AUTOMATIC GENERATION OF OBJECT CLASS DESCRIPTIONS USING SYMBOLIC LEARNING TECHNIQUES

R. L. Cromwell and A. C. Kak

Robot Vision Lab
School of Electrical Engineering, Purdue University
West Lafayette IN 47907 USA
cromwell@ecn.purdue.edu, kak@ecn.purdue.edu

Abstract

Object recognition requires complicated domain-specific rules. For many problem domains, it is impractical for a programmer to generate these rules. A method for automatically generating the required object class descriptions is needed – this paper presents a method to accomplish this goal. In our approach, the supervisor provides a series of example scene descriptions to the system, with accompanying object class assignments. Generalization rules then produce object class descriptions. These rules manipulate non-symbolic descriptors in a symbolic framework; the resulting class descriptions are useful both for object recognition and for providing clear explanations of the decision process. We present a simple method for maintaining an optimal description set as new examples (possibly of previously unseen classes) become available, providing needed updates to the description set. Finally, the system's performance is shown as it learns object class descriptions from realistic scenes – video images of electronic components.

Introduction

Consider the following unsolved problem in model-based computer vision: Suppose a system can discriminate between N different object classes in a library, possibly using one of the many modern graph-theoretic approaches (Bolles and Horaud 1986, Chen & Kak 1989, Faugeras & Hebert 1986, Grimson & Lozano-Perez 1984, Kim & Kak 1991, Oshima and Shirai 1983). Each model might be a graph whose nodes correspond to surfaces and whose arcs signify adjacency, each node and arc represented by a frame of attribute-value pairs. Now suppose we add a new object class to the library. Can the system be designed in such a manner that it would automatically figure out how to discriminate between the now $N+1$ classes? This problem is not the same as addressed in (Hansen & Henderson 1988, Ikeuchi & Kanade 1988); those efforts focus on automatic generation of efficient single-class object recognition strategies rather than on multi-class discriminations. A recognition strategy is more concerned with sequencing various tests to quickly establish the identity and pose of an object and less so with automatically generating descriptions that discriminate between different classes.

To illustrate the problem of multi-class discriminations, suppose the system can already discriminate between red blocks and red cylinders. A computationally efficient system would seek a minimal set of features, and simply ignore color. If the system must now also recognize blue cylinders as a distinct class, the system should automatically add color to its its model graph descriptions.

In this paper, we show how such a system can be built using the methods of symbolic learning, methods that are perhaps best exemplified by the work of Michalski and his colleagues (Dietterich & Michalski 1979, Dietterich & Michalski 1981, Michalski 1980). Central to these methods is generalization, a concept also investigated by other researchers (Baim 1988, Hayes-Roth & McDermott 1977, Hayes-Roth & McDermott 1978, Vere 1975, Vere 1977). Stated superficially, generalization examines a set of symbolic descriptions to find statements that are true of all members. Generalization is not merely a search for common symbolic descriptors, but also draws conclusions such as *"all objects are polygonal"* when told *"type-1 objects are square"* and *"type-2 objects are triangular"*. To form such "higher level" generalizations, the system must know hierarchical relationships between attribute values.

The methods proposed in the cited literature on symbolic learning addressed the problem of learning pattern descriptions – especially 2D patterns in simple line drawings – but their application in computer vision has not been easy. Since these contributions focused more on the process of learning generalizations, the authors assumed that perfect symbolic descriptions of the patterns were available. In practical computer vision problems, object descriptions are rarely purely symbolic. Even in 2D vision, segmentation problems may preclude categorical assessments of object shape. There may additionally be models that have, at a purely symbolic level, identical descriptions, although they differ with respect to numerical attribute values.

Contrary to what might appear to be the case at first thought, the main difficulty with using numerical attribute values is not their usual associated uncertainty, but the large number of possible generalizations. Suppose we show two cylinders, of lengths 10 cm and 12 cm, to a learning vision system. The system may form any of the following five generalizations:

This work was supported by the National Science Foundation under Grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems.

1) *All lengths are equal to or shorter than 12 cm.*
2) *All lengths are equal to or longer than 10 cm.*
3) *All lengths are between 10 cm and 12 cm.*
4) *All lengths are shorter than 10 cm or longer than 12 cm.*
5) *All lengths take one of two integer values; 10 cm or 12 cm.*

Given several possible generalizations for an object class, one of them should be chosen to represent the class on the basis of the following principle: *the generalization chosen should be maximally general while possessing sufficient discriminatory power for all the required interclass distinctions.* If a human told the system that there existed a different class of objects consisting of two cylinders of lengths 14 cm and 18 cm, it should discard generalizations #2 and #4. If at a later time a cylinder of length 10.5 cm were added to the first class, the system should discard (or modify) generalization #5.

In this paper, we have embedded these ideas in a working vision system that evaluates the different generalizations that are possible when numerically valued attributes are used and discards those violating the above principle. As new examples become available, possibly of previously unseen classes, the system automatically updates the object class descriptions as needed. In the experiments described at the end of this paper, the system first learned generalized descriptions for two object classes. Examples of a third class were then introduced, and the system updated the set of descriptions. The updated descriptions were used to classify previously unseen objects and the error performance measured. The error rates are clearly a function of the number of examples used for the formation of descriptions. Our preliminary results are encouraging in the sense that even with very few training samples, the system correctly identifies most of the previously unseen objects.

Learning via Generalizations

A learning system in our context must be capable of four critical tasks. The first is automatically forming graph-theoretic or first-order predicate-calculus generalized descriptions of objects, the two description types being isomorphic. Second, evaluating descriptions to test their discriminatory power. Third, modifying object class descriptions with available information until the desired discriminatory power is attained. Finally, using new information as it becomes available to form new descriptions that either possess greater discriminatory power and/or are more computationally efficient.

Even a learning system must possess what may be called generic knowledge, provided by the supervisor and possibly context-dependent. The system must know that an entity has measurable qualities that can be described by attribute-value pairs (henceforth called AV-pairs). It must also be conversant in the language of object descriptions, meaning that it knows about all attributes measurable by available sensors, the ranges of values of these attributes, and how these AV-pairs can be embedded in a graph representation.

Before explaining how the system can combine explicit example descriptions into generalizations, in the process discovering which attributes are useful for the desired discrimination, we must first formally define an object description.

Definition 1: Given n measurable qualities of an object, the ith quality expressed as AV-pair $(A_i: V_i)$, a description D is a WFF with AV-pairs as sentence symbols,

$$D = ((A_1: V_1) \wedge (A_2: V_2) \wedge \cdots \wedge (A_n: V_n)),$$

and is thus a logical statement about the values taken by a set of attributes.

Sensory information for an example object j provides an assignment function v_j specifying explicit values for the attributes; the question *"Does the description D hold for object j?"* is answered by the truth value of $v_j(D)$.

Definition 2: For a single AV-pair, $v_j((A_i: V_i)) = TRUE$ if and only if any of the following is true:

(*a*) A_i was measured for object j and found to have value V_i, or
(*b*) V_i is a variable, or
(*c*) V_i is an expression satisfied by the measured value (e.g., if v_j specifies any $length \leq 8$, then $v_j((length: \{X: 8 \leq X\})) = TRUE$.

If description D is a conjunction of more than one AV-pair, then $v_j(D) = TRUE$ if and only if $v_j((A_i: V_i)) = TRUE$ for all AV-pairs $(A_i: V_i)$.

With a formal definition of a description and a method for determining if that description holds for an example object, we can now define a generalization.

Definition 3: Given N descriptions, each corresponding to one of N example objects, a generalization is a description that holds for all N objects. Given example descriptions $\{D_1, D_2, \ldots D_N\}$, the generalization **D** is denoted by:

$$\forall D_i \in \{D_1, D_2, \ldots D_N\}: \{ D_i \models \mathbf{D} \}$$

If $v_j(D_i) = TRUE$ for any D_i in the example set, then $v_j(\mathbf{D}) = TRUE$ also.

This definition all by itself is too weak to be useful. Consider the following three descriptions for the each of the object classes *jeep* and *car*:

$D_{jeep-1} = ((color: blue) \wedge (size: 20) \wedge (shape: boxy) \wedge (wheels: 4))$
$D_{jeep-2} = ((color: red) \wedge (size: 27) \wedge (shape: boxy) \wedge (wheels: 4))$
$D_{jeep-3} = ((color: white) \wedge (size: 30) \wedge (shape: boxy) \wedge (wheels: 4))$
$D_{car-1} = ((color: red) \wedge (size: 22) \wedge (shape: sleek) \wedge (wheels: 4))$
$D_{car-2} = ((color: blue) \wedge (size: 30) \wedge (shape: sleek) \wedge (wheels: 4))$
$D_{car-3} = ((color: white) \wedge (size: 32) \wedge (shape: sleek) \wedge (wheels: 4))$

In accordance with the definition of a generalization, identical generalized descriptions could found for the classes *jeep* and *car*:

$$D_{jeep} = ((wheels: 4)), \text{ and}$$
$$D_{car} = ((wheels: 4))$$

Such generalizations are clearly useless for discriminating between jeeps and cars, as they are far too general. At the other extreme, the system could instead construct the following, which are disjunctions of the example descriptions:

$$D'_{jeep} = (((color: blue) \wedge (size: 20) \wedge (shape: boxy) \wedge (wheels: 4)) \vee ((color: red) \wedge (size: 27) \wedge (shape: boxy) \wedge (wheels: 4)) \vee ((color: white) \wedge (size: 30) \wedge (shape: boxy) \wedge (wheels: 4)))$$

$$D'_{car} = (((color:red) \wedge (size:22) \wedge (shape:sleek) \wedge (wheels:4)) \vee ((color:red) \wedge (size:30) \wedge (shape:sleek) \wedge (wheels:4)) \vee ((color:white) \wedge (size:32) \wedge (shape:sleek) \wedge (wheels:4)))$$

While logically valid, these two generalizations are excessively specific, due to the conjuncts within each disjunct. The first two generalizations found, D_{jeep} and D_{car}, allow the *color*, *shape*, and *size* to take any value, but the second two, D'_{jeep} and D'_{car}, specify values for each; they are of limited utility, since each is unlikely to describe any future examples.

When we say "D_1 is more general than D_2", we mean that that D_1 imposes fewer constraints on the attribute values. For example, D_{jeep} is more general than D'_{jeep} as ((*wheels*: 4)) holds not only for all *jeep* examples, but also for all *cars*. Saying that a description is more specific has the converse meaning. The terms "general" and "specific" can also be applied to single AV-pairs in the same way.

Our goal is for the learning system to automatically construct object class descriptions that are maximally general but yet carry the required discriminatory power. In the ideal case, each generalization would be true only for examples of its own object class but false for those of other object classes. For most practical cases, each generalization, while being true for examples of its own class, would be false for a maximum number of examples of other classes.

In general, such descriptions include AV-pairs of varying degrees of generality, especially where attributes with numeric values are involved. In the next section, we present a set of generalization rules intended specifically for numeric valued attributes; these rules extend the rule set in (Michalski 1980). These additional rules provide the needed varying generality, but can also lead to an excessive number of generalizations. Subsequently, we present a control strategy that significantly prunes away unneeded generalizations. Following that, we discuss a method for updating a set of class descriptions when new examples are shown to the system.

Generalization Rules

In the spirit of Michalski's earlier work (Dietterich & Michalski 1979, Dietterich & Michalski 1981, Michalski 1980), rules produce generalizations based on one or more descriptions considered as inputs. The following presents, via simple examples, extensions to Michalski's rule set making the system more capable of manipulating non-symbolic descriptions. Michalski's rules included the Dropping Condition Rule, the Internal Disjunction Rule, the Climbing Generalization Tree, and the Closed Interval Rule, which we call G_{DC}, G_{ID}, G_{CGT}, and G_{CI}, respectively; for explanations of those see (Michalski 1980).

Variable Value Rule (G_{VV}) – If an attribute is allowed a variable value in any initial description, it must have a variable value in the generalization:

$$\begin{matrix}((length:\ 12.7)) \\ ((length:\ X))\end{matrix} \models_{G_{VV}} ((length:\ X))$$

Open Interval Rule (G_{OI}) – Michalski's Closed Interval Rule G_{CI} specifies an allowable range for the values of an attribute. The rule G_{OI} may be thought of as the converse of G_{CI}, as it specifies a proscribed range:

$$\begin{matrix}((length:\ 12.7)) \\ ((length:\ 8.3))\end{matrix} \models_{G_{OI}} ((length:\ \{X: (X \le 8.3) \vee (12.7 \le X)\}))$$

Given N distinct input descriptions, there are $N-1$ possible proscriptions.

Placing Upper (Lower) Bounds Rule (G_{PUB},G_{PLB}) – Similar to G_{CI} and G_{OI}:

$$\begin{matrix}((length:\ 12.7)) \\ ((length:\ 8.3))\end{matrix} \models_{G_{PUB}} ((length:\ \{X:\ X \le 12.7\}))$$

$$\begin{matrix}((length:\ 12.7)) \\ ((length:\ 8.3))\end{matrix} \models_{G_{PLB}} ((length:\ \{X:\ 8.3 \le X\}))$$

Relative Value Rule (G_{RV}) – Explicit numerical attributes may be replaced by a symbolic relationship.

$$\begin{matrix}((length:12.7) \wedge (height:2.4)) \\ ((length:8.3) \wedge (height:4.1))\end{matrix} \models_{G_{RV}} ((length:X) \wedge (height:\ \{Y: Y \le X\}))$$

External Disjunction Rule (G_{ED}) – The simplest generalization of two descriptions is their disjunction:

$$\begin{matrix}((length:\ \{W: W \le 12.7\}) \wedge (height:X)) \\ ((length:Y) \wedge (height:\ \{Z:4.1 \le Z\}))\end{matrix} \models_{G_{ED}} (((length:\ \{W: W \le 12.7\}) \wedge (height:X)) \vee ((length:Y) \wedge (height:\ \{Z: 4.1 \le W\})))$$

This rule could be used to form the not very useful generalizations D'_{jeep} and D'_{car} shown earlier. Since such generalizations are of little use, this rule should not be used on example descriptions. For this reason, in our system G_{ED} is only invoked in what will be called the High Level part of the control strategy where this rule constructs new generalizations from previously generalized descriptions.

Control of Generalization

A naïve control strategy would produce all possible generalizations of the example descriptions, and each generalization would then have to be tested for its discriminatory power. This quickly runs into computational difficulties; therefore a focusing strategy is needed to prune away needless generalizations.

The input to our system is a description, produced by a scene analysis package, for each available example object (see module #1 of Fig. 1) . These example descriptions are then grouped into classes as assigned by the human supervisor (module 2). These class-assigned descriptions are then sent to the generalization module (module 3) and, at the same time, made available to the high level control (module 4). In the generalization module, a low level control strategy, to be explained in the next section, prevents the formation of unneeded generalizations. However, module 3 accepts the input class-assigned example descriptions only after the high level control decides that

either a new class description is needed or an existing one should be modified. To make that decision, the high level control examines both the current set of class descriptions, as gleaned from the output of module 5, and all the example descriptions, as available from the current and any previous outputs of module 2. If new generalizations are needed for some class, the rules in module 3 produce them and provide them as input to module 5 where the high level control helps select the most discriminating generalizations by testing them against all the known example descriptions. These selected generalizations then serve as classification rules.

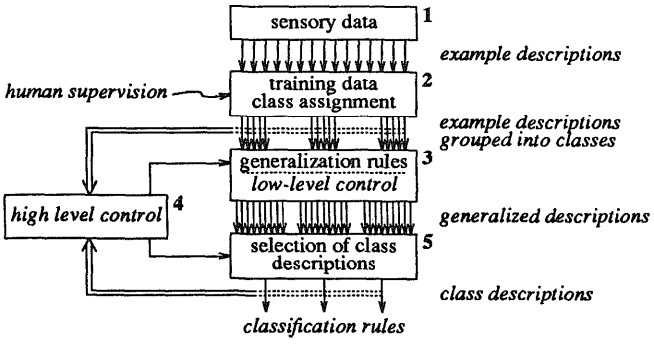


Figure 1: *Flow of control of the learning system.*

Low Level Control of Generalization

The goal of what we call the low level control strategy is to limit the formation of generalizations that are devoid of any discriminatory power. For example, since the earlier generalizations D_{jeep} and D_{car} possess no discriminatory power, their formation would be blocked.

Our low level control strategy, since it is tightly integrated with the computational process required to form generalizations, is best explained by first describing the steps required to form a generalization from example descriptions. This we do by returning to the *car-jeep* example. However, in order to simplify the explanation, we shorten the descriptions by including only the *size* and *shape* attributes. Say we have N_d descriptions of N_a attributes each (here $N_d = 3$, $N_a = 2$), and assume that N_r rule applications are possible for each AV-pair. For the *size*, the rules G_{CI}, G_{PUB}, G_{PLB}, and G_{DC} can be used. For the *shape*, whose values obey the hierarchy of Fig. 2, G_{CGT} and G_{DC} can be used.

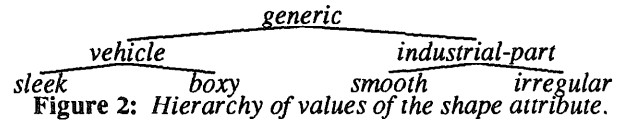


Figure 2: *Hierarchy of values of the shape attribute.*

Multi-conjunct descriptions are generalized by a split-generalize-merge sequence. In order to both control and maintain a trace of the generalization process, a network is formed for each class. The first state of the *jeep* network is N_d disconnected nodes, one for each example description. It is important that these descriptions have equivalent structures. By that we mean that if all values in each description were replaced by free variables, then all descriptions would be logically equivalent. If the descriptions are not structurally equivalent, additional AV-pairs with free variables for values are added as needed.

The N_d descriptions are split into N_a attribute clusters, linking the $N_d N_a$ new nodes to the initial descriptions as in Fig. 3.

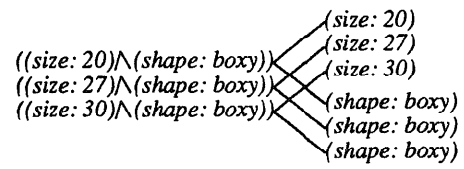


Figure 3: *Splitting initial descriptions into clusters.*

N_r generalization rules are applied to each attribute cluster, and the $N_a N_r$ cluster generalization nodes are linked to the cluster nodes, as in Fig. 4.

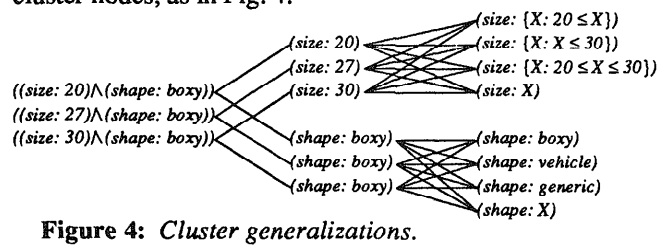


Figure 4: *Cluster generalizations.*

All possible generalizations of the initial set of descriptions may be formed by selecting one descriptor from each set of cluster generalizations. This adds $N_r^{N_a}$ new nodes linked to cluster generalizations, as in Fig. 5.

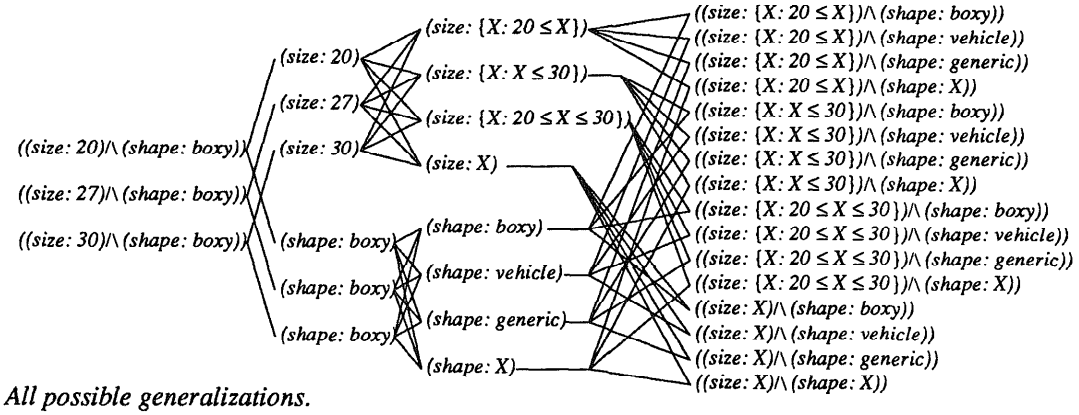


Figure 5: *All possible generalizations.*

A total of $N_d N_a + N_a N_r + N_r^{N_a}$ nodes were added to the network by this process. While the network does contain the desired generalization $((size{:}\,X) \wedge (shape{:}\,boxy))$, many other generalizations added in the expensive final step are useless, as they also hold for most (or even all) examples of *cars*. We will now describe a strategy for pruning nodes at the level of cluster generalizations, the result being far fewer generalizations at the final level. The philosophy that guides this pruning is as follows.

If a cluster generalization has absolutely no discriminatory power of its own, it cannot contribute to any description's discriminatory power. Therefore it is pointless to construct generalizations containing such AV-pairs. The system discards these AV-pairs, except for the ones whose values are free variables; they are needed to allow the system to ignore particular attributes. In the car-jeep example, no discriminatory power is gained by retaining information about the *color* attribute. The low level control allows the system to ignore *color* by including the cluster generalization $(color{:}X)$ (for the sake of intelligibility, such variable-instantiated AV-pairs were omitted from D_{jeep} and D_{car}).

To detect cluster generalizations with no discriminatory power, a WFF is formed for each. These WFF's are structurally equivalent to the initial descriptions, except that all attribute values unspecified by the cluster generalization under test are free variables. For the generalizations of the *size* attribute cluster, the four test descriptions

$((size{:}\,\{X{:}\,20 \le X\}) \wedge (shape{:}\,Y))$,
$((size{:}\,\{X{:}\,X \le 30\}) \wedge (shape{:}\,Y))$,
$((size{:}\,\{X{:}\,20 \le X \le 30\}) \wedge (shape{:}\,Y))$, and
$((size{:}\,X) \wedge (shape{:}\,Y))$,

hold for 100%, 67%, 67%, and 100% of the *car* examples, respectively. Since the AV-pair $(size{:}\,\{X{:}\,20 \le X\})$ has no discriminatory power, it is discarded.

Turning to the generalizations of the *shape* attribute cluster, the test descriptions

$((size{:}\,X) \wedge (shape{:}\,boxy))$,
$((size{:}\,X) \wedge (shape{:}\,vehicle))$,
$((size{:}\,X) \wedge (shape{:}\,generic))$, and
$((size{:}\,X) \wedge (shape{:}\,Y))$,

hold for 0%, 100%, 100%, and 100% of the *car* examples, respectively. The cluster generalizations $(shape{:}\,vehicle)$ and $(shape{:}\,generic)$ may be discarded, as have no discriminatory power. The required network reduces to that of Fig. 6 (boxes enclose pruned nodes).

Two interesting facts may be gleaned from this analysis. The first is that N_d has a very limited effect on the total number of nodes in the network. The second is that the presence of examples of other classes may greatly reduce the number of generalized description nodes needed. Pruning some fraction of the cluster generalization nodes effectively prunes an even greater fraction of the generalized description nodes.

Evaluation of a Generalized Description

Selecting a set of class descriptions in module 5 of Fig. 1 requires the system to select, from the output of module 3, those descriptions with optimal discriminatory power and maximal generality. To measure a description's discriminatory performance with respect to a single class, we measure the fraction of the examples of that class for which it holds. The set of examples of class i for which D holds is: $\{\forall j \in C_i : v_j(D)\}$, where $j \in C_i$ indicates that the supervisor assigned example j to class i. If there are N_i examples of class i, and $|x|$ is the cardinality of x, the fraction for which D holds is:

$$\eta_i(D) = \frac{|\{\forall j \in C_i : v_j(D)\}|}{N_i}$$

Representing the union of all classes other than i as class $\bar{i}$, our idealized goal is an object class description D for which a predicate $G_i(D) = TRUE$. If $\eta_i(D) = 1.0$ and $\eta_{\bar{i}}(D) = 0.0$, then $G_i(D) = TRUE$. If this goal is unattainable, the "best" description must be chosen. A simple evaluation function estimating how close an expression comes to satisfying $G_i(D)$ is:

$$h_i(D) = \alpha_i \eta_i(D) + \beta_i(1.0 - \eta_{\bar{i}}(D)),$$

where α_i and β_i reflect domain-specific false-negative and false-positive error costs for class i. Predicate $\mathcal{E}_i(D) = TRUE$ if D has an acceptable error rate. For example, if $h_i(D)$ were greater than some threshold, then $\mathcal{E}_i(D) = TRUE$.

$G_i(D)$ and $\mathcal{E}_i(D)$ evaluate the error performance of a description, allowing the system to meet the primary goal of finding an expression with sufficient accuracy. Of all generalizations with adequate discriminatory power, as measured by $G_i(D)$ and $\mathcal{E}_i(D)$, the system should select the one that is, in some sense, most general. It is beyond the scope of this paper to detail how this is done, suffice it to say that heuristics estimate the generality of each conjunct, and the estimated generality of a description is the sum of the generalities of its conjuncts. In this way, if the two descriptions below have equal discriminatory power, the

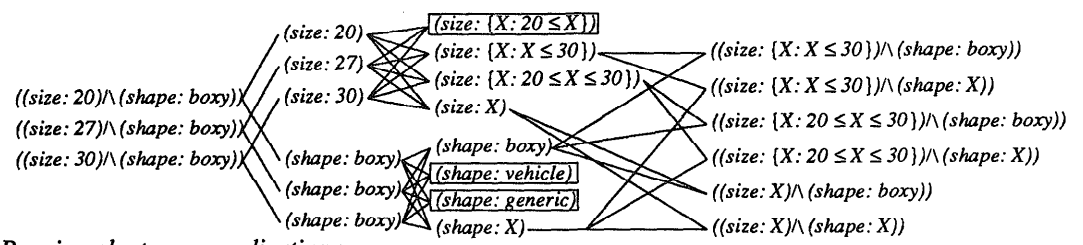


Figure 6: *Pruning cluster generalizations.*

system would choose D_2.

$D_1 = ((size: \{X: 8 \le X \le 12\}) \wedge (shape:X) \wedge (wheels:4))$

$D_2 = ((size: \{X: X \le 12\}) \wedge (shape:X) \wedge (wheels:Y))$

High Level Control of Generalization

As shown in Fig. 1, high level control orchestrates the overall learning process. Returning again to the *car-jeep* example, let us assume that some initial examples were used to find generalized descriptions for each class, and some set of these was selected as the current class descriptions, where $\mathbf{D}_i$ is the description for class i. Assume $((size:X) \wedge (shape: boxy))$ in Fig. 6 was designated as the current $\mathbf{D}_{jeep}$.

Let us assume that a new *jeep* example description D_{n+1} now becomes available. The high level control becomes aware of this, and immediately checks the current description set to see if any updating is required. If $\mathbf{D}_{jeep}$ does not hold for this new example, the high level control causes module 3 to form new generalizations that take this new information into account. From these new generalizations, module 5 then selects a new $\mathbf{D}'_{jeep}$. This new $\mathbf{D}'_{jeep}$ might be a generalization of the initial $\mathbf{D}_{jeep}$ and the new example description. If $\mathbf{D}'_{jeep}$ is used to classify a previously unseen object, the human supervisor may wish to know how the system produced $\mathbf{D}_{jeep}$. Since a network is built when generalizations are formed, links can be traced back to the original examples, as in Fig. 7 (note that attribute cluster and cluster generalization nodes are omitted for clarity). The network formed for class i is called $\mathbf{Q}_i$.

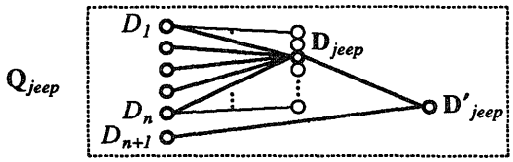
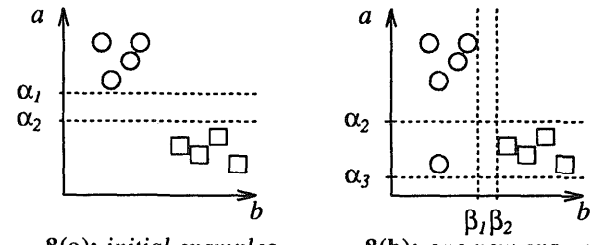


Figure 7: *Generalization* $\mathbf{D}'_{jeep}$ *extracted from network.*

When the new example description D_{n+1} becomes available, the high level control really has two choices. It can form the new description $\mathbf{D}'_{jeep}$ by generalizing $\mathbf{D}_{jeep}$ and D_{n+1}, as shown in Fig. 7, or it can effectively repeat the process by forming the generalizations of all examples D_1 through D_{n+1}. The most efficient choice, in terms of the effort required for generalization, is to generalize $\mathbf{D}_{jeep}$ and D_{n+1}. However, this can lead to overly general descriptions, which would force the system to repeat much previous work by generalizing the set $\{D_1, ..., D_n, D_{n+1}\}$.

This is conceptually illustrated in Fig. 8a. Assume that circles and squares represent two classes for which four descriptions each, in terms of attributes a and b, are initially seen by a learning system. Although the value of b is the true distinguishing factor, the system may have initially found a description for the class *circle* that specified $a \ge \alpha_1$. Similarly, the class *square* might have been described by the generalization $a \le \alpha_2$. But, if a new example of *circle* were added as in Fig. 8b, and its description generalized with the initial $\mathbf{D}_{circle}$, the new $\mathbf{D}_{circle}$ would specify $a \ge \alpha_3$. This new description would also describe every single example of *square*! In this case, the system must re-examine the original examples to discover that $b \le \beta_1$ is true for all examples of *circle* and that $b \ge \beta_2$ is true for all examples of *square*.



8(a): *initial examples* **8(b):** *one new example*

Figure 8: *Result of incorrect early generalizations.*

Returning now to the *car-jeep* example, let us further assume that a third set of example objects was originally considered to form a distinct class *truck*. After $\mathbf{Q}_{truck}$ was formed and some $\mathbf{D}_{truck}$ selected, the human supervisor may decide that *jeeps* and *trucks* should not really be separate classes, but merged into a single class *jeep2*. An external disjunction of $\mathbf{D}_{jeep}$ and $\mathbf{D}_{truck}$ could form the generalized description of this new class, as in Fig. 9. This is the reason that rule G_{ED} is needed, and its use is limited to situations of this nature.

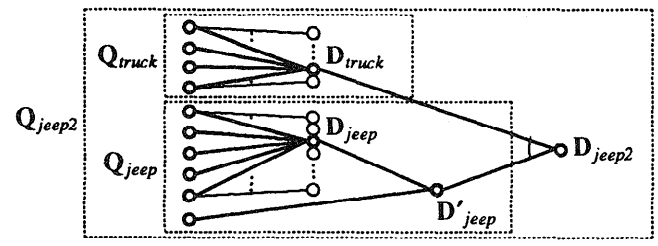


Figure 9: *Network with class truck added.*

Maintaining Multi-Class Dynamic Domains

In order to maintain an updated set of object class descriptions, the high level control must monitor both the available examples and the current set of object descriptions; in other words, module 4 of Fig. 1 monitors the outputs of modules 2 and 5. We have already shown, in an informal fashion, how the high level control monitors these outputs; in Fig. 7 a new example of class *jeep* was added. At that time, although it was not expanded in the discussion, the high level control first had to determine if the new example represented a previously known class, and also which of the current class descriptions held for it. There are six possible cases that describe the state of the system when a new example becomes available: The new object may or may not represent a currently known class. If it does not, current descriptions for other classes may or may not incorrectly hold for it. If it is of a known class, current descriptions for other classes may or may not incorrectly hold for it; additionally, the current description for its own class may or may not hold for it.

To illustrate these six cases, we denote the current set of object classes known to the system by $\mathbf{C} = \{C_1, C_2, ... C_n\}$; initially, $\mathbf{C} = \varnothing$. This six-case stra-

tegy specifies the action required whenever a new example description D_i, classified by the supervisor as an instance of class C_j, is presented to the system. Recall that $v_i(\mathbf{D}_j) = TRUE$ if $\mathbf{D}_j$, the current class description for class C_j, holds for example description D_i.

Case 1: $C_j \in \mathbf{C},\ v_i(\mathbf{D}_j) = TRUE,\ \forall k \neq j\ \{v_i(\mathbf{D}_k) \neq TRUE\}$
The new example belongs to a previously known class, and no classification errors were caused. Add node D_i to $\mathbf{Q}_j$.

Case 2: $C_j \in \mathbf{C},\ v_i(\mathbf{D}_j) \neq TRUE,\ \forall k \neq j\ \{v_i(\mathbf{D}_k) \neq TRUE\}$
The example is of a known class, with no false positive errors, but $\mathbf{D}_i$ does not hold. If no useful substitutes are available to module 5, module 3 finds further generalizations of class C_j. One of those generalizations is chosen as the new $\mathbf{D}_i$.

Case 3: $C_j \in \mathbf{C},\ v_i(\mathbf{D}_j) = TRUE,\ \exists k \neq j\ \{v_i(\mathbf{D}_k) = TRUE\}$
Class C_j is known, and $\mathbf{D}_j$ is still valid. But some other class description also holds for the new example. Again, module 5 examines the output of module 3, with the possible discovery of additional generalizations by module 3, to select a new $\mathbf{D}_i$.

Case 4: $C_j \in \mathbf{C},\ v_i(\mathbf{D}_j) \neq TRUE,\ \exists k \neq j\ \{v_i(\mathbf{D}_k) = TRUE\}$
A new $\mathbf{D}_j$ is chosen, as in case 2 above. Additionally, the erroneous descriptions for other classes are replaced, as in case 3.

Case 5: $C_j \notin \mathbf{C},\quad \forall k \neq j\ \{v_i(\mathbf{D}_k) \neq TRUE\}$
The fifth case is novel, as the new example is of an unknown class. The new class description $\mathbf{D}_j$ is simply D_i, and $\mathbf{C}$ is augmented to include C_j.

Case 6: $C_j \notin \mathbf{C},\quad \exists k \neq j\ \{v_i(\mathbf{D}_k) = TRUE\}$
The new class is added as in case 5. Additionally, the false positive classification of D_i as C_k must be handled as in cases 3 and 4.

Experimental Results

This learning system is implemented in Prolog. We will discuss the results obtained for two domains: symbolic descriptions of vehicles and numerical descriptions obtained from images of electronic components. In each domain, the system first uses examples of two classes to find generalized class descriptions. The third class is then added, and the system automatically updates the class descriptions. For the vision domain, we used the generalizations discovered to classify further examples not yet seen by the system, to test the error performance.

Symbolic Data: Vehicle Domain

The first experiment shown here is in the symbolic vehicle world we used for illustration throughout this paper. As expected, the system produced the following when given the *car* and *jeep* example descriptions presented earlier in the paper.

$$\mathbf{D}_{jeep} = ((size:X) \wedge (shape:boxy) \wedge (color:Y) \wedge (wheels:Z))$$
$$\mathbf{D}_{car} = ((size:X) \wedge (shape:sleek) \wedge (color:Y) \wedge (wheels:Z))$$

We then added examples of the new class *motorcycle*:

$$D_{cycle1} = ((size:14) \wedge (shape:boxy) \wedge (color:green) \wedge (wheels:2))$$
$$D_{cycle2} = ((size:12) \wedge (shape:boxy) \wedge (color:blue) \wedge (wheels:2))$$
$$D_{cycle3} = ((size:10) \wedge (shape:boxy) \wedge (color:black) \wedge (wheels:2))$$
$$D_{cycle4} = ((size:19) \wedge (shape:boxy) \wedge (color:red) \wedge (wheels:2))$$

The system's state was then described by Case 6 – an existing class description (*jeep*) incorrectly held for examples of a previously unknown class. The system dropped $\mathbf{D}_{jeep}$, and specified two potential replacements.

$$D_1 = ((size:20\ or\ more) \wedge (shape:boxy) \wedge (color:X) \wedge (wheels:Z))$$
$$D_2 = ((size:X) \wedge (shape:boxy) \wedge (color:Y) \wedge (wheels:4\ or\ more))$$

The system then generalized the examples of *motorcycle* and found two potential $\mathbf{D}_{motorcycle}$ descriptions:

$$D_3 = ((size:19\ or\ less) \wedge (shape:X) \wedge (color:Y) \wedge (wheels:Z))$$
$$D_4 = ((size:X) \wedge (shape:Y) \wedge (color:Z) \wedge (wheels:2\ or\ less))$$

$\mathbf{D}_{car}$ remained valid; no changes were needed.

Real Data: Component Domain

For the second experiment, an image analysis system produced descriptions of electronic components seen by a CCD camera. Each image was segmented into contiguous regions, and several attributes were measured for each region; of these, the number of pixels, the greylevel variance, the border length, and the ratio of the variance to the mean of the 2-D radius were retained (called *np*, *gv*, *bl*, and *r*, respectively). Figs. 10, 11, and 12 show examples of the classes *capacitor*, *transistor*, and *resistor*.

The system first used only the *capacitor* and *transistor* examples; the following descriptions were used (note that not all the available examples were used):

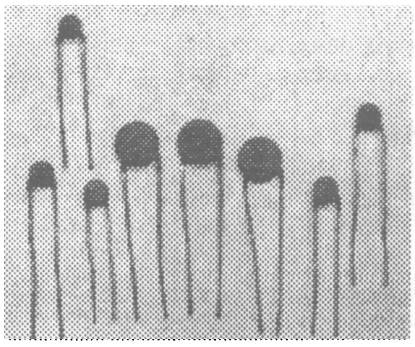


Figure 10: *Capacitors.*

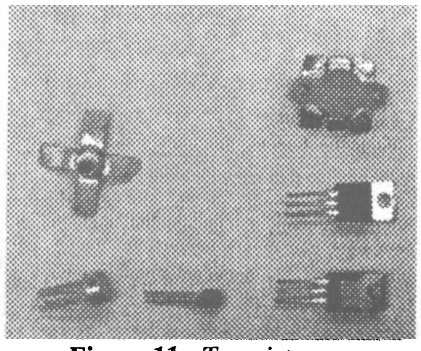


Figure 11: *Transistors.*

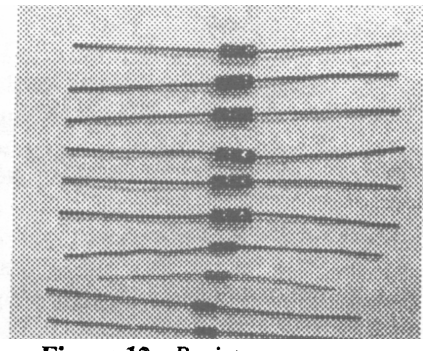


Figure 12: *Resistors.*

$D_{cap1} = ((np:697) \wedge (gv:399.7) \wedge (bl:115.4) \wedge (r:0.601))$,
$D_{cap2} = ((np:743) \wedge (gv:484.2) \wedge (bl:108.4) \wedge (r:0.274))$,
$D_{cap3} = ((np:2138) \wedge (gv:256.9) \wedge (bl:198.3) \wedge (r:0.634))$,
$D_{cap4} = ((np:2109) \wedge (gv:207.7) \wedge (bl:192.0) \wedge (r:0.494))$,
$D_{cap5} = ((np:2083) \wedge (gv:248.0) \wedge (bl:200.0) \wedge (r:0.647))$,

$D_{trans1} = ((np:1779) \wedge (gv:502.5) \wedge (bl:229.8) \wedge (r:1.105))$
$D_{trans2} = ((np:4936) \wedge (gv:250.5) \wedge (bl:400.0) \wedge (r:1.802))$
$D_{trans3} = ((np:985) \wedge (gv:378.7) \wedge (bl:209.2) \wedge (r:1.917))$
$D_{trans4} = ((np:745) \wedge (gv:928.9) \wedge (bl:122.9) \wedge (r:1.444))$

The system produced generalizations of these descriptions and selected potential class descriptions:

For the class *capacitor* –
$D_5 = ((np:not\ (743\ to\ 2083)) \wedge (gv:not\ (248.0\ to\ 256.9)) \wedge (bl:X) \wedge (r:Y))$
$D_6 = ((np:X) \wedge (gv:Y) \wedge (bl:Z) \wedge (r:0.647\ or\ less)$
For the class *transistor* –
$D_7 = ((np:not\ (1779\ to\ 4936)) \wedge (gv:not\ (378.7\ to\ 502.5)) \wedge (bl:X) \wedge (r:Y))$
$D_8 = ((np:X) \wedge (gv:not\ (378.7\ to\ 502.5)) \wedge (bl:not\ (122.9\ to\ 209.2)) \wedge (r:Y))$
$D_9 = ((np:X) \wedge (gv:Y) \wedge (bl:Z) \wedge (r:1.105\ or\ more))$

Generalizations D_6 and D_9 are intuitive – disc capacitors are nearly round, and so their radius has little variation; while transistors have highly irregular outlines. However, the other potential class descriptions (D_5, D_7, D_8) are not immediately obvious. This is an interesting result, as the system has discovered something unexpected.

The system was then shown the following *resistor* examples, again causing a Case 6 situation:

$D_{res1} = ((np:563) \wedge (gv:562.3) \wedge (bl:105.9) \wedge (r:2.07))$
$D_{res2} = ((np:687) \wedge (gv:839.7) \wedge (bl:117.5) \wedge (r:1.761))$
$D_{res3} = ((np:705) \wedge (gv:663.2) \wedge (bl:123.0) \wedge (r:2.064))$
$D_{res4} = ((np:638) \wedge (gv:627.4) \wedge (bl:134.6) \wedge (r:2.435))$
$D_{res5} = ((np:705) \wedge (gv:696.4) \wedge (bl:122.7) \wedge (r:2.216))$

The system modified the networks and presented the below as potential class descriptions. Note that D_6 was retained for the *capacitor* class.

For the class *capacitor* –
$D_6 = ((np:X) \wedge (gv:Y) \wedge (bl:Z) \wedge (r:0.647\ or\ less))$,
For the class *transistor* –
$D_{10} = ((np:745\ or\ more) \wedge (gv:X) \wedge (bl:not\ (122.9\ to\ 209.2)) \wedge (r:Y))$
$D_{11} = ((np:X) \wedge (gv:not\ (502.5\ to\ 928.9)) \wedge (bl:Y) \wedge (r:1.105\ or\ more))$,
For the class *resistor* –
$D_{12} = ((np:705\ or\ less) \wedge (gv:X) \wedge (bl:not\ (105.9\ to\ 117.5)) \wedge (r:Y))$
$D_{13} = ((np:X) \wedge (gv:562.3\ to\ 839.7) \wedge (bl:Y) \wedge (r:Z))$
$D_{14} = ((np:X) \wedge (gv:562.3\ or\ more) \wedge (bl:not\ (122.7\ to\ 123.0)) \wedge (r:Y))$

These descriptions were then used to classify components seen in other images (seventeen capacitors, seventeen resistors, and eight transistors). The overall error rates for the above generalized class descriptions on new test data are:

Class	*False-positive*	*False-negative*
cap: D_6	4.0%	11.8%
trans: D_{10}	23.5%	50.0%
D_{11}	2.9%	25.0%
res: D_{12}	4.0%	0.0%
D_{13}	8.0%	17.6%
D_{14}	8.0%	11.8%

We would like to emphasize that none of the test data was utilized for learning. Note that the error rates shown were achieved after fewer than a half-dozen examples of each class were used for learning. Impressive though these results are, the system's performance would improve with increased learning opportunities.

References

Baim, P. W. 1988. A Method for Attribute Selection in Inductive Learning Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):888-896.

Bolles, R. C., and Horaud, P. 1986. 3DPO: A three dimensional part orientation system. *International Journal of Robotics Research*, 5(3):3-26.

Chen, C. H., and Kak, A. C. 1989. A robot vision system for recognizing 3-D objects in low-order polynomial time. *IEEE Transactions on Systems, Man, and Cybernetics* 19(6):1535-1563.

Dietterich, T. G., and Michalski, R. S. 1979. Learning and Generalization of Characteristic Descriptions: Evaluation Criteria and Comparative Review of Selected Methods. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 223-231. Tokyo: International Joint Conference on Artificial Intelligence, Inc.

Dietterich, T. G., and Michalski, R. S. 1981. Inductive Learning of Structural Descriptions: Evaluation Criteria and Comparative Review of Selected Methods. *Artificial Intelligence* 16(3):257-294.

Faugeras, O. D., and Hebert, M. 1986. The Representation, Recognition, and Location of 3-D Objects. *The International Journal of Robotics Research* 5(3):27-52.

Grimson, W. E. L., and Lozano-Perez, T. 1984. Model-Based Recognition and Localization from Sparse Range or Tactile Data. *The International Journal of Robotics Research*, 3(3):3-35.

Hansen, C. and Henderson, T. 1988. Toward the Automatic Generation of Recognition Strategies. In Proceedings of the Second International Conference on Computer Vision, 275-279. Tampa FL: IEEE Computer Society.

Hayes-Roth, F. and McDermott, J. 1977. Knowledge Acquisition from Structural Descriptions. In Proceedings of the 5th International Joint Conference on Artificial Intelligence, 356-362. Cambridge MA: International Joint Conference on Artificial Intelligence, Inc.

Hayes-Roth, F. and McDermott, J., 1978. An Interference Matching Technique for Inducing Abstractions. *Communications ACM*, 21(5):401-410.

Ikeuchi, K., and Kanade, T. 1988. Applying Sensor Models to Automatic Generation of Object Recognition Programs. In Proceedings of the Second International Conference on Computer Vision, 228-237. Tampa FL: IEEE Computer Society.

Kim, W. Y., and Kak, A. C. 1991. 3-D Object Recognition Using Bipartite Matching Embedded in Discrete Relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):224-251.

Michalski, R. S. 1980. Pattern Recognition as Rule-Guided Inductive Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2(4):349-361.

Oshima, M., and Shirai, Y. 1983. Object recognition using three-dimensional Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5(4):353-361.

Vere, S. A. 1975. Induction of Concepts in the Predicate Calculus. In Proceedings of the 4th International Joint Conference on Artificial Intelligence, 281-287. Tbilisi, USSR: International Joint Conference on Artificial Intelligence, Inc.

Vere, S. A. 1977. Induction of Relational Productions in the Presence of Background Information. In Proceedings of the 5th International Joint Conference on Artificial Intelligence, 349-355. Cambridge MA: International Joint Conference on Artificial Intelligence, Inc.