COGNITIVE SCIENCE **20**, 499–537 (1996)

Accounting for Graded Performance within a Discrete Search Framework

CRAIG S. MILLER
Carnegie Mellon University

JOHN E. LAIRD
The University of Michigan

This article presents a process account of some typicality effects and related similarity-dependent accuracy and response time phenomena that arise in the context of supervised concept acquisition. We describe Symbolic Concept Acquisition (SCA), a computational system that acquires and activates category prediction rules. In contrast to gradient representations, SCA performs by probing for prediction rules in a series of discrete steps. For learning new rules, it acquires general rules but then incrementally learns more specific ones. In describing SCA, we emphasize its functionality in terms of accuracy and efficiency and motivate its design within the set of symbolic mechanisms and memory structures defined by the Soar architecture (Laird, Newell, & Rosenbloom, 1987). For replicating human behavior, we first show how SCA exhibits some typicality effects in the course of learning responding faster and more accurately to more typical test examples. Then, using data from human experiments, we evaluate SCA's qualitative predictions on accuracy and response time on individual dataset instances. We show how SCA's predictions correlate with human data across three experimental conditions concerning the effect of instruction on learning strategy.

1. INTRODUCTION

Within the last 15 years, research in concept and memory organization has moved away from symbolic representations, such as discrimination nets (Feigenbaum & Simon, 1984) and logical descriptions (Michalski, 1983), and towards gradient, probabilistic representations, such as neural nets (Gluck & Bower, 1988; Kruschke, 1992; Rumelhart, Hinton, & Williams,

The research was sponsored in part by NASA Ames contract NCC2-517 at the University of Michigan and by the Markle Foundation at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the Markle Foundation. We thank Dorrit Billman, Paul Rosenbloom, James Greeno, Jill Lehman, Randy Jones, Roger Remington, and an anonymous reviewer for providing many valuable comments on earlier versions of this paper. In addition, we are indepted to Doug Medin and Mike Pazzani for their suggestions stemming from previous presentations of this work.

Correspondence and requests for reprints should be sent to Craig Miller, Department of Mathematics and Computer Science, Dickinson College, Carlisle, PA 17013. E-mail: <miller cr@dickinson.edu>

1986) and probabilistic declarative structures (Anderson, 1991; Fisher, 1988). In part, this is a response to empirical evidence suggesting that category membership lies on a continuum as manifested by such metrics as human response times and accuracy rates. Gradient models provide an immediate answer to the source of these phenomena—the behavior is a direct reflection of internal representations of varying degrees of category membership. From a functional viewpoint, these models have also been touted for their flexibility in handling noisy, incomplete data—a functionality clearly demanded of humans. In short, gradient models seem to address these empirical and functional demands, and have subsequently become the leading candidates for modeling human memory.

Despite the attractiveness of gradient models of human memory, we believe the dismissal of discrete, symbolic models would be premature without a more thorough exploration of their capability. In this article, we intend to advance our understanding of their potential capability by presenting a discrete rule-based model and by demonstrating how it tackles some of the empirical and functional demands addressed by gradient models.

In presenting our approach, we will motivate its underlying process in terms of functionality and its grounding within a pre-established architecture. We will first describe our models's process as a means of learning and activating category prediction rules efficiently and accurately. Then, we show how this process follows from a pre-established set of symbolic mechanisms and memory structures, namely those defined by the Soar architecture (Laird, Newell, & Rosenbloom, 1987; Newell, 1990), a candidate Unified Theory of Cognition (UTC).

Following the functional and architectural motivation, we evaluate our model by comparing its behavior to that of humans. We focus on phenomena that are seemingly problematic for discrete rule-based architectures and that have thus motivated gradient representations. In particular, we address accuracy and response time as a function of typicality, defined, in part, by a category member's similarity to other category members. By addressing typicality effects and related similarity-dependent phenomena, we intend to show how a discrete rule-based model might account for gradient category membership. We then investigate behavioral properties that may in turn be problematic for architecturally-fixed gradient representations. In particular, we address how strategic advice impacts the classification process and the extent to which it alters error rate and response time profiles.

2. THE SUPERVISED LEARNING TASK

Our model performs a supervised learning task. The system is presented with training examples,[1] described in terms of attributes and symbolic

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

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}

As training examples, 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 (Breiman, Friedman, Olshen, & Stone, 1984; Quinlan, 1986; Schlimmer & Fisher, 1986; Utgoff, 1988), logical concept descriptions (Michalski, 1983), artifical neural nets (Rosenblatt, 1962; Rumelhart et al., 1986), genetic classifiers (Holland & Reitman, 1978), and stored instances (Aha, Kibler, & Albert, 1991). Likewise, work in psychology has produced models based on similar representations including discrimination nets (Feigenbaum & Simon, 1984), rules (Anderson, Kline, & Beasley, 1979), neural nets (Gluck & Bower, 1988; Kruschke, 1992) and instances (Hintzman, 1986; Medin & Schaffer, 1978).

We 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 constaint: not only do humans learn incrementally on this task, they learn incrementally throughout their lives.

Technically any learning system can be trivally 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 knowledge whenever new training examples are presented. In order to exclude approaches that require excessive computational resources every time additional training examples are provided, we 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, that is, O(1), with respect to the total number of training examples already encountered.*

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 proce-

dure upon the introduction of every new training example. We emphasize 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 continually grows with the amount of experience.

This is an important functional constraint on behavior since it recognizes the need to minimize computational resources 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 processing training examples. Similarly, the task 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 we 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 serving as a training example must be separated from the background of the total environment. For our 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.

Nevertheless, even in the context of these simplifications, comprehensively modeling the task phenomena remains a difficult endeavor, and, clearly, humans can and do perform these tasks. Many psychological experiments make the same representational assumptions, and 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.

3. DESCRIPTION OF THE MODEL

Our learning model has two parts. The first part is the basic mechanism that accesses and learns prediction rules. Prediction rules test for specific attributes and values in examples, and when one rule matches an example, it

predicts a category. The second part learns heuristics, which guide the selection of attributes and values 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 focus of this work. However, in order to empirically evaluate SCA, we also present the second mechanism, which is required to functionally complete the model.

For now, we describe SCA outside the context of Soar (the UTC). Later in this section, we discuss how Soar has motivated the design of SCA and how the structural design of SCA differs from previous approaches.

3.1 Rule Retrieval and Acquisition

In general terms, SCA is a symbolic rule-based system that incrementally acquires prediction rules as it is trained. By a *symbolic* rule-based system, we mean that rule activation is a discrete "all or none" match. That is, a rule matches if and only if the rule's conditions are fully consistent with the internal representation of the object's description. Furthermore, as with all symbol systems, the symbolic matching affords distal access to a rule base of arbitrary size (Newell, 1990).

As SCA starts learning, it first learns very general rules that test only a few features of an example, but as learning progresses, more specific rules are acquired that test more features. Thus, there may be many rules at different levels of specificity (and correctness) that predict the same category. In trying to predict the category of an example, SCA's search process favors specific rules.

With SCA, a "concept" is ultimately defined by a distributed set of rules, for which there may be many per concept. However, because of conflicts in the rule-base, the rule retrieval process also plays a determining role in defining an SCA concept.

3.1.1 Representing Concepts as Rules

The examples SCA accepts for prediction are described in terms of symbolic attributes and values. An example can be described as follows:

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

To avoid confusion, we distinguish between the *given* example description and the internal representation SCA uses for further processing. We depict the internal representation in brackets:

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

SCA's rules test for features and predict categories. Some rules are very general (as a shorthand convention, 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 make correct predictions, and thus, the SCA search process favors more specific rules for matching 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. Given the previous set of rules and our example, the system would predict category: globe, after removing the size attribute. If several competing rules match at the same time, the system arbitrarily guesses from among one of the competing predictions.

Before we address how SCA acquires rules, let us emphasize its distinguishing properties on how it represents concepts. First, SCA's search from specific to general is controlled by knowledge that determines which feature to remove next, which is itself subject to learning. Ideally, irrelevant features would be removed first. 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,[2] or a parallel associative

[2] 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, Hopcroft, & Ullman, 1974).

```
define Predict(D)
   do
      /* Comment: retrieve category labels (R) associated
                  with description D */
      set R = recall(D)

      if size(R) > 0 then
         /* retrieved at least one label */
         set Match = true
         /* randomly select one label from retrieved set R */
         set p = choose element from R
      else
         /* no retrieval--continue search with
            modified (one less feature) description */
         set D = D - Select_Feature(D)
   until Match or size(D) = 0

   return(p)
end Predict
```

Figure 1. Performance specification.

memory, can be used for rule application. Thus, this avoids the rule-utility problem (Minton, 1988) where match time grows with the number of rules in memory. By avoiding this problem, the model can acquire large numbers of diverse concepts (Doorenbos, 1995, reports an implementation of SCA learning over 1,000,000 rules). 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.

Figure 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.

3.1.3 Rule Acquisition

When learning rules, SCA accepts an example description that includes the correct category label. Its goal is to integrate the knowledge implicit in the training example with its existing rule-based knowledge. During learning SCA searches not for the first-matched rule, but for a matching prediction rule that makes the *correct* prediction. 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.

SCA follows a simple strategy for learning a new rule that is a compromise between previously acquired knowledge and the knowledge implicit in the training example. In particular, 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.

Initially, SCA will frequently fail to find pre-existing rules that produce the correct prediction. For each of these cases, SCA must create a new rule at the most general level. This can be accomplished by either creating the rule from scratch or, as in our Soar implementation, by deriving it from a preexisting set of condition-free default rules that are only accessed once no features remain in the object description. In either case, this initial rule's condition consists of the feature that was last removed from the object description.

We will later note that the Soar architecture motivates the design of including the last removed feature in the new rule. Nevertheless, the inclusion of the last feature also has a potential performance benefit. If we assume a feature removal strategy that removes irrelevant features first, the inclusion of the last removed feature is the most relevant feature not already in the matching rule's conditions.

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 of a category prediction. Let us assume that '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. The acquisition of this new rule may be merely an intermediate step towards the acquisition of still more specific ones. Subsequent training examples will result in still more specific rules, thereby reducing the number of conflicts.

Figure 2 provides the specification of the performance process modified for training. We 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

```
define Train(D,c)  /* Description D, correct label c */
   /* after first 'do' iteration, D' will be one
      feature more specific than D */
   set D' = D
   do
      set R = recall(D)
      if c in R or size(D) = 0
         /* for training, search stops with a correct retrieval
            or when description has no features */
         set Match = true
         set p = c
         /* store new production associating c to D' */
         store(D' --> c)
      else
         set D' = D
         set D = D - Select_Feature(D)
   until Match or size(D) = 0

   return(p)
end Train
```

Figure 2. Performance specification modified for training.

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 a more specific rule with only one more feature in the condition represents a compromise between previously obtained knowledge and the knowledge implicit in the newly presented training example. Should the previously obtained knowledge be incomplete, that is, 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. As long as there are more specific rules to be learned, error recovery occurs automatically with the presentation of additional training examples that 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 typically produces the most common category for examples with the matched attributes.

The ability to recover from noisy examples by learning yet more specific rules assumes that there are more specific rules to be learned. With enough examples, the SCA rule base will have a complete set of maximally specific rules and will then have no means of recovering. Although the extent to which this happens in practice is attenuated by presenting examples with many features (including spurious, irrelevant features) and with features that change over the course of training, this observation presents a limitation on the SCA's ability to recover from noisy examples. Later, we will review this question in the context of empirical results.

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 matched. If relevant attributes are removed from the description before a match occurs, the quality of the category prediction will certainly suffer. A second consideration is that search using one feature selection order will fail to match rules acquired under a different order. Thus, a good search heuristic should also be relatively consistent 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 attributes. In the case where attributes are selected erratically, the progression towards learning specific rules will be slower. This is because the rule search may overlook a specific rule if the order of feature removal was different than when the rule was acquired. However, after enough training examples, specific rules are still acquired.

At this point, we make no specific commitment on SCA's heuristic for selecting features. We suggest that feature selection is a *deliberate* process, that is, potentially any relevant knowledge is brought to bear on selecting features. For example, knowledge stemming from advice and causal theories may impact the selection as well as any "data-driven" strategy. Despite the potential variation in control strategy, we will argue that many key behavioral properties of SCA occur independently of any particular feature selection strategy. Nevertheless, in order to implement the model for evaluation, we now specify a "default" strategy for choosing appropriate features.

3.2.1 Default Feature Selection Strategy

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 number. For example, let us assume an object description activates these two matching rules:

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

The number of conflicting prediction is two. 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), we resort to a reasonable heuristic.

We 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. We 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 different predictions increases the model's immunity to spurious inconsistencies while also stabilizing the order of feature removal.

Figure 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 we will use for modeling human behavior in Section 4. After recalling a set of predic-

```
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 attribute prediction record */
         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)
   /* maintain record of prediction averages by attribute */
   for each d in D
      Update_Avg(Attribute(d),s)
end Update

define Select_Feature(D)
   /* heuristically choose least relevant attribute in D:
      large number of avg retrievals suggests irrelevant attribute */
   set d such that Get_Avg(Attribute(d)) =
                   Max(Get_Avg(Attribute(D)))
   return(d)
end Select_Feature
```

Figure 3. Feature selection heuristic in the context of SCA.

tions, *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.

The selection heuristic of averaging attribute relevancy across all values and categories is not intended as a comprehensive model of how humans learn which features are relevant. Rather, we intend it to be an approximation of our assumption that feature relevance should shift in accordance to how useful certain features are in predicting categories.

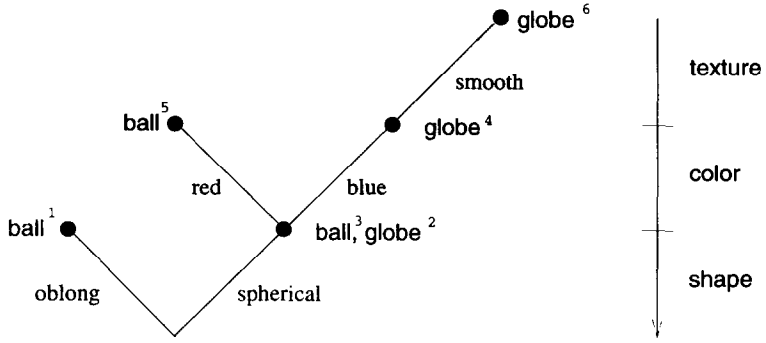

Figure 4. Graphical depiction of rules.

3.3 An Extended Example

Let us now go through an extended training trial where the feature selection order changes. In doing so, we 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 texture, color, shape. With this order, the following training examples are presented to SCA (underlined 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 4 is a graphical depiction of these rules as viewed from the current feature removal order (texture, color shape). Each dot represents a prediction rule where the prediction is the category next to the dot and the rule's

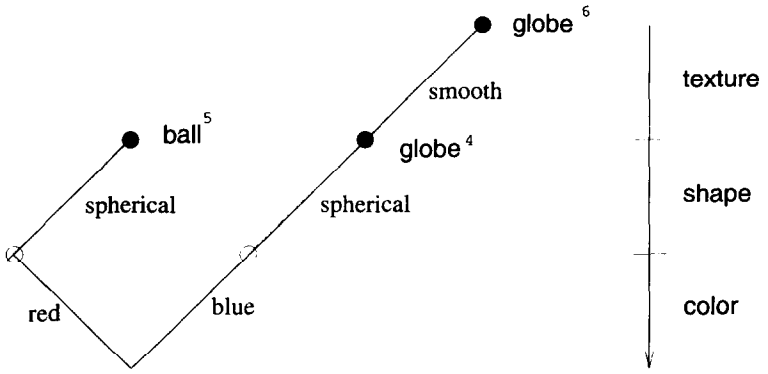

Figure 5. Graphical depiction of rules with alternate feature ordering.

conditions are all the attribute values connected below the dot. The superscripts on the predictions correspond to the rule and example numbers. Within the context of the curent feature removal order, the model's search process accesses the most specific rule that matches the object description. For example, with the performance instance of {spherical, red, smooth} and a feature removal ordering of texture, color, then shape, rule 5 would match, after the texture attribute (smooth) is removed from SCA's internal representation of the object.

Figure 4 only represents one feature ordering. After the sixth training example, the feature selection heuristic may rate shape as more relevant. In this case, color and shape are swapped producing a new ordering: texture, shape, color. Under this ordering, Figure 5 depicts the usable remainder of the same set of rules. An open dot suggests where a rule could exist but has yet to be acquired. 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. Should the order revert back to the original, rules 1, 2, and 3 would again be accessible. Rules 4, 5, and 6 are still accessible since their conditions contain both attributes whose order has changed. Thus, reordering attribute does not necessarily suppress access to 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

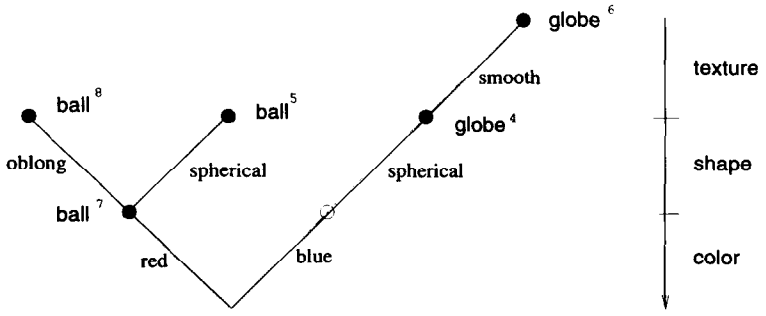

Figure 6. Final graphical depiction of rules.

Figure 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, SCA's design primarily derives from the constraints of the Soar architecture. It is primarily these constraints that distinguish SCA from all other models. In this section, we review these constraints and their consequences on SCA's design. We are not offering SCA as the definitive Soar model of category learning. Yet, as the constraints of the Soar architecture motivate its design, the following analysis illustrates how Soar memory models can address the functional and empirical demands of category learning tasks. Furthermore, by contrasting SCA's properties with other models, including those of human memory and concept acquisition, we help locate SCA in the larger space of learning models.

The major architectural constraints (and therefore distinguishing characteristics of SCA) can be summarized as follows:

1. Knowledge (long-term memory) is encoded as rules.
2. A task is performed by applying a linear sequence of discrete, deliberate operations on a temporary declarative representation (working memory).
3. A rule in long-term memory is only accessed by its successful match of working memory contents (i.e., rules cannot be directly examined).
4. All learning is the result of performance achieved through the application of prior knowledge (explanation-based learning).
5. Once a new rule is acquired, it is never lost.
6. Rule matching occurs over a discrete representation.

These are the constraints of the Soar architecture, a system of mechanisms that applies knowledge, represented as productions,[3] in creating intelligent

[3] In this article, the terms *production* and *rule* are used interchangeably.

behavior. The choice of mechanisms has led to some universal constraints within which a large, diverse set of human behaviors have been modeled (Lewis, 1993; Lewis et al., 1990; Newell, 1990; Weismeyer, 1992). Likewise, these constraints aply to SCA and thus intrinsically shape the structure of the model.

As stated earlier, SCA refers to the rule-based component of our implementation and not to the feature selection mechanism. For this article, our intention is that the implementation of the selection mechanism represents an approximation of how people use prediction feedback for evaluating the utility of selected features. While feature selection as a function of performance is not necessarily inconsistent with the Soar architecture, our particular implementation, since it recalls and updates continuous averages, is inconsistent. It remains an open issue if a comparable feature selection mechanism can be implemented within Soar's constraints.

While Soar is a rule-based architecture (Constraint 1), it is the second constraint that distinguishes Soar's (and SCA's) knowledge representation from other schemes, whether they be "rule-based" or something else (e.g., cases, frames, or prototypes). 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 (rule) 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 alter its working memory contents "in search" of the knowledge represented *procedurally* as productions. This contrasts with other rule-based systems whose learning mechanisms access the rule conditions and actions and modify them (Anderson, 1983; DeJong & Mooney, 1986; Holland & Reitman, 1978). Functionally, this constraint may seem unnecessarily restrictive; however, our means of accessing rules incorporates efficient match strategies whose matching cost does not significantly degrade as more productions are acquired (Doorenbos, Tambe, & Newell, 1992; Forgy, 1982). Randomly accessing and operating on rules would require additional time and may increase the time complexity of the learning process.

This stance along the access/cost tradeoff has several immediate consequences for our model. It allows for the liberal acquisition of rules since each additional rule comes at litle cost. However, the opaqueness of the rule-base presents a difficulty for explicitly evaluating the utility of individual rules. Rather than creating additional mechanism for indirectly monitoring rule utility, SCA simply continues to learn more specific rules, a strategy that contrasts with most other rule-based systems, which only consider more specific rules if the current rules prove inadequate (e.g., Nosofsky, Palmeri, & McKinley, 1994).

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). As a consequence, obsolete prediction knowledge must somehow be suppressed through the acquisition of new knowledge. One approach is to acquire new knowledge that explicitly rejects structure created by obsolete knowledge (Laird, 1988; Rosenbloom & Aasman, 1990). SCA takes an alternate approach that avoids the explicit rejection of obsolete structure. Since SCA probes for more specific rules first, recovery occurs by eventually learning more specific rules. As long as inaccurate rules remain less specific, they will be masked from performance.

This rule-matching strategy delivers frequency effects without explicit frequency counts, weights or probabilities (Constraint 6). For example, consider the situation where SCA is presented with a ball but is incorrectly told it is a 'globe'. SCA would acquire a general rule classifying the object as a 'globe'. 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 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 implicitly factors in specificity during performance through a serially-ordered search in long-term memory.

The classifier model (Holland & Reitman, 1978) is another production-based model. This 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 for more flexibility in acquiring and modifying rules. For performance, a series of productions may apply as determined by their strengths.

In neither the ACT model nor the classifier model does prediction performance depend on a series of *deliberate choices*, where any pertinent knowledge within the system is brought to bear. 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. The feature selection learning mechanism 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).

In related work, Billman and Heit's CARI (Billman & Heit, 1988) learns prediction rules whose construction is 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 (Feigenbaum & Simon, 1984; Simon & Feigenbaum, 1964) is most similar to SCA in that it conforms to most of Soar's architectural constraints. Indeed, a close analogue of EPAM has been implemented in Soar (Rosenbloom, Laird, & Newell, 1988). Both EPAM and SCA are discrete systems that incrementally learn more specific discriminations. EPAM is a discrimination net that discriminates among concepts by focusing on the minimal number of features needed to distinguish between concepts whereas SCA always learns a more specific rule with each new training example. As a consequence, training frequency affects the specificity of SCA's rule base more than that of EPAM's. Also, SCA's specific to general rule search differs from EPAM's serial search that effectively starts with the most general and, by incrementally testing more features, moves to the more specific. Another critical difference is that EPAM fixes which features the model should discriminate on. 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 a comparable feature selection mechanism can be implemented within Soar's constraints.

Another way to compare systems is to examine the context in which the system learns from a training example. The ACT model and classifer model learn by trying to predict the category of a training example. Learning occurs by adjusting production parameters depending on whether the pre-

diction is correct or not. Similarly, feed-forward connectionist networks (e.g., backprop nets of Rumelhart et al., 1986, and the configural-cue model of Gluck & Bower, 1988) 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 continues to make predictions until the correct prediction is made. Learning then occurs by summarizing the experience of making the correct prediction. This process leads to the acquisition of a new rule whose conditions consist of the features that led to the correct prediction, that is, the features that matched the predicting rule as well as the feature that was last removed so that the match could occur. This inductive variant of explanation-based learning (Constraint 4) is supported by Soar's chunking mechanism, a learning process that is similar to the explanation-based learning mechanisms of other AI systems (DeJong & Mooney, 1986; Mitchell, Keller, & Kedar-Cabelli, 1986). One consequence of applying chunking to an inductive task is the initial need to generate all possible predictions so that the correct prediction is available for acquiring the first, most general rule.

Exemplar-based models contrast with the above approaches of learning through performance. Examples of these models are Medin and Schaffer's Context Model (Medin & 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. Ultimately, however, both SCA and exemplar-based models represent a concept with multiple combinations of features.

Other models combine several of the above approaches. ALCOVE (Kruschke, 1992) 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 on 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. This differs from SCA, which does not have direct access to its long-term representation (Constraint 3). Both ALCOVE and COBWEB differ from SCA in that they incorporate a gradient component (i.e., weights or probabilities) in the storing and retrieving of examples (cf. Constraint 6).

In this section, we have claimed that, in addition to functional considerations, Soar's architectural principles have placed constraints on SCA's structural design, and it is these constraints and their consequences that distinguish SCA's design from other models. Given this design, we now turn to its behavioral consequences.

4. SIMULATING HUMAN BEHAVIOR

In this section, we describe some phenomena manifested by humans in learning categories and evaluate how well the model exhibits 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 & Laird, 1991) reports how SCA with hand-coded feature selection heuristics exhibits some typicality effects and a reasonable distribution of extension errors. In this article, the results are generated from SCA working with the "by attribute" selection heuristic described in Section 3.2.1 and cover a broader set of phenomena. However, much of the model's behavior is still explained through SCA's learning properties.

Category learning is perhaps one of the most studied tasks in cognitive psychology, and many effects have been reported and modeled. Among these are the behavioral consequences of varying category types, exemplar similarity, exemplar ordering, and exemplar frequency, where the behavioral consequences include measurements of prediction accuracy, response time, learning rates, and verbal confidence ratings. Our goal is not a comprehensive model of category learning. Rather, our interest is to explore the viability of discrete rule-based models and their ability to account for seemingly continuous phenomena. As such, we limit ourselves to a set of general phenomena that have motivated many gradient models, namely the ability to account for variations in both response time and error rates as a continuous function of an instance's similarity to other category members.

For our simulations, we first compare SCA's response times and error rates across three levels of typicality. Through the course of learning, we show that SCA classifies instances with higher typicality faster and more accurately. The next body of experiments evaluates SCA's predictions on typicality effects in greater detail by comparing response times and error rates of individual instances to human data. We also increase SCA's coverage by varying feature selection heuristics that correspond to different strategies that subjects were instructed to use.

In this article, we limit coverage 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, 1992) and exemplar-based models, which often have parameters controlling similarity calculations (Medin & 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.

4.1 Modeling 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). Many learning theories that use explicit gradient representations do not commit to an *algorithmic procedure* describing how the quantitative data is processed. For example, the context model (Medin & 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. 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. Similarly, ALCOVE (Kruschke, 1992) mathematically defines probabilistic, causal relationships between connections in its artificial neural net, without committing to a process that implements them. For these models, response time predictions are obtained by mapping internal metrics to response time.

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 upsupervised Kohonen net and top–down constraints of a supervised learning component.

To our knowledge, no other rule-based system provides distinct response time predictions that are separate from accuracy predictions. For example, the RULE-EX model (Nosofsky, Palmer, & McKinley, 1994) successfully produces accuracy fits for a wide range of phenomena; however, it is not

clear how one would derive response time predictions from its performance process. Another rule-based model, a discrimination tree model proposed by Ling and Marinov (1994), produces response time fits but only through an indirect interpretation of its accuracy predictions.

SCA's response time predictions are the consequence of its process, defined in terms of which processes occur in parallel (e.g., production matching) and which are performed serially (e.g., deliberate search). The rule acquisition portion of the model does not use weights, similarity measures, frequency counts or probabilities in acquiring and activating its rules. Instead it relies on an incremental search that indirectly 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 fixed 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.

We will be measuring response time as the number of feature removal iterations. We realize that there are potentially other sources of response time variation applicable to our model. For example, removal iterations do not account for the extra time that may be required in handling conflicting prediction rules that have been simultaneously activated. Also, the measure does not account for time variations for feature selection at each decision point during search. Accounting for time variations in either of these processes would require additional commitments to their implementation, which we are not prepared to do at this time. Despite these limitations, measuring time in terms of discrete search iterations still allows our model to produce testable predictions for response time variations.

4.1.1 Typicality Effects

People have a general understanding that some objects are more typical instances of categories than other objects. One example is the comparison of a robin and a penguin in the bird category. Generally, the robin is considered a more typical bird. While this does not tell us whether a penguin's status as a bird is somehow weaker than a robin, it does appear that typicality plays a role in how categories are processed. Several behavioral effects occur that vary as a function of an instance's typicality (Rosch, 1978):

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 of 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 such as fuzzy sets (Zadeh, 1965) and frequency distributions (Fisher, 1987; Lebowitz, 1987), but, as Bergadano, Matwin, Michalski, and Zhang (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, we take a more objective position by focusing on how well the external behavior of the model fits the external behavior of humans. Our focus is response times and error rates that occur over the course of learning. The third effect (giving a typical instance as an example) extends beyond the scope of our model's immediate task, that is, the supervised learning task.

To understand our 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 measured by the instance's similarity to the other instances in the same category. Rosch, Simpson, and Miller (1976) 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 fewer errors. Ideally, we would use a set of stimuli that directly corresponds to the Rosh et al. stimuli. However, their stimuli consists of letter strings, and because representing positional information would require additional assumptions which may or may not be relevant, we resort to creating our own dataset, which nevertheless uses the same theoretical measure for determining typicality. Later in this article, we will use a dataset that directly corresponds to the stimuli presented to human subjects.

Table 1 shows the stimuli used for testing typicality effects. For these 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, we presented the data for ten training cycles, where one cycle consists of each instance presented once. The presentation order was separately randomized for each cycle. Performance trials (predicting the category name) followed each training cycle in order to access performance. As we shall see, ten exposures of each training example was sufficient to illustrate the entire learning trend. We chose to repeat this process

TABLE 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

1000 times in order to ensure tight confidence intervals for each data point. This large number was able to compensate for the two sources of variation between individual runs, namely the randomization of the example presentation order and the model initially guessing which feature to include in the first rules.

Figures 7 and 8 show the results averaged over the 1000 runs. For both of these graphs, independent data points are given for each level of typicality after each training cycle (indicated by the x-axis). Figure 7 shows performance in terms of accuracy. The y-axis indicates the fraction of correct responses, where a response is considered correct if it is consistent with the training example's categorization. With 1000 runs, the largest of the 95% confidence intervals was $\pm .015$. Figure 8 shows performance in terms of response time. The y-axis indicates the number of feature-removal iterations. For response time, the largest of the 95% confidence intervals was $\pm .05$.

For all typicality levels, both performance graphs reveal an incremental improvement in performance. This is consistent with human data, as Estes (1994) notes that reaction time for categorization steadily decreases over a series of trials for all studies of which he is aware. Between typicality levels, the instances of higher typicality were generally processed more accurately and faster than instances of lesser typicality. However these differences in performance steadily decrease, especially after the 5th training cycle, and by the 8th cycle are essentially indistinguishable. At this point, almost all examples are activating maximally specific rules. As to whether the model

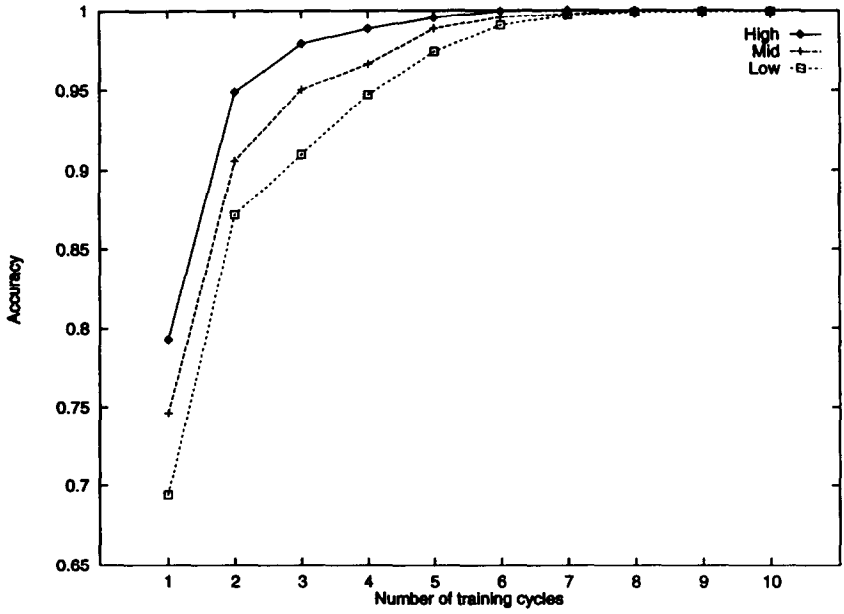

Figure 7. The effect of typicality on error rate.

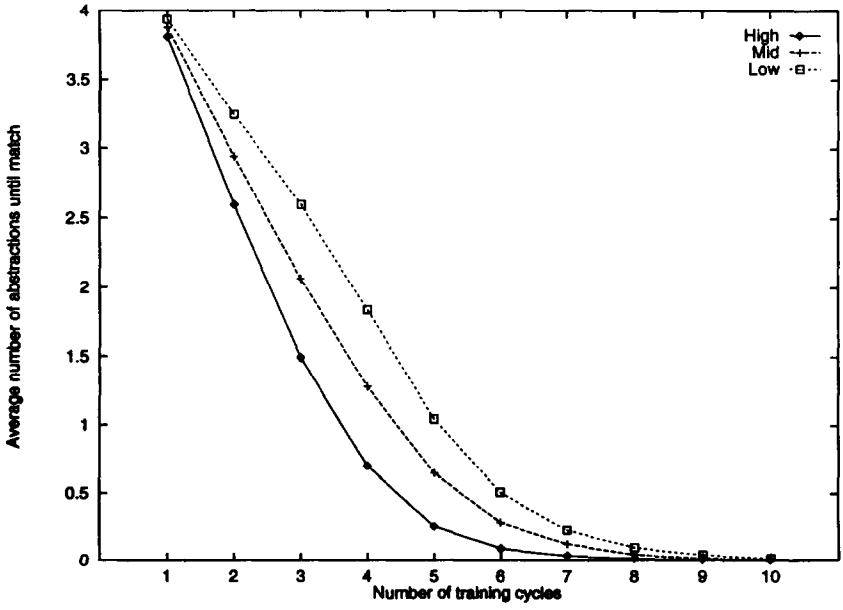

Figure 8. The effect of typicality on response time.

uniformly predicts a decrease in performance differences among typicality levels warrants further discussion, but first we shall review why the model exhibits performance differences before it has obtained maximally specific rules.

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. Instances that share features and combinations of features with other instances of the same category will access the more specific rules, and since SCA searches for specific rules first, these "typical" instances will match rules faster. A prediction from 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.

Our simulation results predict an eventual decay of performance differences across different levels of typicality. We should qualify the extent to which SCA uniformly predicts this decay. Our dataset examples are described by a small, consistent set of features and are perhaps best understood as simplified approximations of more realistic stimuli. By including additional contextual, redundant, and probabilistic features within the example, performance differences are prolonged, perhaps beyond the horizon of any observed human behavior. Furthermore, gradual incremental change in feature encodings could infinitely extend performance differences. Nevertheless, the possibility of SCA ultimately acquiring maximally specific rules suggests a potential qualification for when it exhibits typicality performance differences. Similarly, the potential for acquiring maximally specific rules presents a limitation on its ability to recover from ill-classified examples. Despite the residual effects of noise and context, these limitations imply two performance predictions as learning approaches expertise, namely that (1) performance differences across different levels of typicality decrease and (2) recovery from error becomes increasingly difficult.

The typicality effects that the model exhibits should be further qualified. For the results presented here, typicality is measured as the total similarity of an instance with other instances in the same category. In addition to this intracategory typicality, Rosch and Mervis (1975) reported intercategory typicality where typicality is defined by the degree of contrast of competing categories. SCA's search process from specific to general is perhaps overly simple in that it does not predict faster response times for instances with high intercategory typicality. Whether an instance is close to a contrasting category makes no difference in SCA's response time, at least when measured as the number of feature removal iterations. 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 that predicts a contrasting category has no bearing on the response time.

One of us (Miller, 1993, 1994) has described extensions to SCA that accounts for intercategory typicality by backtracking whenever conflicting predictions match, and, in general, can account for a tradeoff between accuracy and response time. Despite the extension possibilities, we use the simplified specific to general process in our simulations here in order to understand the behavior of the system with minimal mechanism.

4.2 Detailed Comparison to Human Data

Medin and Smith (1981) report some experimental results that allow us to test SCA's response time and accuracy predictions in greater detail. 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.

By using the results from this study, we accomplish two goals for further evaluating SCA. First, we can test SCA on a set of stimuli actually used in human experiments. By comparing relative accuracy and response time predictions for individual stimuli, we achieve a finer-grained evaluation of SCA. For example, we will test whether SCA's propensity to favor instances that share combinations of features (in contrast to individual features) is consistent with human performance. Second, we can illustrate how typicality in the context of SCA is not a strict similarity-dependent concept. In particular, we show how symbolic knowledge, presented through instruction, can penetrate SCA's classification process and thus alter the relative accuracy rates and response times.

For the Medin and Smith experiments, subjects, divided into three groups, were all trained on the stimuli presented in Table 2. They were all told they needed to learn how to categorize each stimulus into one of the two categories. Subjects in the first group (default group) were not given any particular strategy or hint on how to learn the categories. Subjects in the second group (rule-plus-exception group) were instructed to pay particular attention to one attribute (referred to as D3 in Table 2). 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 (prototype 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 by giving 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 error rates and response times on the original

TABLE 2
Stimuli Learned with Different Strategies
(adapted from Medin & Smith, 1981)

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

TABLE 3
Effects of Stimulus Type and Strategies on Human Subjects
(adapted from Medin & Smith, 1981)

Stimulus Number	Default RT	Default ER	Rule-plus Exception RT	Rule-plus Exception ER	Prototype RT	Prototype 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	.04
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

stimuli (the subjects were told their responses were being timed). Table 3 presents human response times (in seconds) and error rates for subjects' performance according to learning strategy. For this task, Medin and Smith report several interesting, statistically significant results:

- **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, that is, stimulus 4 consists of category A's most common features. On the other hand, stimulus 7 shares combinations of three features with two stimuli (15 and 4) whereas stimulus 4 only shares one such combination (that with 7).

- **The strategy affected the subjects' average performance.** On average, subjects in the prototype group performed the worst in terms of response time and accuracy. Subjects in the default group had the fastest response times while subjects in the rule-plus-exception group had the fewest errors.[4]
- **The strategy affected 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 & 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 that the researchers select in order to minimize the performance difference between the model and human subjects. There is no accompanying theory as how to assign these weights a priori.

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 responded within 1–3 seconds, a time frame where human behavior is still tightly constrained by the architecture's mechanisms (Newell, 1990). Second, the task was a category naming task, as is SCA's task, as opposed to a category verification task. Furthermore, SCA, as defined within the context of a UTC, is committed to how strategies can affect its performance. For the most part, SCA is a fixed process that acquires new rules as it guesses what category an object belongs to. Variation from this process is only granted to feature selection, and thus feature selection is the only process that is penetrable by additional knowledge, including strategic advice. The model implies that strategic advice can only affect how features are selected.

We now specify separate feature selection strategies that functionally approximate the strategic advice given to the subjects. By motivating the

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

strategies independent of the experimental results, our simulations will produce testable predictions, with which we can then compare with the human results. 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 default feature selection strategy (using averaged feedback based on conflicting rules) that is used in all other simulations in this section.
- **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, that is, 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; until rules with multiple features are learned, it must initially focus on only one feature. In order to comply best with the strategic advice, SCA must choose to focus randomly in learning rules.

By directly encoding the biases into our system, we are not committing to how these biases arise within the system. To do so would require a language understanding component as well as some limited self-awareness of the feature selection process. Without a full account of the bias learning and selection process, our model does not fully account for the timing of these processes. To the extent that human processing time varies across strategies when applying feature selection, our model will only be able to account for time differences within a strategy. However, we will still look at relative accuracy rankings across strategies and some general response time differences may be of interest.

Each of these versions of SCA was trained on the stimuli presented in Table 2. After training through five passes of the stimuli set, the model completed the performance task. At this point, error rates were roughly the same as the human data. Stopping at this point ensures that SCA has yet to learn maximally specific rules and thus preserves performance differences. In a sense, this arguably constitutes a simulation parameter. However, it is determined independent of the model's predicted performance rankings across the stimuli.

TABLE 4
Effects of Stimulus Type and Strategies on SCA

Stimulus Number	Default		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

The results in Table 4 are averages of 10,000 runs, sufficient for attaining a 95% confidence interval of ± 0.01 for both error and response time. The error figure is the fraction of times the model guessed wrong in categorization. The response time is the aveage number of iterations of feature abstraction before a prediction rule matched. Response times from incorrect predictions were filtered out of the data, as was the case for the human data.

SCA's behavior is consistent with the statistically significant results reported in Medin and Smith. We will discuss these points one by one. But first, let us review how accuracy and response time can vary. Accurate performance is the result of matching rules whose conditions (a) have discriminating features, and (b) 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.

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 time.** Stimulus 7 shares more combinations of features with other stimuli in the same category than 4. Thus, more

TABLE 5
Correlations Between Human Data and Model Predictions

Model Rankings	Human Data Rankings: Ranked by Error			Ranked by RT		
	Default	Rule	Proto	Default	Rule	Proto
Default error	**0.941**	0.881	0.872	0.941	0.712	0.848
Rule error	0.825	**0.966**	0.727	0.825	0.919	0.698
Proto error	0.941	0.908	**0.790**	0.941	0.787	0.762
Default RT	0.879	0.820	0.736	**0.879**	0.867	0.667
Rule RT	0.735	0.887	0.651	0.735	**0.946**	0.561
Proto RT	0.720	0.644	0.494	0.720	0.667	**0.467**

specific rules match the stimulus, resulting in fewer errors and faster response times.

- **The strategy affected 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 affected the relative difficulty of some stimuli.** The most problematic stimuli for the rule-plus-exception strategy are those that are exceptions to the rule (numbers 13 and 2), whereas, for the default model, these are just as difficult as two other stimuli (numbers 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. For example, stimulus 15 was ranked first in the human data (default strategy) because it has the lowest error rate. Using a Pearson correlation coefficient, a quantitative measure of correlation ranging from 1 (perfectly correlated) to -1 (perfectly inversely correlated), the ordinal ranking of the human data was compared to the ordinal ranking of SCA's data. We also compared the response time rankings of SCA to those of the subjects.

The corresponding coefficients for the default, rule-plus-exception, and prototype strategies were respectively .941, .966, and .790 for the error rankings. These figures can be located in Table 5, which shows all pairwise

TABLE 6
Correlations Among Human Data Rankings

Human Rankings	Human Data Rankings: Ranked by Error			Ranked by RT		
	Default	Rule	Proto	Default	Rule	Proto
Default error	1.000					
Rule error	0.836	1.000				
Proto error	0.912	0.782	1.000			
Default RT	1.000	0.836	0.912	1.000		
Rule RT	0.703	0.904	0.619	0.703	1.000	
Proto RT	0.912	0.703	0.937	0.912	0.500	1.000

correlations between the human data rankings and the model's data rankings. The correlation figures in boldface indicate the pairings that we motivated a priori. The remaining figures show the extent to which the alternate SCA models fit the different experimental conditions and thus indicate whether other variations of SCA might provide a better model of a particular condition. That the a priori pairings for default and rule-plus-exception error rankings were the best among the other pairings suggest that these pairings were appropriate. On the other hand, that the default SCA model is a better predictor than random SCA model for the human prototype data suggests that a variant of the default model may be a better model for this group.

As an additional point of reference, the coefficient between human data with the default strategy and human data with the rule-plus-exception strategy was .836 (this figure can be found in Table 6, which shows the pairwise correlations among the human data rankings). 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), again suggesting that the pure random strategy does not capture the peculiarities of the aggregate human data.

Response time predictions followed a pattern similar to error rate predictions with coefficients of .882, .946, and .483 (for default, rule-plus-exception, and prototype, respectively), which can also be located in Table 5. For the sake of completeness, this table also provides comparisons across the dependent variables of error and response time and thus indicates whether the model's RT's predict human errors and vice versa. These results seem relatively mixed and appear to reflect the extent to which human errors and RT's are correlated.

One additional point of reference for comparing the ranking correlations between SCA and human data is the ranking correlations between the human data for performing the timed task and human data for performing

an untimed task with the same stimuli. In addition to the timed classification task, Medin and Smith (1981) had subjects perform classification where the responses were not timed. The tested stimuli were the same used in the timed task, but also included 7 additional stimuli that 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 timed 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 timed classification than human data from the otherwise identical task of untimed classification.

Across all these points of reference, the default and rule-plus-exception models seemed particularly useful for explaining the corresponding human data whereas the random-selection model was unable to capture peculiarities of the human prototype data and only seemed to match the human data in that its rankings shared some overall similarity with all other rankings. Indeed, the default model provided better predictions to the prototype human data. From the perspective of our model, these results suggest that the subjects in the prototype condition ultimately favored some features over others and thus violated the intent of the condition's instructions. Alternatively, it may be the case that SCA cannot provide a context for matching the subjects' classification strategies in the prototype condition. The issue could possibly be resolved by an ad hoc fit using alternate selection strategies. Inevitably a better fit would be found as evidenced by the model's default predictions. However, to avoid the possibility of overfitting, future comparisons will ultimately require the consistent use of selection strategies across many datasets.

We performed one additional simulation for evaluating SCA's ability to account for graded performance. 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 for its ability to predict the categories of 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.

[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.

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 (1981) 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 what Newell calls the time band of immediate behavior and where he claims human behavior is most constrained by an architecture's mechanisms (Newell, 1990). 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.

5. CONCLUSION

We have described SCA, a discrete rule-based model that yields graded performance as a function of category similarity. Amidst the implementational details, SCA is distinguished by a particular handful of structural properties, listed in Figure 9. These are essential properties on which the successful evaluation of SCA depends. In a sense, these lay out a class of models, of which we 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 paper. Thus the essential properties depict a set of least commitments constraining the construction of future models.

Foremost is the rule-based representation. Through the use of symbolic values and by restricting direct access to the rule-base, our model's implementation is able to efficiently access category names and thus scale up to plausibly large numbers of examples and categories. This efficiency brings a tradeoff of not directly accounting for flexible, graded performance. Instead, the remaining commitments bring us to a model that achieves partial match performance and the ability to recover from incorrect associations although

- **Distributed, rule-based representation.** A concept is represented by a distributed set of rules (productions). *Section 3.1.1.*
- **Discrete matching of rules with parallel activation.** Rules fully matching the internal symbolic object description are activated in parallel, within constant time. *Section 3.1.2.*
- **Serial search.** Rules that do not fully match the internal representation are activated with a serial search. *Section 3.1.2.*
- **Search from specific to general.** The serial search pattern tries to activate 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 3.2, 3.4 and 4.2.*
- **Learning specific rules from general rules.** New, more specific rules are acquired from applying more general rules. *Section 3.1.3.*

Figure 9. SCA's essential structural properties

we have noted that this ability, as it stems from continually learning more specific rules, is ultimately limited to the extent that more specific rules can still be acquired.

We suspect our rule-based representation also has consequences beyond our chosen task of category naming. The opaque rule-base is not immediately operational for other category tasks. For example, using the category-naming rule-base to specify category examples would require a deliberate search strategy of hypothesizing plausible examples and then evaluating the quality of their example according to its specificity. Extending the model to other tasks within the Soar architecture could make new predictions between the relationship of how knowledge structure impacts the task and vice versa.

Absent from our list is a commitment to our default feature selection strategy. We believe that many strategies and knowledge sources play a role in selecting features and, with all else being equal, selects features in order to improve performance. At this time, we are ambivalent as to whether this requires maintaining average performance values as used in our default strategy, or whether a simpler strategy that maintains less information content would suffice. Our simulations showed graded performance with several different strategies with only a few local order differences in the rankings, suggesting the extent to which feature selection alters the performance of our model. One item for further investigation are more detailed simulations comparing relative performance with other feature selection strategies.

Also missing from our list is a specific granularity at which objects are represented. In our simulations, objects are described with four or five

symbolic features. Human subjects may actually encode more features, which may be irrelevant, noisy, or redundant. For this reason, and because of other theoretically-independent implementational details, such as the feature selection implementation and the model's measure of time, we have resorted to ordinal comparisons to human data. Presumably, the simplified object description leaves intact the relational correspondence between individual stimuli and thus allows for reasonable ordinal predictions.

Finally, we emphasize that our model's properties are not directly drawn from human behavior. Instead, they are a result of applying constraints imposed by functionality and architectural demands. As a consequence, we have been led to a model that serially searches discrete, symbolic rules, and thus contrasts with currently popular "gradient" approaches. We have demonstrated how this discrete search process produces flexible behavior with varied error rates and response times that conform to human behavior. In so doing, the model suggests that typicality, as manifested by accuracy and response time, need not be a phenomenon directly supported by architecture. Rather, it can be an emergent property of the process that runs on top of the architecture and that meets the demands of the task. Our model also suggests that typicality is not necessarily a wholly similarity-dependent concept, as it can be subject to background knowledge. In particular, we have demonstrated how symbolic knowledge originating from instruction can penetrate SCA's classifying process and thus influence accuracy and response time.

REFERENCES

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

Aha, D.W., Kibler, D., & Albert, M.K. (1991). Instance-based learning algorithms. *Machine Learning, 6,* 37–66.

Aho, A.V., Hopcroft, J.E., & Ullman, J.D. (1974). *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley Publishing Company.

Anderson, J.R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

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

Anderson, J.R. (1991). The adaptive nature of human categorization. *Psychological Review, 98,* 409–429.

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

Bergadano, F., Matwin, S., Michalski, R.S., & Zhang, J. (1992). Learning two-tiered descriptions of flexible concepts: The POSEIDON System. *Machine Learning, 8,* 5–43.

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

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1,* 145–176.

Doorenbos, R.B. (1995). *Production matching for large learning systems.* Unpublished doctoral thesis, Carnegie Mellon University, Pittsburgh.

Doorenbos, R., Tambe, M., & Newell, A. (1992). Learning 10,000 chunks: What's it like out there. In *Proceedings of the National Conference on Artificial Intelligence,* pp. 830–836.

Estes, W.K. (1994). *Classification and cognition*. New York: Oxford University Press.

Fayyad, U.M. (1991). *On the induction of decision trees for multiple concept learning.* Unpublished doctoral thesis, The University of Michigan, Ann Arbor.

Feigenbaum, E.A., & Simon, H.A. (1984). EPAM-like models of recognition and learning. *Cognitive Science, 8,* 305–336.

Fisher, D.H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2,* 139–172.

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

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

Gluck, M.A., & Bower, G.H. (1988). Evaluating an adaptive network model of human learning. *Journal of Memory and Language, 27,* 166–195.

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

Holland, J.H., & Reitman, J.S. (1978). Cognitive systems based on adaptive algorithms. In D.A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic.

Kruschke, J.K. (1991). Dimensional attention learning in models of human categorization. In *The 13th Annual Conference of the Cognitive Science Society,* pp. 281–286.

Kruschke, J.K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review, 99,* 22–44.

Laird, J.E. (1988). Recovery from incorrect knowledge in soar. In *Proceedings of AAAI-88,* pp. 618–623.

Laird, J.E., Newell, A., & Rosenbloom, P.S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33,* 1–64.

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

Lewis, R.L. (1993). *An architecturally-based theory of human sentence comprehension.* Unpublished doctoral thesis, Carnegie Mellon University, Pittsburgh.

Lewis, R.L., Huffman, S.B., John, B.E., Laird, J.E., Lehman, J.F., Newell, A., Rosenbloom, P.S., Simon, T., & Tessler, S.G. (1990). Soar as a unified theory of cognition: Spring 1990. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*. Cambridge, MA.

Ling, C.X., & Marinov, M. (1994). A symbolic model of the nonconscious acquisition of information. *Cognitive Science, 18,* 595–621.

Martin, J.D., & Billman, D.O. (1991). Variability bias and category learning. In *Machine Learning: Proceedings of the Eighth International Workshop*, pages 90–94.

Medin, D.L., & Schaffer, M.M. (1978). Context theory of classification learning. *Psychological Review, 85,* 207–238.

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

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

Miller, C.S. (1993). *Modeling concept acquisition in the context of a unified theory of cognition*. Unpublished doctoral thesis, The University of Michigan, Ann Arbor. (Also available as Technical Report CSE-TR-157-93).

Miller, C.S. (1994). Modeling inter-category typicality within a symbolic search framework. In *The 16th Annual Conference of the Cognitive Science Society,* pp. 635–639.

Miller, C.S., & Laird, J.E. (1991). A constraint-motivated model of concept formation. In *The 13th Annual Conference of the Cognitive Science Society*, pp. 827–831.

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

Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1,* 47–80.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Nosofsky, R.M., Palmeri, T.J., & McKinley, S.C. (1994). Rule-plus-exception model of classification learning. *Psychological Review, 101(1),* 53–79.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning, 1,* 81–106.

Rosch, E. (1978). Principles of categorization. In E. Rosch & B.B. Lloyd (Eds.), *Cognition and categorization*. Hillsdale, NJ: Erlbaum.

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

Rosch, E., Simpson, C., & Miller, R.S. (1976). Structural bases of typicality effects. *Journal of Experimental Psychology: Human Perception and Performance, 2,* 491–502.

Rosenblatt, F. (1962). *Principles of neurodynamics*. New York: Sparten.

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

Rosenbloom, P.S., Laird, J.E., & Newell, A. (1988). The chunking of skill and knowledge. In B.A.G. Elsendoorn & H. Bouma (Eds.), *Working models of human perception*. London: Academic.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart & J.L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge, MA: MIT Press.

Schlimmer, J.C., & Fisher, D. (1986). A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 115–121.

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

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

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

Weismeyer, M. (1992). *An operator-based model of human covert visual attention*. Unpublished doctoral thesis, The University of Michigan, Ann Arbor.