
- **Real time response.** SCA reliably responds within seconds under the timing constraints of Soar's architectural mechanisms. *Sections 1.2, 3.1.1, and 5.1.*
- **Noise tolerance.** SCA's performance gracefully degrades in the presense of noise. *Section 4.4.*
- **Bias from goals, beliefs and theories.** SCA allows the possibility for goals, beliefs and theories in guiding object classification. *Sections 5.4 and 6.1.4.*

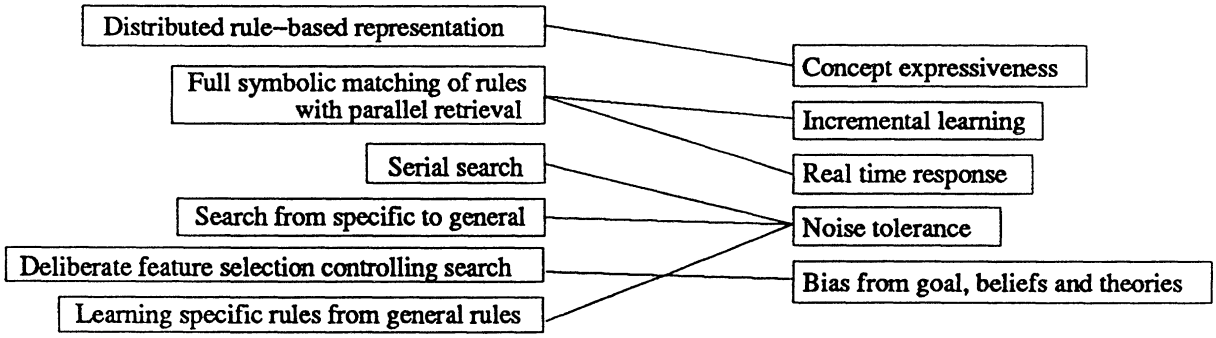


Figure 7.3: Functional goals and their dependencies.

- **Gating bias.** SCA learns categories defined by a few features (*e.g.* Type I) faster than categories defined by several features (*e.g.* Type II or Type VI categories). *Section 5.3.*
- **Typicality effect (rt).** SCA responds faster for objects which are more similar to other objects in the same category. *Section 5.1.2.*
- **Typicality effect (accuracy).** SCA predicts more accurately for objects which are more similar to other objects in the same category. *Section 5.1.2.*
- **Practice effect (rt).** With practice, the time to respond decreases. *Section 5.1.1.*
- **Practice effect (accuracy).** With practice (even with the same training examples), accuracy improves. *Section 5.6.*
- **Learning non-linear categories.** SCA learns some non-linear category structures as fast as linear categories. *Section 5.2.*
- **Effect of strategies.** SCA predicts how some kinds of strategic advice effects prediction performance. *Section 5.4.*
- **Initial guessing of Type VI categories.** For category structures defined by multiple features, SCA requires several training cycles before prediction performance is better than chance. *Section 5.3.*
- **Training response time.** Response time for training on an instance correlates with the response time for predicting with the instance. *Section 5.1.3.*

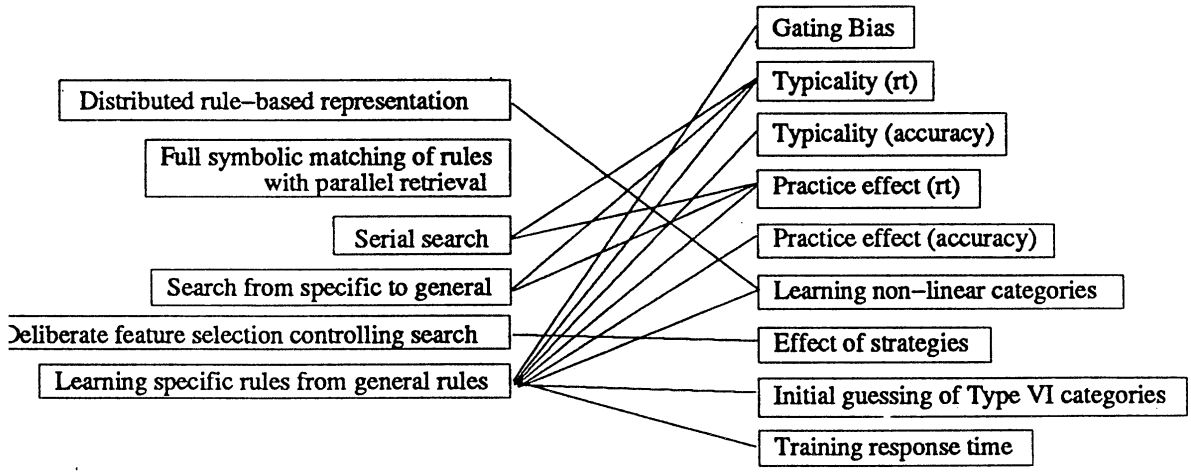


Figure 7.4: Behavioral properties and their dependencies

# Appendix A

# SCA Implementation in Soar

In Chapter 3, I first discussed SCA independent of the Soar architecture. I then motivated the design of SCA in terms of Soar by describing the major architectural constraints and how SCA conformed to them. My presentation strategy had the advantage of being able to construct principle abstract characteristics of the model, so that I could more easily describe their architectural relationship and their functional and behavioral consequences. However, in so doing, I have neglected several technical details required for implementing SCA within the problem solving and learning mechanisms supported by Soar. While some of these details are independent of SCA's principle structural properties, they are necessary in achieving a working implementation in Soar. Here I present some additional details of one working implementation of SCA in Soar [Miller and Laird, 1991].

Problem solving in Soar [Laird *et al.*, 1987] consists of the sequential selection and application of operators to a representational state within a problem space. Both the selection and application of operators is determined by the knowledge represented in a long-term recognition memory encoded as productions. Learning involves the construction of new productions, called chunks, which summarize the results of a subgoal. A chunk in Soar is similar to an operationalized result in explanation-based learning [Mitchell *et al.*, 1986; Dejong and Mooney, 1986]. Subgoals are born out of impasses caused by conflicting knowledge or a lack of knowledge. Subgoaling relies on additional knowledge to consider the alternatives, often requiring further search. During subgoaling, the system may try out the alternatives or it may recast the problem. Once the subgoal resolves the problem that caused the impasse, the architecture traces back through the conditions that led to the impasse's resolution. Using these conditions, a new production is created that summarizes the subgoal's result so that, in analogous applications, the summarizing information is retrieved. This avoids the impasse and thus the costly search encountered when the chunk was first created.

In the above learning scenario, no new knowledge is actually acquired. Instead, the knowledge in the problem space has been partially operationalized in that an increase in efficiency has occurred. In contrast, new knowledge can be acquired only through learning by instructions or by inducing knowledge from examples observed in the external environment. This latter type of learning is often called knowledge-level learning [Dietterich, 1986]. Previous work with Soar has made knowledge-level learning technically feasible [Rosenbloom *et al.*, 1988]. This approach, called data-chunking, is a precursor to the work presented here.

The design of SCA is the direct result of applying Soar to the concept prediction task. Here Soar's explicit goal is to predict the category name of the incoming object description. Category prediction rules serve as task knowledge which may be immediately applicable in fulfilling the task's goal. When no such knowledge is immediately applicable, an impasse results. In this case, subgoaling relies on additional knowledge by modifying the object description so that more general rules may apply. This is accomplished by selecting and then applying an operator that removes a feature from the object description. Selecting an operator is a deliberate act, *i.e.* all applicable knowledge (represented as productions) are brought to bear in the operator's selection. If this knowledge is insufficient or inconsistent, an impasse will result setting up the subgoal of resolving the selection tie or conflict. Further deliberation may resolve the impasse which may include look-ahead search,

retrieving strategic advice, and retrieving relevant domain knowledge. If all else fails, an operator can always be selected at random.

Once an operator is selected and applied by modifying the object description, a prediction rule may apply, thereby fulfilling the task goal. If not, impasses and subgoaling continues until a prediction rule does apply. Chunking, the experienced-based learning mechanism, automatically learns by summarizing the results from resolving the impasse. In this case, it learns a more specific rule based upon the conditions of the rule that matched and the feature that was last removed from the object description.

One critical technical detail is the means in which more general rules are inhibited until the object description only includes features in the rule's conditions. For example, let us consider the situation where the current object description has the representation `[blue, spherical, small]` and one of the prediction rules has the condition `[blue]`. Normally, a Soar production fires whenever its conditions fully match any part of the representation in working memory. Thus the more general production would fire before the other two features are deliberated removed from the object description. There are several ways to inhibit these productions from firing prematurely, without making any changes to the architecture. One simple approach requires representing object descriptions as a linked list of features. Provided that features are listed in a canonical order and that the last feature is marked as being the last in the list, the only production that fires will be the one whose conditions include all of the features in the list. More general productions will not fire until the appropriate features are first deliberately removed from the linked list. Other possible ways of implementing this kind of restricted match include placing explicit negations in productions' conditions, explicitly marking features as 'irrelevant' and then matching the irrelevant markings, and constructing new symbols representing combinations of features. A remaining challenge is to develop restricted match technique for more complex object representations.

The above set of implementational details have led to one successful implementation of SCA. Others probably exist. Possible deviations include different uses of goals and impasses and alternate implementations of restricting production matching. A future research issue involves understanding and implementing alternate methods which could incorporate more sophisticated object representations. As long as a new method adhered to SCA's general properties, the resulting model would still possess the same functional and behavioral properties I have identified in this thesis.

# Appendix B

# Formal specification of a concept

This appendix provides a formal definition of a concept as implemented by the SCA model. This formal specification is not a process model in the sense that SCA is a "strong" cognitive model, and thus does not make process-dependent predictions such as typicality effects in terms of response time.

Here, a concept is defined in terms of a set of *prediction assertions* and a *context*. The constructs have intuitive mappings to SCA. A *prediction assertion* described here functions as a prediction rule in SCA. A *context* maps to feature selection since it constrains which prediction assertions are applicable to the object description.

## B.1 Object descriptions

Object descriptions consist of attributes predicated by values. We denote a list of $m$ attributes:

$$A_1, A_2, \ldots, A_m$$

Each attribute Ax can be predicated by one of $n_x$ values denoted:

$$V_{x,1}, V_{x,2}, \ldots, V_{x,n_x}$$

Object descriptions consist of a subset of attributes predicated by values where $o <= m$ and $i_x \neq i_y$:

$$A_{i_1}(V_{i_1,j_1}), A_{i_2}(V_{i_2,j_2}), \ldots, A_{i_o}(V_{i_o,j_o})$$

Since the name of the value implies the attribute, attributes with specified values can simply be noted as:

$$V_{i_1,j_1}, V_{i_2,j_2}, \ldots, V_{i_o,j_o}$$

Typically, in our examples, we list attributes and values by their names instead of indexing them. An example list of attributes may be *texture, color, shape, rigidity* and *size*. An example list of possible values for *texture* may include *fuzzy, smooth, rough, pebbled*. Here's an example object description:

```
Texture(fuzzy), Color(red), Shape(spherical), Size(small)
```

or simply:

```
fuzzy, red, spherical, small
```

## B.2 Concept definition

A concept definition is defined by a *context* and a set of *prediction assertions.* Given an object description, the concept definition produces the category of the object.

### B.2.1 Prediction assertions

Prediction assertions consist of a condition-prediction pairing of the form:

```
<condition; prediction>
```

The condition part is identical to the object description construction. The prediction part is a concept label. The following are several examples of prediction assertions:

```
<smooth, red, spherical; ball>
<fuzzy, blue; ball>
<smooth, red; ball>
<smooth, blue; globe>
<smooth, blue, large; globe>
<smooth, blue, large, spherical; ball>
```

**Definition 2** *Prediction assertion p is* **consistent** *with object description d if for every value v in the condition of p (written* $v \in_p Cond(p)$*), v is also in d.*

For example,

```
<smooth, blue; globe>
<smooth, blue, large; globe>
<smooth, blue, large, spherical; ball>
<smooth, soft; sponge>
```

are all consistent with

```
smooth, blue, large, spherical, soft
```

**Definition 3** *The* **consistency set** *of a set of prediction assertions P and an object description d is the set of all the prediction assertions in P that are consistent with d.*

**Definition 4** *For prediction assertions p and q,* $Cond(p) \subseteq_p Cond(q)$ *if, for every d with which q is consistent, p is also consistent. Also,* $Cond(p) \subset_p$ *if* $Cond(p) \subseteq_p Cond(q)$ *and* $Cond(p) \neq Cond(q)$.

Examples:

```
p = <smooth, blue; globe>
q = <smooth, blue, large, spherical; globe>
s = <smooth, blue, large, spherical; ball>

Cond(p) ⊆p Cond(q)
Cond(q) ⊆p Cond(s)
```

Note that the relation, $\subseteq_p$, imposes a partial order on prediction assertion conditions.

**Definition 5** *Given the consistency set Q taken from a set of prediction assertions P and an object description d, the* **prediction set** *of P and d, denoted* $\Pi(P, d)$*, is the set*

$$\{p | p \in Q \wedge \neg\exists[x \in Q] Cond(p) \subset_p Cond(x)\}$$

Intuitively, the prediction set is the most specific set of prediction assertions from P that are consistent with d. For example, for the following prediction assertions and object description,

```
<smooth, red, spherical; ball>
<fuzzy, blue; ball>
<smooth, red; ball>
<smooth, blue; globe>
<smooth, large; globe>
<smooth, large, spherical; ball>
<smooth, soft; sponge>

smooth, blue, large, spherical, soft
```

the resulting prediction set is

```
<smooth, large, spherical; ball>
<smooth, soft; sponge>
```

### B.2.2 Contexts

Contexts are used to reduce the prediction set to a smaller size, often to one unique prediction.

**Definition 6** *Given a set of attributes A, a context C is a total order, $\leq_c$, on A.*

Example:

$\texttt{texture} \leq_c \texttt{shape} \leq_c \texttt{size} \leq_c \texttt{rigidity} \leq_c \texttt{color}$

**Definition 7** *Given an object description d, a set of prediction assertions P, and a context C, the* **defining assertion(s)**, *denoted Def(d,P,C), is*

$$\Pi(P, d) \cap \{p | \forall [v, w \in_p \wedge v \in_p d \wedge w \notin_p d] v \leq_c w\}$$

The context thus constrains which attributes are used in reducing the size of the prediction set.

**Definition 8** *If there is only one defining assertion, the* **concept** *for d, as defined by P and C, is that assertion's prediction. If there is more than one defining assertion, the* **concept** *is undefined.*[1]

The defining assertion using the examples from Definition 5 is

```
<smooth, large, spherical; ball>
```

Thus, the concept is `ball`.

[1] For computer simulations emulating a forced-choice task, the model must randomly choose one of the defining assertions.

# Bibliography


[Aha *et al.*, 1991] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[Aha, 1989] D. W. Aha. Incremental, instance-based learning of independent and graded concept descriptions. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 387–391, 1989.

[Ahn and Medin, 1992] Woo-Kyoung Ahn and D. L. Medin. A two-stage model of category construction. *Cognitive Science*, 16:81–121, 1992.

[Aho *et al.*, 1974] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

[Anderson and Murphy, 1986] J. A. Anderson and G. L. Murphy. Psychological concepts in a parallel system. *Physica*, pages 318–336, 1986.

[Anderson *et al.*, 1979] J. R. Anderson, P. J. Kline, and C. M. Beasley, Jr. A general learning theory and its application to schema abstraction. *The Psychology of Learning and Motivation*, 13:277–318, 1979.

[Anderson, 1983] J. R. Anderson. *The Architecture of Cognition.* Harvard University Press, Cambridge, Massachusetts, 1983.

[Anderson, 1987] J. R. Anderson. Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review*, 94:192–210, 1987.

[Anderson, 1991] J. R. Anderson. The adaptive nature of human categorization. *Psychological Review*, 98:409–429, 1991.

[Bergadano *et al.*, 1992] F. Bergadano, S. Matwin, R. S. Michalski, and J. Zhang. Learning two-tiered descriptions of flexible concepts: The POSEIDON System. *Machine Learning*, 8:5–43, 1992.

[Billman and Heit, 1988] D. Billman and E. Heit. Observational learning from internal feedback: A simulation of an adaptive learning method. *Cognitive Science*, 12:587–625, 1988.

[Blumer *et al.*, 1987] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 23:377–380, 1987.

[Booker *et al.*, 1989] L.B. Booker, D.E. Goldberg, and J.H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

[Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Monterey, CA, 1984.

[Carey, 1982] S. Carey. Semantic development: the state of the art. In E. Wanner and L. R. Gleitman, editors, *Language Acquisition: The State of the Art*, pages 347–389. Cambridge University Press, New York, 1982.

[Cheeseman *et al.*, 1988] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AutoClass: A baysesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, 1988.

[Clark, 1973] E. V. Clark. What's in a word? on the child's acquisition of semantics in his first language. In T. Moore, editor, *Cognitive development and the acquisition of language*. Academic Press, New York, 1973.

[Collins and Quillian, 1969] A. M. Collins and M. R. Quillian. Retrieval time for semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8:240–247, 1969.

[Conrad, 1972] C. Conrad. Cognitive economy in semantic memory. *Journal of Experimental Psychology*, 92:149–154, 1972.

[Dejong and Mooney, 1986] G. Dejong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176, 1986.

[Dietterich, 1986] T. G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1:287–316, 1986.

[Doorenbos *et al.*, 1992] R. Doorenbos, M. Tambe, and A. Newell. Learning 10,000 chunks: what's it like out there. In *Proceedings of the National Conference on Artificial Intelligence*, 1992.

[Dromi, 1987] E. Dromi. *Early Lexical Development.* Cambridge University Press, New York, 1987.

[Elio and Anderson, 1984] R. Elio and J. R. Anderson. The effects of information order and learning mode on schema abstraction. *Memory and Cognition*, 12:20–30, 1984.

[Fayyad, 1991] U. M. Fayyad. *On the induction of decision trees for multiple concept learning.* PhD thesis, The University of Michigan, 1991.

[Feigenbaum and Simon, 1984] E. A. Feigenbaum and H. A. Simon. Epam-like models of recognition and learning. *Cognitive Science*, 8:305–336, 1984.

[Fisher and KcKusick, 1989] D. H. Fisher and K. B. KcKusick. An empirical comparison of ID3 and Back-propagation. In *Eleventh International Joint Conference on Artificial Intelligence*, pages 788–793, 1989.

[Fisher and Langley, 1990] D. Fisher and P. Langley. The structure and formation of natural categories. Technical Report RIA-90-02-15-1, NASA Ames Research Center, 1990.

[Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[Fisher, 1988] D. H. Fisher. A computational account of basic level and typicality effects. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 233–238, 1988.

[Fisher, 1989] D. H. Fisher. Noise-tolerant conceptual clustering. In *Eleventh International Joint Conference on Artificial Intelligence*, pages 825–830, 1989.

[Forgy, 1982] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[Gluck and Bower, 1988a] M. A. Gluck and G. H. Bower. Evaluating an adaptive network model of human learning. *Journal of Memory and Language*, 27:166–195, 1988.

[Gluck and Bower, 1988b] M. A. Gluck and G. H. Bower. From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General*, 117:227–247, 1988.

[Hanson and Bauer, 1989] S. J. Hanson and M. Bauer. Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3:343–372, 1989.

[Haussler, 1988] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36:177–221, 1988.

[Hintzman, 1986] D. L. Hintzman. Schema abstraction in a multiple-trace memory model. *Psychological Review*, 93:411–428, 1986.

[Holland and Reitman, 1978] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-directed Inference Systems*, pages 313–329. Academic Press, New York, 1978.

[Huffman *et al.*, 1993] S.B. Huffman, C.S. Miller, and J.E. Laird. Learning from instruction: A knowledge-level capability within a unifed theory of cognition. In *Proceedings of the 15th Annual Meeting of the Cognitive Science Society*, 1993.

[John, 1988] B. E. John. *Contributions to Engineering Models of Human-Computer Interaction.* PhD thesis, Carnegie-Mellon University, 1988.

[Kohonen, 1982] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Kolodner, 1983] J. L. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:243–280, 1983.

[Kruschke, 1990] J. K. Kruschke. ALCOVE: A connectionist model of category learning. Research Report 47, Cognitive Science Program, Indiana University, 1990.

[Kruschke, 1991] J. K. Kruschke. Dimensional attention learning in models of human categorization. In *The 13th Annual Conference of the Cognitive Science Society*, 1991.

[Laird *et al.*, 1987] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[Lakoff, 1987] G. Lakoff. *Women, Fire and Dangerous Things.* The University of Chicago Press, Chicago, 1987.

[Lebowitz, 1987] M. Lebowitz. Experiments with incremental concept formation: Unimem. *Machine Learning*, 2:103–138, 1987.

[Lewis *et al.*, 1990] R. L. Lewis, S. B. Huffman, B. E. John, J. E. Laird, J. F. Lehman, A. Newell, P. S. Rosenbloom, T. Simon, and S. G. Tessler. Soar as a unified theory of cognition: Spring 1990. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, pages 1035–1042, Cambridge, MA, 1990.

[Martin and Billman, 1991] J. D. Martin and D. O. Billman. Variability bias and category learning. In *Machine Learning: Proceedings of the Eighth Internation Workshop*, pages 90–94, 1991.

[Medin and Schaffer, 1978] D. L. Medin and M. M. Schaffer. Context theory of classification learning. *Psychological Review*, 85:207–238, 1978.

[Medin and Schwanenflugel, 1981] D. L. Medin and P. J. Schwanenflugel. Linear separability in classification learning. *Journal of Experimental Psychology: Human Learning and Memory*, 7:355–368, 1981.

[Medin and Smith, 1981] D. L. Medin and E. E. Smith. Strategies and classification learning. *Journal of Experimental Psychology: Human Learning and Memory*, 7:241–253, 1981.

[Michalski and Stepp, 1983] R. S. Michalski and R. E. Stepp. Learning from observation: conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 331–363. Morgan Kaufmann, Los Altos, CA, 1983.

[Michalski, 1983] R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.

[Miller and Laird, 1991] C. S. Miller and J. E. Laird. A constraint-motivated model of concept formation. In *The 13th Annual Conference of the Cognitive Science Society*, pages 827–831, 1991.

[Miller and Laird, 1992] C. S. Miller and J. E. Laird. A simple symbolic algorithm for incremental concept acquisition. Technical Report CSE-TR-153-93, Department of Electrical Engineering and Computer Science, The University of Michigan, 1992.

[Minton, 1988] S. Minton. Quantitative results concerning the utility of explanation-based learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 564–569, 1988.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[Mitchell, 1990] T. M. Mitchell. The need for biases in learning generalizations. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning*, pages 184–190. Morgan Kaufmann, San Mateo, CA, 1990.

[Mooney *et al.*, 1989] R. Mooney, J. Shavlik, G. Towell, and A. Gove. An experimental comparison of symbolic and connectionist learning algorithms. In *Eleventh International Joint Conference on Artificial Intelligence*, pages 775–780, 1989.

[Murphy and Medin, 1985] G. L. Murphy and D. L. Medin. The role of theories in conceptual coherence. *Psychological Review*, 92:289–316, 1985.

[Murphy and Smith, 1982] G. L. Murphy and E. E. Smith. Basic-level superiority in picture categorization. *Journal of Verbal Learning and Verbal Behavior*, 21:1–20, 1982.

[Newell, 1990] A. Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA, 1990.

[Nosofsky and Gluck, 1989] R. M. Nosofsky and M. A. Gluck. Adaptive networks, exemplars, and classification rule learning. Paper presented at the 30th annual meeting of the Psychonomic Society, Atlanta, 1989.

[Paxton, 1990] J. T. Paxton. *Martian: A Concept Learning System.* PhD thesis, The University of Michigan, 1990.

[Quinlan, 1986] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Ratcliff and Murdock Jr., 1976] R. Ratcliff and B. B. Murdock Jr. Retrieval processes in recognitional memory. *Psychological Review*, 83:190–214, 1976.

[Ratcliff, 1978] R. Ratcliff. A theory of memory retrieval. *Psychological Review*, 85:59–108, 1978.

[Reich, 1976] P. A. Reich. The early acquisition of word meaning. *Journal of Child Language*, 3:117–123, 1976.

[Rips, 1989] L. J. Rips. Similarity, typicality and categorization. In S. Vosniadou and A. Ortony, editors, *Similarity and analogical reasoning*, pages 21–59. Cambridge University Press, Cambridge, 1989.

[Rosch and Mervis, 1975] E. Rosch and C. B. Mervis. Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7:573–605, 1975.

[Rosch *et al.*, 1976a] E. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8:382–439, 1976.

[Rosch *et al.*, 1976b] E. Rosch, C. Simpson, and R. S. Miller. Structural bases of typicality effects. *Journal of Experimental Psychology: Human Perception and Performance*, 2:491–502, 1976.

[Rosch, 1978] E. Rosch. Principles of categorization. In E. Rosch and B. B. Lloyd, editors, *Cognition and Categorization*, pages 27–48. Erlbaum, Hillsdale, NJ, 1978.

[Rosenblatt, 1962] F. Rosenblatt. *Principles of Neurodynamics.* New York, Sparten, 1962.

[Rosenbloom and Aasman, 1990] P. S. Rosenbloom and J. Aasman. Knowledge level and inductive uses of chunking (EBL). In *Proceedings, Eighth National Conference on Artificial Intelligence*, pages 821–827, 1990.

[Rosenbloom *et al.*, 1988] P. S. Rosenbloom, J. E. Laird, and A. Newell. The chunking of skill and knowledge. In B. A. G. Elsendoorn and H. Bouma, editors, *Working Models of Human Perception.* Academic Press, London, 1988.

[Rumelhart *et al.*, 1986a] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland. A general framework for parallel distributed processing. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1*, pages 318–362. The MIT Press, Cambridge, MA, 1986.

[Rumelhart *et al.*, 1986b] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1*, pages 318–362. The MIT Press, Cambridge, MA, 1986.

[Schank *et al.*, 1986] R. C. Schank, G. C. Collins, and L. E. Hunter. Transcending inductive category formation in learning. *Behavioral and Brain Sciences*, 9:639–686, 1986.

[Schlimmer and Fisher, 1986] J. C. Schlimmer and D. Fisher. A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 115–121, 1986.

[Schlimmer, 1987] J. C. Schlimmer. Incremental adjustment of representations for learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 79–90, 1987.

[Schyns, 1991] P. G. Schyns. A modular neural network model of concept acquisition. *Cognitive Science*, 15:461–508, 1991.

[Shavlik *et al.*, 1991] J. W. Shavlik, R. J. Mooney, and G. G. Towell. Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111–143, 1991.

[Shepard *et al.*, 1961] R. N. Shepard, C. I. Hovland, and H. M. Jenkins. Learning and memorization of classifications. *Psychological Monographs: General and Applied*, 75(13):1–41, 1961.

[Shortliffe, 1976] E. H. Shortliffe. *Computer-based medical consultations: MYCIN.* American Elsevier, New York, 1976.

[Simon and Feigenbaum, 1964] H. A. Simon and E. A. Feigenbaum. An information processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning. *Journal of Verbal Learning and Verbal Behavior*, 3:385–396, 1964.

[Tambe and Newell, 1988] M. Tambe and A. Newell. Some chunks are expensive. In *Proceedings of the Fifth International Conference on Machine Learning*, 1988.

[Utgoff, 1988] P. E. Utgoff. ID5: An incremental ID3. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 107–120, 1988.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[Vapnik, 1982] V. N. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer, New York, 1982.

[Wiesmeyer and Laird, 1990] M. Wiesmeyer and J. Laird. A computer model of 2d visual attention. In *The Twelfth Annual Conference of the Cognitive Science Society*, pages 582–589, 1990.

[Zadeh, 1965] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.