

A brief overview of Symbolic Concept Acquisition (SCA)

Robert E. Wray (wray@soartech.com)

Soar Technology, Inc. 3600 Green Court Suite 600 Ann Arbor, MI 48105 USA

Symbolic Concept Acquisition

SCA is an exemplar category learning model and has been shown to be consistent with a range of phenomena in human category learning such as typicality effects, response time effects for typicality and practice, comparable learning rates for linearly separable and non-separable categories, and base-level effects (Miller 1993; Miller and Laird 1996). This document seeks to introduce the SCA algorithm independent of its implementation in Soar(Newell 1990).

Figure 1 presents a high-level representation of the SCA learning and prediction algorithm. The same processing loop is used for both prediction (no feedback) and training/learning (feedback available). When tasked to predict the category of an instance without feedback, SCA performs a specific-to-general search over prediction rules. It first attempts to recall prediction rules for all features (2), and then progressively abstracts features (3) to search for less specific prediction rules. When feedback is present, SCA performs the search for a matching prediction rule. This rule is specialized (6) with the last feature abstracted from the instance (remembered at 4). It stores this specialized rule as a new prediction rule (7). Over multiple learning trials, this learning results in a general-to-specific search over the feature space. That is, SCA generally learns rules sensitive to one feature, then to two, and so on. The concept representation becomes more specific as more features (and combinations of features) are incorporated into learned prediction rules.¹

<pre>1. instance = features and values 2. while (no matching prediction rule for instance) 3. abstract feature from instance 4. remember most recently abstracted feature 5. if (no feedback) return prediction else 6. restore most recently abstracted feature to instance 7. store new prediction rule for instance</pre>	<p>SCA Learning Example:</p> <p>Available prediction rules:</p> <ol style="list-style-type: none">1. (null) ⇒accept2. (null) ⇒reject3. fuel 20⇒accept <p>New instance:</p> <ul style="list-style-type: none">– size S, turbulence 3, fuel 20– category accept– Abstraction order: size, turbulence, fuel <p>New Prediction Rule:</p> <p>Rule 4. fuel 20, turbulence 3 ⇒accept</p>
<p>Figure 1 A pseudocode representation of the SCA algorithm (left) and an example learning trial (right). (adapted from (Wray and Chong 2003)).</p>	

Figure 3 also presents an SCA learning example. The example assumes that the model has previously learned a prediction rule (Rule 3) that indicates an instance with a fuel percentage of 20 should be accepted. A new positive instance (S,3,20) is presented. For this example, assume the abstraction order is size, then turbulence, then fuel. There are no matching prediction rules for the input instance. SCA then abstracts size from the instance, leaving (3,20). Again, it looks for prediction rules for this instance, and, finding none, abstracts the turbulence value, leaving (20). Rule 3 matches this instance. SCA now specializes Rule 3, adding the last abstracted feature and value (turbulence 3). The new rule, Rule 4, indicates that (3,20) instances should be accepted. Had the example instance been negative, SCA would have learned a prediction rule that indicated instances with fuel values of 20 should be rejected. In this case, given the previously learned prediction rule (i.e., fuel 20⇒accept), the model would have come to recognize that fuel values of 20 could not be used, by themselves, to make category predictions.

¹ Examples in this document are drawn from the Agent Modeling & Behavior Representation (AMBR) simulated air-traffic control task. In this task, a subject/model must learn to accept or reject an “altitude change request” based on three two-valued features (fuel [20 or 40], turbulence [1 or 3], and size [S or L]). Wray & Chong (2003) discuss this task and the application of SCA to this task.

SCA’s learning rate will vary based on the order of feature abstraction. There are two obvious possibilities: a random abstraction order or a systematic abstraction order. With random abstraction order, the search over prediction rules is similar to a breadth-first search, generating many one-feature rules, then two-feature rules, etc. With a systematic abstraction order, the search is more like a depth-first search over prediction rules, specializing rules with the most relevant (last abstracted) feature, to ones with the two most relevant features, etc. Examples of the progression of rule learning for the two types of abstraction orderings are shown in Table 1. The random approach is generally slower than the systematic approach because the depth of the search space over all prediction rules is relatively shallow, relative to its breadth, and all leaf nodes represent acceptable predictions (i.e., because the experimental design assumed deterministic exemplars over all possible feature vectors).

Random Abstraction Order Example Rule Learning	Systematic Abstraction Order Rule Learning
1. fuel 20 ⇒accept	1. turb 1 ⇒accept
2. size L ⇒reject	2. turb 3 ⇒reject
3. fuel 40 ⇒reject	3. turb 1, fuel 20 ⇒accept
4. turb 1 ⇒accept	4. turb 1 ⇒reject
5. size S ⇒accept	5. turb 1, fuel 40 ⇒reject
6. size L, turb 1 ⇒reject	6. turb 3 ⇒accept
7. turb 1 ⇒reject	7. turb 3, fuel 20 ⇒accept
8. size L ⇒accept	8. turb 1, fuel 40, size S ⇒reject
9. fuel 20, turb 1 ⇒accept	9. turb 3, fuel 40 ⇒reject
10. size S ⇒reject	10. turb 1, fuel 20, Size S ⇒accept

Table 1. Example progression of rule learning in random (left) and systematic (right) abstraction orderings

The SCA model also includes relevant feature recognition, which can impact abstraction order. In order to capture this kind of knowledge, SCA includes a simple implementation for relevant feature detection. The model proposes that a feature may be most relevant to the prediction when an ignored (e.g., abstracted) feature leads to an incorrect prediction. It then considers the feature that was ignored a “relevant feature” and abstracts it last, rather than randomly, in the future. The relevant feature can/will change as the model is run when more than one feature is needed for detection..

Transfer Task

Transfer tasks are used in psychological experiments to test the responses of subjects to novel stimuli after having been trained on a related training set. Without adding additional knowledge to SCA, SCA would revert to random choice for transfer task examples with completely novel attribute values. For instances that mixed both trained and novel stimuli, it might be able to recall intermediate prediction rules for an instance. For example, for the instance fuel=20, size=XS, turbulence=3, then a rule that made a prediction using only these values of fuel and turbulence might be accessed during the transfer task.

The use of only these intermediate rules, overlooking common knowledge of the relationships between novel and trained attribute values, does not seem very plausible. Therefore, this version of SCA includes added knowledge to the model that will “map” unknown feature values to the values specified in the task instructions. This knowledge reflects very basic, common sense knowledge about the values of scalars and sizes. For instance, the value “10” is closer to “20” than “40”; therefore, values of 10 will be mapped to 20. It seemed plausible that subjects would use this kind of knowledge to map an outlier value to a known value for a prediction; thus, fuel=20, size=XS, turbulence=3 would be mapped to fuel=20, size=S, turbulence=3, a trained instance, and the subject would respond with the prediction learned for this instance.

There are two unresolved issues for this strategy. First, how should intermediate values like 30 be mapped? We considered three obvious possibilities: 1) map to the lower value, 2) map to the higher value and 3) leave the value unchanged. Thus, an instance with a fuel value of 30 could get mapped to the known values of 20 or 40 or not changed at all. For the transfer task, we designed the model to make a random choice among the three options, each option having equal probability. In only the last case (fuel 30) would any intermediate prediction rules be used.

Second, we considered capacity limitations on the mappings. For example, an instance of 10,4,XL requires 3 mappings (to 20,3,L) and then the retrieval of the prediction rule for this trained instance. There is no ambiguity in the mapping, but rather some question as to whether subjects can readily perform all three mappings and then retrieve a prediction for the instance held in mind. Our goal was to develop a solution that would limit the power of the mappings but also to minimize any changes to the model, including the mappings, since re-working the model was not in the spirit of a transfer task.

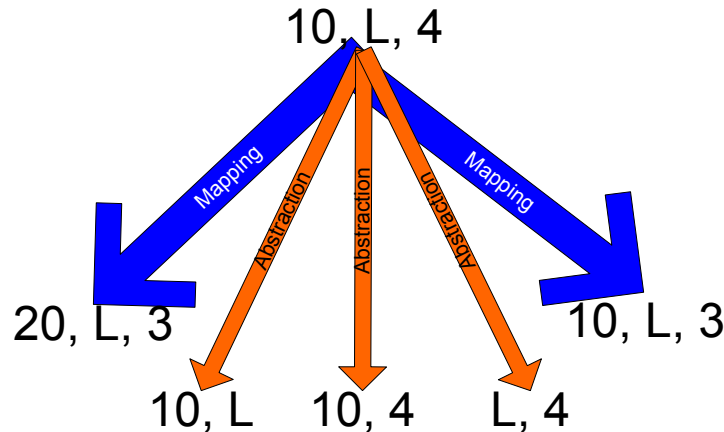


Figure 2. Example showing the competition between abstraction and mapping in the Round 2 transfer task model.

In the SCA transfer task extensions, the only change we made to the model was to allow prediction and mapping to compete. In SCA, prediction is accomplished through the deliberate choice to ignore (abstract) a feature which then allows the model to retrieve any prediction rules matching the partial instance. We changed the mapping knowledge so that the model could, in any decision, choose to map a feature or abstract it. Abstraction operators are proposed at the beginning of the prediction process as well as any mapping operators (as before, equidistant values lead to the proposal of three mapping operators: map lower, map higher, do not map). From the perspective of a human subject, the result is that sometimes a particular feature is mapped, and sometimes it is ignored. Figure 2 illustrates the process. Each choice is equally likely. When one of the candidates is selected, its operation is executed and then process repeats. For example, if the model abstracted the size feature in this example (“L”), then the other four choices shown in the figure would be available for the resulting “10 4” instance.

References

- Miller, C. S. (1993). Modeling Concept Acquisition in the Context of a Unified Theory of Cognition. EECS. Ann Arbor, University of Michigan.
- Miller, C. S. and J. E. Laird (1996). "Accounting for graded performance within a discrete search framework." Cognitive Science **20**: 499-537.
- Newell, A. (1990). Unified Theories of Cognition. Cambridge, Massachusetts, Harvard University Press.
- Wray, R. E. and R. S. Chong (2003). Quantitative Explorations of Category Learning with Symbolic Concept Acquisition. International Conference on Cognitive Modeling, Bamberg, Germany.