17

CONCEPTUAL CLUSTERING:

Inventing Goal-Oriented Classifications of Structured Objects

Robert E. Stepp III
University of Illinois at Urbana-Champaign

Ryszard S. Michalski*
Massachusetts Institute of Technology

Abstract

An important form of inductive learning is inventing a meaningful classification of given objects or events. This chapter extends the authors' previous work on this problem that was based on *conceptual clustering*, that is, grouping objects into conceptually simple classes. In contrast to the past work, the new method deals with classifying objects represented by structural descriptions rather than by sequences of attribute values. These descriptions are expressed in *Annotated Predicate Calculus* (APC), which is a typed predicate logic calculus with additional operators.

It is shown that in order to create a meaningful classification, a system must be equipped with *background knowledge*, which includes goals of classification, classification evaluation criteria, and deductive and inductive inference rules. The goals and goal-relevant descriptive concepts are organized into a *Goal Dependency Network* (GDN). Inference rules permit the system to derive high-level descriptive concepts such as functional and causal attributes from lower-level descriptive concepts provided initially. Example classifications created by the program CLUSTER/S and by people are presented.

---

*On leave of absence from the University of Illinois at Urbana-Champaign.

17.1 INTRODUCTION

Creating a classification[1] is typically the first step in developing a theory about a collection of observations or phenomena. This process is a form of learning from observation (learning without a teacher), and its goal is to structure given observations into a hierarchy of meaningful categories. The problem of automatically creating such a hierarchy has so far received little attention in AI. Yet creating classifications is a very basic and widely practiced intellectual process.

Past work on this problem was done mostly outside AI under the headings of numerical taxonomy and cluster analysis (Anderberg, 1973). Those methods are based on the application of a mathematical measure of similarity between objects, defined over a finite, a priori given set of object attributes. Classes of objects are taken as collections of objects with high intraclass and low interclass similarity. The methods assume that objects are characterized by sequences of attribute/value pairs and that this information is sufficient for creating a classification. The methods do not take into consideration any *background knowledge* about the semantic relationships among object attributes or global concepts that could be used for characterizing object configurations. Nor do they take into consideration possible goals of classification that might be indicated by background knowledge.

As a result, classifications obtained by traditional methods are often difficult to interpret conceptually. The problem of interpreting the results has remained a challenging task for the data analyst. In addition, traditional classification-building methods describe objects by attribute value sequences and therefore are inadequate for creating classifications of structured objects. The description of such objects must involve not only attributes of objects as a whole but also attributes of object components and relationships among these components.

This chapter describes a method for automated generation of classifications of structured objects through a process of *conceptual clustering*. This process generates classes (clusters of objects) by first generating conceptual descriptions of the classes and then classifying the objects according to these descriptions. The method is illustrated by a sample problem, and classifications produced by machine are compared to those produced by people.

17.2 THE GOAL OF THIS RESEARCH

The idea of conceptual clustering leads to an entirely new approach to the problem of creating classifications (Michalski, 1980a; Michalski and Stepp, 1983a, 1983b; Stepp, 1984). This idea states that objects should be arranged into classes that

---

[1]Creating or building a classification involves two subprocesses: (1) generating an appropriate set of categories and (2) classifying all given entities according to the generated categories.

represent simple concepts rather than classes defined solely by a predefined measure of similarity among their members.

In the earlier work on conceptual clustering, objects or events were described by attribute-value sequences. The method arranged the objects into a hierarchy of classes described by *conjunctive concepts.* These concepts are expressed as logical products of relations on selected object attributes. The generated sibling classes of any node in the hierarchy represented the most preferred (sub)classification from this node according to a given *preference criterion.* The background knowledge included the definitions of the attributes used in object descriptions, their domains and types, and the classification preference criterion.

This research extends the previous work in three ways:

- Objects and classes are described by structural descriptions, which are expressed in *Annotated Predicate Calculus* (APC), a typed predicate calculus with additional operators.
- The background knowledge includes inference rules for deriving high-level descriptive concepts from the low-level concepts initially provided.[2]
- The system is supplied with a general goal of the classification, which provides the means for identifying relevant descriptors and inference rules for deriving new descriptors. This avoids the necessity of defining them explicitly, as in the previous method.

An important aspect of this approach is the emphasis placed on the role of background knowledge for constructing meaningful and useful classifications. In this method the background knowledge consists of a network of goals of the classification, inference rules and heuristics for deriving new descriptors, definitions of attribute domains and types, and the classification preference criterion. The network of goals, called the *Goal Dependency Network* (GDN), is used for guiding the search for relevant descriptors and inference rules.

The necessity of using background knowledge in any form of inductive learning is indicated in the theory of inductive learning (Michalski, 1983). Important work involving background knowledge has been done by Winston (see chap. 3 of this volume), who describes an incremental learning process in which the background knowledge contains relevant precedents, exercises, and *unless conditions.* In chapter 19 DeJong presents a method of using background knowledge to acquire explanatory schemata that describe sequences of events presented as stories. Background knowledge has also been used by Mitchell and Keller (1983) to guide an inductive learning program for acquiring a problem-solving heuristics in integral calculus. In learning

[2]The descriptive concepts are called *descriptors* and include attributes, *n*-ary functions, and relations used to characterize objects or events.

by analogy, described by Burstein in chapter 13, a large body of causal knowledge is used to "fill out" incomplete descriptions and guide analogical inference. Carbonell (1983, and chap. 14 of this volume) developed a method for acquiring problem-solving strategies by analogy to solutions to similar problems. Rendell's Probabilistic Learning System demonstrated the usefulness of clustering points in the solution space for reducing the search required in problem solving (Rendell, 1983). As for the problem of learning structural descriptions from examples, various aspects of this problem are discussed in Winston (1984) and Dietterich and Michalski (1983).

To provide the necessary background, section 17.3 presents a brief overview of the authors' earlier method of attribute-based conjunctive conceptual clustering. Section 17.4 focuses on the role of background knowledge and goals in building classifications. Following that, section 17.5 presents a sample problem involving building a classification of structured objects. Finally, section 17.6 presents two methods for constructing classifications of structured objects that employ background knowledge.

17.3 ATTRIBUTE-BASED CONJUNCTIVE CONCEPTUAL CLUSTERING (PREVIOUS WORK)

This section briefly describes the authors' previous work on classifications using the method of *attribute-based conjunctive conceptual clustering* (AC^3), which is the sorting basis of the method presented here. The main idea behind AC^3 is that a configuration of objects forms a class only if it can be described by a conjunctive concept involving relations on object attributes. AC^3 is a special case of general conceptual clustering that generates a network of concepts to characterize a collection of objects. The problem posed in the framework of AC^3 is defined as follows:

Given: A set of objects (physical or abstract),
A set of attributes to be used to characterize the objects, and
A body of background knowledge, which includes the problem constraints, properties of attributes, inference rules for generating new attributes, and a criterion for evaluating the quality of candidate classifications;

Find: A hierarchy of object classes, and their descriptions in the form of conjunctive statements. Subclasses that are descendants of any parent class should have logically disjoint descriptions and maximize a *clustering preference criterion.*

As mentioned before, in conventional data analysis classes of objects are formulated solely on the basis of a measure of object similarity. The similarity between any two objects is characterized by a single number: the value of a similarity function

applied to symbolic descriptions of objects. These symbolic descriptions are vectors, whose components are scores on selected object attributes. Such measures of similarity are *context free;* that is, the similarity between any two objects A and B depends solely on the properties of the objects and is not influenced by any context (the *environment* surrounding the objects). Consequently, methods that use such measures are fundamentally unable to capture the *gestalt* properties of object clusters, that is, properties that characterize a cluster as a whole and are not derivable from properties of individual entities. In order to detect such properties, the system must be equipped with the ability to recognize configurations of objects representing certain global concepts.

This idea is the basis of conceptual clustering. Instead of similarity between objects, say, A and B, the method uses *conceptual cohesiveness* of A and B, which depends not only on those objects and surrounding objects E (the *environment*) but also on a set of concepts C that are available for describing A and B together. Thus, the conceptual cohesiveness between two objects A and B is a four-argument function $f(A,B,E,C)$ in contrast to an ordinary two-argument similarity function $f(A,B)$.

The conjunctive conceptual clustering method consists of two phases: a *clustering* phase and a *hierarchy-building* phase. The clustering phase arranges objects into classes using conceptual cohesiveness, so that the obtained clustering maximizes the given context-based clustering preference criterion. The hierarchy-building phase starts with building first-level conceptual classifications of all objects (at the *root* of the hierarchy). Then it recursively builds a classification for each sibling group of objects from the previous classification until the *stop growth* criterion is met.

The clustering phase algorithm works by alternately selecting a set of *seed* objects (one per class) and using the seeds to guide inductive inference over positive-only events to produce generalized, but mutually disjoint, descriptions of object classes. This process insures that each seed object is placed into a separate class. Each cluster description is as general as possible (various generalization transformations are exhaustively applied) so that it covers the given seed but no other seeds. Different seeds are used over several iterations while the clustering preference criterion is monitored. The algorithm halts when the clustering preference criterion does not improve for a dynamically determined number of iterations. The algorithm is described in detail in Michalski and Stepp (1983b).

17.4 THE USE OF BACKGROUND KNOWLEDGE AND GOALS

Suppose that we are observing a typical restaurant table on which there are such objects as food on a plate, a salad, utensils, salt and pepper, napkins, a vase with flowers, a coffee cup, and so on, as illustrated in figure 17-1. Suppose a person is

Figure 17-1: A typical restaurant table.

asked to build a meaningful classification of objects on the table. One way to create a classification is to perform the following chain of inferences:

- Salt and pepper are seasonings
  Seasonings are used to add zest to food
  Seasoned food is something to be eaten
  Things that are to be eaten are edible
  Salt and pepper are edible
- Salad is a vegetable
  Vegetables are food
  Food is something to be eaten
  Things that are to be eaten are edible
  Salad is edible

A similar chain of inferences applied to *meat on a plate* and *cake on a dessert plate* will also lead to the concept *is edible.* On the other hand, a napkin is not food and is therefore not edible. A vase containing flowers is not food and is therefore not edible. Consequently, one meaningful classification of objects on the table is simply *edible* versus *inedible.*

One may observe that when the background knowledge contains many such rules of inference, a large number of different but equally meaningful classifications can be created. The problem is then how to decide which of the classifications is best or most appropriate. For example, if inference rules about food types, suppliers, processing, and packaging were contained in the knowledge base, they could be used for generating other classifications. Some new classifications might produce categories such as *domestic* versus *imported* or *perishable* versus *nonperishable*. The problem of which classification to select can be resolved by assuming a general goal or purpose to be served by the classification. Assume, for example, that the classification is to be useful to an agent who wants to *survive*. Such a behavioral goal to *survive* dictates that a person has to ingest food and liquids, and be safe. Furthermore, the subgoal *ingest* can be linked to the two modes of ingestion, that is, consuming food and drinking liquids. In the context of the subordinate goals reached by links from the most general goal node, the relevant attributes are, for example, *is_edible*, and *is_potable*, *tastes_good*. The attribute *tastes_good* is linked by the implication relation to *is_edible* or *is_potable* (if something tastes good then it is either edible or potable).

Thus there is a general goal leading to subgoals and then to one or more attributes that are relevant in the context of the goal. Such relationships are captured in the Goal Dependency Network (GDN) mentioned earlier. This network links goals, subgoals, and relevant attributes together. Part of a hypothetical GDN headed by the *survive* goal is shown in figure 17-2. In the illustration, main goals are denoted by double

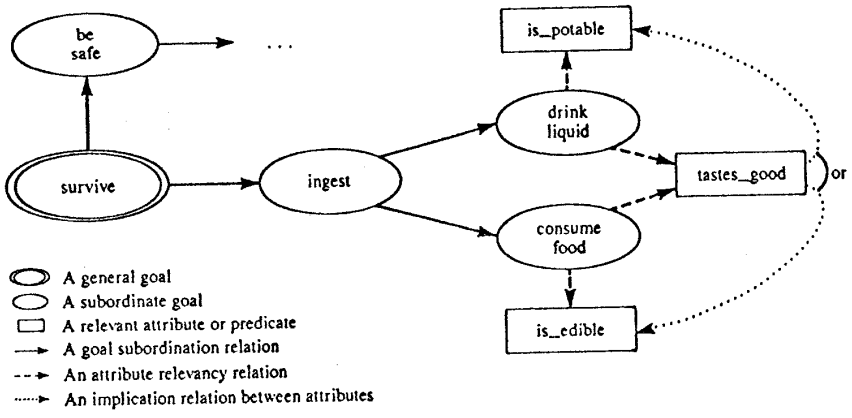


Figure 17-2: A GDN headed by the goal *survive*.

ellipses, and subgoals and relevant descriptors are denoted by regular ellipses and rectangles, respectively. The solid arcs between nodes are directed from goal nodes towards subordinate goal nodes. The dashed arcs between nodes and attributes are directed from goal nodes to relevant attribute nodes, and the dotted arcs link an attribute with an implied attribute.

Suppose that the goals of the agent include not only *survive* but also *be healthy and beautiful.* When both goals are involved, a GDN such as the one in figure 17-3 is used. Here, the links from the two top-level goals converge at the *consume dietary food* subgoal which links to the subordinate goals *consume lean foods* and *consume balanced diet.* Attached to these latter nodes are the relevant descriptors *fat content*

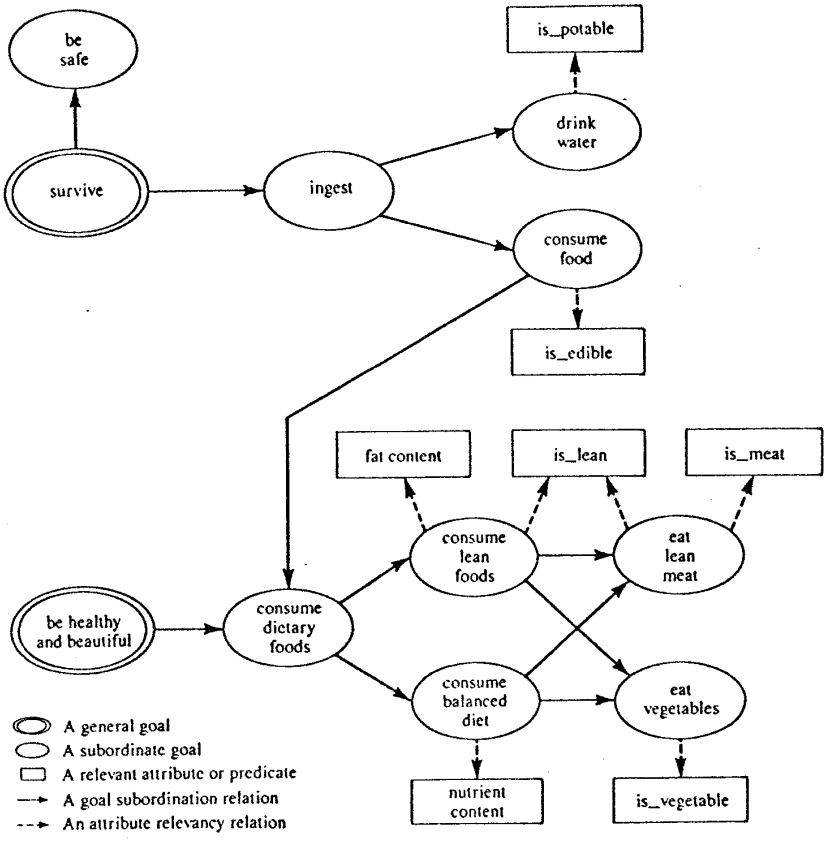


Figure 17-3: A GDN for the goals *survive* and *be healthy and beautiful.*

and *is_lean*, and *nutrient content*, respectively. The two subgoal nodes mentioned above have subordinate goal nodes of their own. These include *eat lean meat* and *eat vegetables*. The relevant descriptors attached to these nodes include the predicates *is_lean*, *is_meat*, and *is_vegetable*. Thus, by the addition of the top-level goal *be healthy and beautiful*, five additional relevant attributes are proposed by the GDN.

Adding a top-level goal may reduce the number of attributes thought to be relevant. Suppose we add a *vegetarian life-style* goal. Link paths from the three top-level goals converge at the subordinate goal *eat vegetables*. This increases the relevancy of the *is_vegetable* predicate which now dominates in relevancy over the other attributes. The GDN for this last situation is illustrated in figure 17-4.

Let us now consider a specific problem: the system is given symbolic descriptions of objects on the table in terms of their physical attributes (including structure) along with *survive* as a general goal of classification, and we want it to create the classification into edible versus inedible objects. Notice first that creating such a classification solely on the basis of original attributes is practically impossible, because objects that are in the same functional class (edible or inedible) can be vastly different in terms of their physical properties (see Winston, 1984, for a discussion of this problem). A program that could classify objects on a table as edible or inedible would have to be equipped with background knowledge consisting of the previously described inference rules and with the ability to use them in a goal-directed way.

Background knowledge built into the program can be divided into *general purpose* and *domain specific*. General-purpose knowledge consists of fundamental constraints and criteria specifying general properties of classifications. This includes a specification of the domain of each descriptor, the type of the domain (unordered, linearly ordered, or tree-structure ordered), and a sequence of elementary criteria to be applied lexicographically with tolerances to evaluate classifications. The *Lexicographical Evaluation Functional with tolerances*, or LEF (see section 17.6.2), is used to select from among candidate classification schemes the one that is the most preferred viewpoint of the given goal.

Domain-specific background knowledge consists of inference rules for deriving values for new descriptors and GDN to infer which descriptors (attributes, functions, or predicates) are relevant to the goal of classification.

Event descriptors can be divided into *initial descriptors* and *derived descriptors*. Both kinds of descriptors can appear attached to goal nodes in the GDN. The initial descriptors can be divided into those that are relevant with respect to the goals and those that are irrelevant. In some problems, the relevant descriptors are unknown and not necessarily provided as initial descriptors. A solution can still be obtained in such cases if background knowledge can be used to derive relevant descriptors from those that are initially given. Inference rules in the knowledge base are used to infer the values of the derived descriptors. Domain-specific knowledge in the GDN is used to guide the application of inference rules toward descriptors that are likely to be relevant and thus worth the computational cost of their derivation.

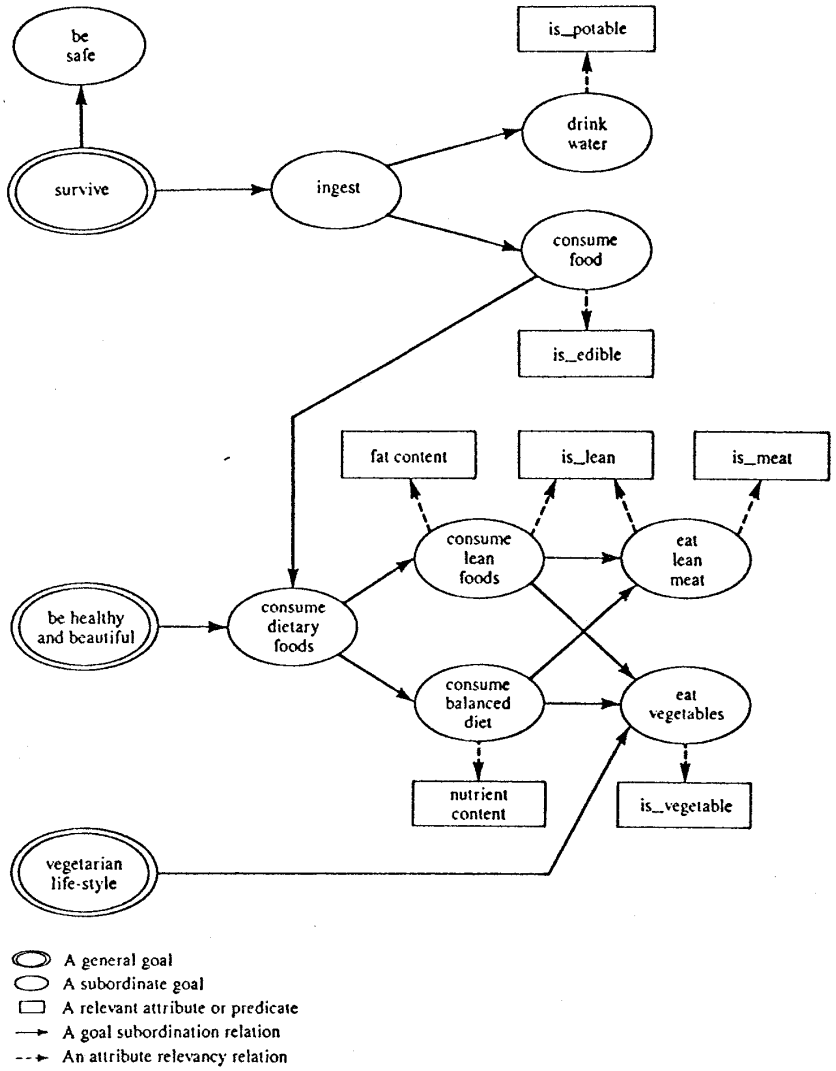


Figure 17-4: A GDN for the goals *survive*, *be healthy and beautiful*, and *vegetarian life-style*.

Derived descriptors can be divided into two categories:

- *Descriptors derived by logical inference.* These descriptors are predicates and functions obtained by the application of general and problem-specific inference rules to the initial descriptions of the objects. In this work, inference rules consist of a condition part and a consequence part. Whenever an object description matches the condition portion of a rule, the consequence portion is applied to the object description. The consequence may be composed either of new predicates and functions to be asserted or of arithmetic expressions that are evaluated. In either case, the new descriptors (unless already present) are appended to the object description and become available as attributes that are potentially relevant for building classifications of the objects.
- *Descriptors derived by special computations, experiments, or devices.* These descriptors are obtained from the initial descriptors by the application of specialized descriptor generation procedures, by running experiments, or by activating some external device, that is, any procedure other than the application of condition-consequence rules. Examples of such descriptors generated by the INDUCE/2 program (Hoff, Michalski, and Stepp, 1983) are "the number of object subparts," "the number of subparts with some specific property," "the number of different values observed for an attribute," and "properties common to all subparts." The program can also automatically generate multiplace predicates to assert "same function value" for several parts—for example, samecolor($p1$,$p2$)—and single-place predicates to assert head and tail positions in a chain of properties—for example, to assert most-ontop($p1$) and least-ontop($p1$) when given ontop($p1$,$p2$) and ontop($p2$,$p3$).

17.5 BUILDING CLASSIFICATIONS OF STRUCTURED OBJECTS

Let us turn now to the problem of classifying structured objects. Consider, for example, the problem of classifying trains,[3] shown in figure 17-5. The trains are structured objects, each consisting of a sequence of cars of different shapes and sizes. The individual cars carry a variable number of items of different shapes. The problem presented is in a class of learning problems known as *learning from observation*, or *concept formation*. It is interesting to both AI researchers and cognitive psychologists.

---

[3]This example is a reformulation of a problem known as "East- and Westbound Trains" (Michalski and Larson, 1977). In the original formulation, two collections of trains were given, those that were eastbound (A to E) and those that were westbound (F to J); the problem was to learn a simple rule for distinguishing between the eastbound and the westbound trains. Thus the original problem was that of *learning from examples*, or *concept acquisition*.

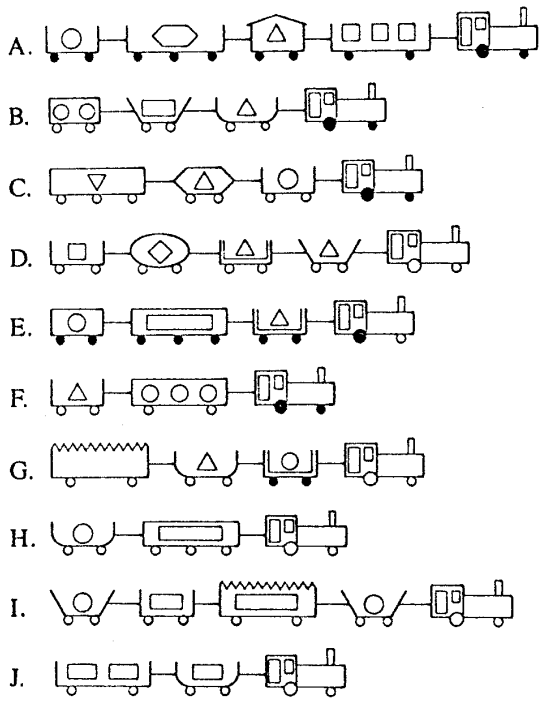


Figure 17-5: How would you classify these trains?

Human classifications of the trains shown in figure 17-5 have been investigated by Medin, Wattenmaker, and Michalski (1985). The ten trains were placed on separate index cards so they could be arranged into groups by the subjects in the experiment. Each subject was instructed to partition the trains according to three methods and to state the rationale used:

1. Arrange the trains into any number of groups.
2. Arrange the trains into two equal groups.
3. Arrange the trains into any number of groups of conceptually similar objects plus an "other" group to hold any unusual or hard-to-classify trains.

The experiment was completed by thirty-one subjects who made a total of ninety-three classification schemes for partitioning the objects. The most popular basis for classification (seventeen repetitions) was the number of cars in the trains (a simple attribute that characterizes each train as a whole). The three clusters formed were the following: "trains containing two cars," "trains containing three cars," and "trains

containing four cars." The second most frequent classification (seven repetitions) was based on the engine wheel color. These two classifications are shown in figure 17-6. Of the ninety-three classifications produced, forty of them were unique. Thus, although there was no explicit goal for classification given, there was a pattern of uniformity among the subjects. The pattern was not a very strong one, however, as witnessed by a wide spectrum of singleton solutions.

This problem is an example of a class of problems for which the implicit classification goal is to generate classes that are conceptually simple and based on easy-to-determine visual attributes. When people are asked to build such classifications, they typically form classes with *disjoint* descriptions, as in the study by Medin. People typically do not suggest intersecting classifications, and it is for this reason that we focus on methods that produce disjoint descriptions.

Classification problems such as this one occur when one wants to organize and classify observations that require structural descriptions—for example, when one wants to classify physical or chemical structures, analyze genetic sequences, build taxonomies of plants or animals, characterize visual scenes, or split a sequence of temporal events into episodes with simple meanings. As an example of the latter problem, consider splitting a kidnapping story into episodes such as kidnapping, bargaining, and exchange (DeJong, 1981).

One problem of concern here is to develop a general method that when applied to the collection of structured objects, such as trains, could potentially generate the conjunctive concepts occurring in human classifications or invent new concepts having similar appeal. We first assume that there is only a very general goal for a classification, such as *simplicity of descriptions of categories* or *good fit of the categories to the examples*. The method should be able to generate conceptual categories that can be described by a conjunction of predicates. These conjunctions should represent a minimal overgeneralization of observed events in the class so as to insure a good "fit" between each class description and the events.

Figure 17-7 shows a hypothetical GDN for a classification for which the general goal is to find simple visual patterns. A subordinate goal is to look for simple geometrical regularities in object descriptions. For the trains problem, this goal node leads to the relevant variables such as *number of cars, color of wheels, number of wheels, number of items carried,* and so on. The *simple geometrical regularities* goal links to the two subordinate goals *shape of components* and *similarity of components*. The first of these subgoals leads to relevant attributes involving shape (*cargo shape, engine shape, car shape*). The second subgoal leads to a variety of relevant attributes relating one component of a train to other components. The *number of different shapes* attribute gives the count of the different car shapes in a train. A count of the number of different cargo shapes in a car would be another attribute of this same type. The *same car shape* or *same color of wheels* attributes are predicates of two or more variables that denote the equality of feature values across several components in the train. If all components have the same value for some attribute, then a *forall*

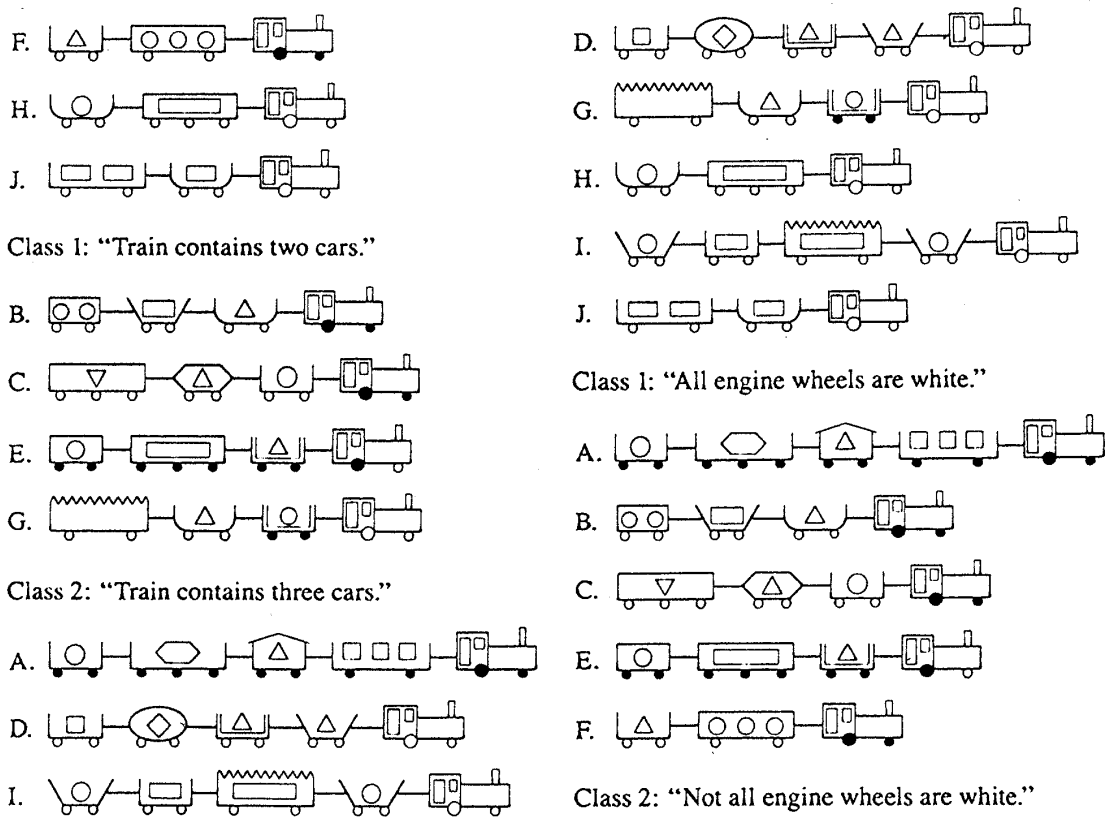


Figure 17-6: The two most popular classifications produced by people.

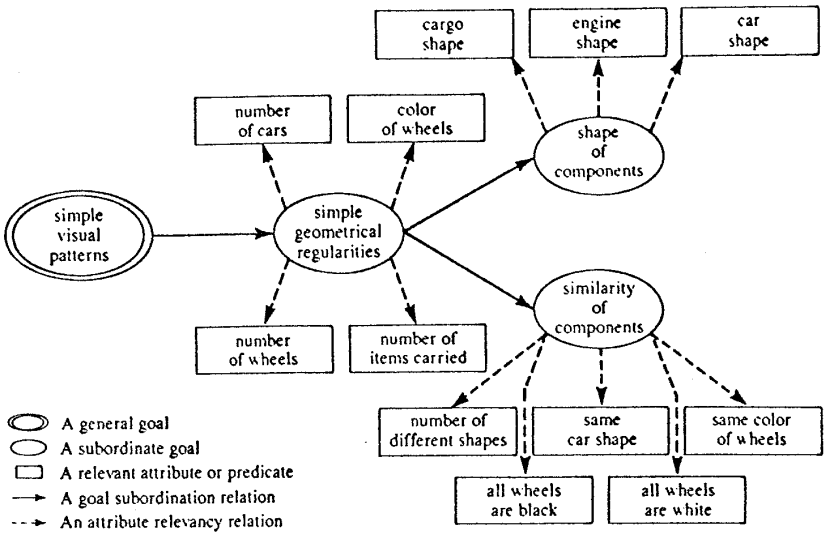


Figure 17-7: A GDN for the goal of finding simple visual patterns.

predicate, such as *all wheels in the train are black*, is a relevant attribute for describing the situation.

As examples of solutions obtained by the program CLUSTER/S implementing the method, figure 17-8 shows two classifications created for the trains problem. For this problem, the structured descriptions of each train involve the descriptors *contains, infront, car shape, number of wheels, wheel color, cargo shape,* and *number of items carried.* The program determined several new descriptors that were not in the initial descriptions, such as *number of different shapes, same-shape* predicates, *same-color-of-wheels* predicates and so on.

The generated attribute vectors were processed using a classification evaluation criterion that attempts to minimize the number of attributes used in a description, maximize the number of attributes that singly discriminate among all classes, and maximize the number of attributes that take different values in different classes. Minimizing the number of attributes used tends to conflict with the other two elementary criteria. This was handled by specifying a high tolerance (90 percent) for the first elementary criterion and zero tolerances for the second and third elementary criteria in the LEF evaluation criterion described in section 17.6.2.

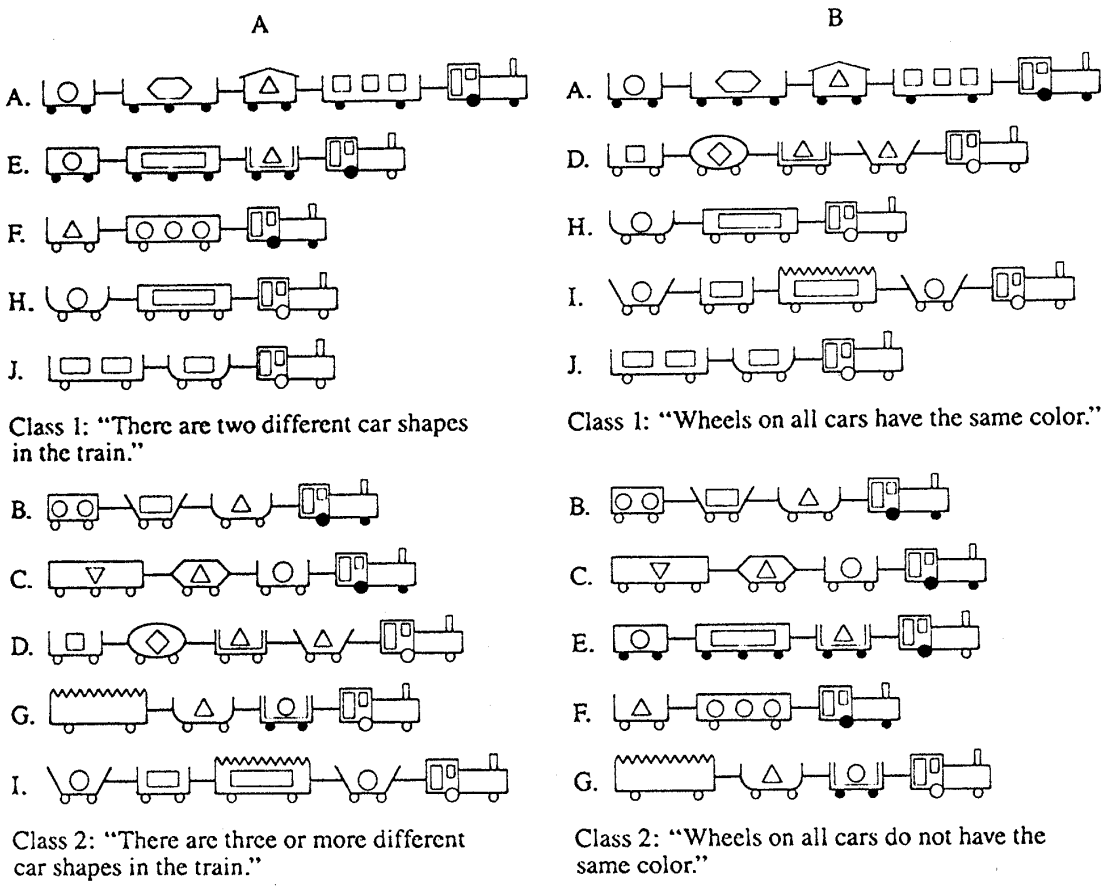


Figure 17-8: Two simple classifications found by the program CLUSTER/S.

Classification A in figure 17-8 was generated by the program with two different sets of class descriptions. The top class ("There are two different car shapes in the train") was also described as "The third car from the engine (if it exists) has black wheels." The bottom class ("There are three or more different car shapes in the train") was also described as "The third car from the engine exists and has white wheels." Classification B in figure 17-8 is based on the derived predicate *samecolor.* Both classifications received the same evaluation criterion score and were considered to be alternative classifications. Solutions of the kind shown in figure 17-8 are appealing because the differences between classes is striking yet not obvious from casual inspection.

17.6 TWO METHODS FOR BUILDING CLASSIFICATIONS

This section describes two methods for solving problems of the kind posed in the preceding section, that is, building a classification of a collection of structured objects. One method is called RD, which stands for *repeated discrimination,* and the other is called CA, which stands for *classifying attributes.* The RD method is based on the authors' previous work and reduces the problem of building a classification into a sequence of concept acquisition problems, specifically, problems of determining discriminant descriptions of objects with given class labels (Michalski and Stepp, 1983b). The CA method is based on generating candidate *classifying attributes* either from the initially given pool of attributes or from derived attributes generated with the aid of inference rules and the Goal Dependency Network.

The two methods are similar in that they both use the same representation language (APC) for describing objects, classes of objects, and general and problem-specific background knowledge. Both methods use the LEF as the general-purpose criterion for measuring the *quality* of generated candidate solutions. The APC and LEF are described in the next two sections, respectively.

17.6.1 The Description Language: Annotated Predicate Calculus

The Annotated Predicate Calculus (APC) is an extension of predicate calculus that uses several novel forms and attaches an *annotation* to each predicate, variable, and function (Michalski, 1983). The annotation is a store of information about the given predicate or atomic function, such as the type and structure of its legal value set, related (more general or more specific) descriptors in descriptor hierarchy, and other information. In addition to all the forms found in predicate calculus, the language also uses a special kind of predicate called a *selector.* A simple selector is in the form

[atomic-function REL value-of-atomic-function]

where REL (relation) stands for one of the symbols $= \neq < > \leq \geq$. An example of such a selector is

[weight(box) > 2kg]

which means "the weight of the box is greater than 2 kg." A more complex selector may involve *internal disjunction* or *internal conjunction.* These two operators apply to terms rather than to predicates and are illustrated by the two corresponding examples:

[color(box) = red & purple]	"The color of the box is either red *or* purple."
[color(box1 & box2) = red]	"The color of box 1 *and* box 2 is red."

The meaning of the internal disjunction operator is defined by

$$[f(x) = a \ \& \ b] \Rightarrow [f(x) = a] \ \& \ [f(x) = b]$$

and the meaning of the internal conjunction operator is defined by

$$[f(x \ \& \ y) = a] \Rightarrow [f(x) = a] \ \& \ [f(y) = a].$$

Selectors can be combined by standard logical operators to form more complex expressions. Background knowledge is expressed as a set of APC implicative rules:

CONDITION ⇒ CONSEQUENCE

where CONDITION and CONSEQUENCE are conjunctions of selectors. Thus a rule in APC is more general than the Horn clause used in PROLOG. If CONDITION is satisfied, the CONSEQUENCE is asserted. To understand the implicative statement, consider the assertion "vegetables are food" from the example in section 17.4. It can be expressed in APC by the following statement, which says, "if an object is a vegetable then it is also a food":

[is_vegetable(object)] ⇒ [is_food(object)]

An alternative way to express this idea in APC is

[object-type(object) = vegetable] ⇒ [object-type(object) = food]

which says, "if the type category of an object is *vegetable* than the type category is also *food.*" In this latter statement *vegetable* and *food* are treated as elements of the structured domain of the attribute *object-type.* This implication expresses a generalizing inference rule called *climbing the generalization tree.* Further details on the APC language are given in Michalski (1983).

17.6.2 Directing the Process by Measuring Classification Quality

Creating a classification is a difficult problem because there are usually many potential solutions with no clearly correct or incorrect answers. This proliferation of answers was seen in the experiment with human classification building presented in section 17.5. The decision about which classification to choose can be based on some perceived set of goals (Medin, Wattenmaker, and Michalski, 1985), a goal-oriented, statistic-based utility function (Rendell, 1983), or some measure of the *quality* of the classification.

One way to measure classification quality that has been successful in both INDUCE/2 and CLUSTER/2 is to define various elementary, easy-to-measure criteria specifying desirable properties of a classification and to assemble them together into one general criterion, called the *Lexicographical Evaluation Functional with tolerances* (LEF) (Michalski, 1980b). Each elementary criterion measures a certain aspect of the generated classifications. Examples of elementary criteria are the relevance of descriptors used in the class descriptions to the general goal, the fit between the classification and the objects, the simplicity of the class descriptions, the number of attributes that singly discriminate among all classes, and the number of attributes necessary to classify the objects into the proposed classes (Michalski and Stepp, 1983b).

The LEF consists of an ordered sequence of elementary criteria along with tolerances that control to what extent different solutions are considered equivalent. First, all classifications are evaluated according to the first elementary criterion. Those that score best or within the given tolerance range from the best are retained. Those retained are then evaluated according to the next elementary criterion, and so on, until either a single classification remains or the list of elementary criteria in the LEF is exhausted. In the latter case, all classifications that remain are judged equal and the algorithm picks one arbitrarily. To control combinatorial explosion, the LEF is also applied during the search process that generates classifications. The LEF provides a powerful heuristic for searching the huge space of hypothetical classifications to find a classification that optimizes several criteria at once.

17.6.3 Using Background Knowledge

Building a meaningful classification relies on finding good classifying attributes (high-level attributes used to define classes). For example, the attribute *is_edible* discussed in section 17.4 is such a high-level classifying attribute. The repeated discrimination and classifying attributes methods, described in sections 17.6.4 and 17.6.5, both use background knowledge in the search for such attributes. The Goal Dependency Network is traversed to find the interactions between the classification goal(s) and the potential descriptors. Background knowledge rules enable the system to perform a chain of inferences to derive values for new descriptors for inclusion in object descriptions. The new descriptors are tested to determine if they make good

classifying attributes by applying the LEF to the classification defined by the classifying attribute.

As described in section 17.4, the background knowledge rules can represent both the built-in general-purpose knowledge and the domain-specific knowledge provided by the data analyst. In the latter case, knowledge for generating inferentially derived descriptors is supplied in the form of an inference rule (called a background rule, or *b-rule*). Special types of b-rules include expressions of arithmetic relationships (*a*-rules), such as

$$\forall \text{ object, girth(object)} = \text{length(object)} + \text{width(object)}$$

and implicative rules that specify logical relationships (*l*-rules), such as:

$$\forall p1,p2,p3, [\text{above}(p1,p2)][\text{above}(p2,p3)] \Rightarrow [\text{above}(p_1,p_3)]$$

or

$$\forall p_1,p_2,p_3, [\text{mother}(p_1,p_2) \,\&\, ([\text{mother}(p_2,p_3)] \vee [\text{father}(p_2,p_3)]) \Leftrightarrow [\text{grandmother}(p_1,p_3)].$$

Each rule is associated with a condition defining the situations to which it is applicable.

17.6.4 Concept Formation by Repeated Discrimination: Method RD

This section explains how a problem of concept formation (here, building a classification) can be solved via a sequence of controlled steps of concept acquisition (learning concepts from examples). We start with a brief description of the program INDUCE/2, which solves concept acquisition tasks involving structured objects.

Given a set of events (by which we mean symbolic descriptions of objects or situations) arranged into two or more classes, INDUCE/2 induces a general description of each class in the form of an annotated predicate calculus expression. First, all events are divided into two sets: set $F1$ of events belonging to the class currently being considered and set $F0$ of events belonging to any other class (counterexamples to set $F1$). One event at a time is selected from set $F1$ (the *seed* event) and a *star* is built that *covers* the seed event *against* all events in set $F0$. The star is the set of all alternative most general descriptions that describe the seed event (and possibly other events from $F1$) and no events from $F0$ (Michalski, 1983; Michalski and Stepp, 1983b).

To control combinatorial explosion, INDUCE/2 determines *bounded* stars rather than complete stars. A bounded star contains only a fixed number of descriptions selected as most promising according to LEF. The highest-rank description in the bounded star is chosen as a part of the solution. The events covered by the resulting description are removed from set $F1$. If any events remain in $F1$, another

seed event (from among those not yet covered) is selected and the whole process is repeated. When all events in the set $F1$ have been covered, the solution is complete: it is the disjunction of the descriptions selected in each iteration.

This algorithm for concept acquisition can be adapted for solving classification construction problems. Given a set of unclassified objects, k seed objects are selected randomly and treated as individual representatives of k imaginary classes. The algorithm then generates descriptions of each seed that are maximally general and do not cover any other seed. These descriptions are then used to determine the most representative object in each newly formed class (defined as the set of objects satisfying the class description). The representative objects are used as new seeds for the next iteration. The process stops either when consecutive iterations converge to some stable solution or when a specific number of iterations pass without improving the classification (from the viewpoint of the criterion LEF).

This approach requires the selection of a defined number of representative objects (corresponding to the number of classes). Since the best number of classes to form is usually unknown, two techniques are used: (1) varying the number of classes and (2) composing the classes hierarchically.

Since the classification to be formed should be simple and easy to understand, the number of classes that stem from any node of the classification hierarchy was assumed to be in the range of two to seven. Since this range is small, it is computationally feasible to repeat the whole process for every number in this range. The solution that optimizes the score on the LEF (with appropriate adjustment for the effect of the number of classes on the score) indicates the best number of classes to form at this level of the hierarchy.

The above idea of repeated discrimination for performing concept acquisition has been implemented in the program CLUSTER/2 for a subset of annotated predicate calculus involving only attributes (zero-argument functions). Besides its relative computational simplicity, this approach has other advantages stemming from descriptions (for both objects and classes) that are quantifier free. Specifically, it should be noted that classifications normally have the property that they can unambiguously classify any object into its corresponding class. To have this property, the class descriptions must be mutually disjoint.

For conjunctive descriptions involving relations on attribute/value pairs, the disjointness property is easy to test and easy to maintain. For the larger subset of APC involving existentially quantified variables, predicates on these variables, and function/value relationships over quantified variables, the test for mutual disjointness of descriptions and the maintenance of disjointness are difficult. As a result of this difficulty, the approach taken for concept acquisition from structured objects involves two processing steps. The first step, using algorithms of INDUCE/2, finds an optimized characteristic generalization of the entire collection of events and then applies it to generate a quantifier-free description of each object (a vector of attribute values). The second step processes the quantifier-free object descriptions with the

CLUSTER/2 algorithm to form optimized classifications. These two processes are combined in the program CLUSTER/S.

A characteristic generalization expresses a common substructure in all structured objects that facilitates the binding of a subset of the free variables (representing object parts) to specific parts. That portion of the structure of each object that is described by the characteristic generalization is called the *core* of each object. With corresponding parts identified in all objects, the cores may be described by a vector of attribute values. Thus the descriptions of object cores need neither quantified variables nor multiplace predicates in their descriptions (i.e., such descriptions can be handled by the CLUSTER/2 program).

It is recognized that structural differences between objects would be lost by the above approach since it focuses on the *common* substructure found in all given objects. To retain some unique structural features of individual objects, an inspection is made of the connections between object subparts within the core and object subparts outside the core. New predicates are automatically generated and added to object descriptions to denote the attachment of different kinds of additional structures to the core of each object.

The descriptions of each substructure connected to the cores of objects are collected and classified by recursive application of the conjunctive conceptual clustering procedure. The resulting types of substructures are given labels (e.g., a unique class number) which are used in the generated predicates that show *what* kind of additional structure is attached *where* to the core structure. The final object descriptions contain attributes for core parts and predicates denoting the kind of attached substructures, as well as derived descriptors for both core subparts and the object as a whole. After this transformation, objects are describable (with reduced detail) by attribute vectors.

The following extension of the trains problem will further illustrate the use of a GDN and problem-specific background knowledge. Suppose that the knowledge base includes an inference rule that can identify trains carrying toxic chemicals. Suppose also that the general goal *survive* has a subordinate goal to *monitor dangerous shipments.* The additional background knowledge can be used to help build a classification.

In the illustrations of the trains a toxic chemical container will be identified as a single sphere (circle) riding in an open-top car. The logical inference rule (*l*-rule) supplied to CLUSTER/S is

[contains(trains,car)][car-shape(car) = opentop]
[cargo-shape(car) = circle][items-carried(car) = 1]
⇔ [has_toxic_chemicals(train)]

In the above rule, equivalence is used to indicate that the negation of the condition part is sufficient to assert the negative of the consequence part. After this rule is applied, all trains will have descriptions containing either the toxic-chemical

predicate or its negation. The characteristic description generated by CLUSTER/S will now contain the additional predicate *has_toxic_chemicals(train)* or its negation.

In the GDN we find the main goal *survive* and a chain of subordinate goals beginning with *be safe* and *monitor dangerous shipments.* Two additional subgoals are *monitor chemicals shipments* and *monitor toxic chemicals shipments.* Attached to these nodes are relevant attributes such as *is_explosive, is_radioactive, is_flammable, is_corrosive, has_toxic_chemicals*, and so on. The GDN is illustrated in figure 17.9. The GDN signals the relevancy of these descriptors to the goal *survive.* Assuming that this goal takes precedence over the goal *find simple visual patterns,* classifications that make use of the *has_toxic_chemicals* descriptor in formulating conceptual classes score higher than those that use descriptors. The classification produced in this case is shown in figure 17-10.

17.6.5 Concept Formation by Finding Classifying Attributes: Method CA

This section describes another approach for building classifications called *classifying attributes* (briefly, CA). This approach attempts to find one or more *classifying* attributes whose value sets can be split into ranges that define individual classes. The important aspect of this approach is that the classifying attribute can be derived through a goal-directed chain of inferences from the initial attributes. The

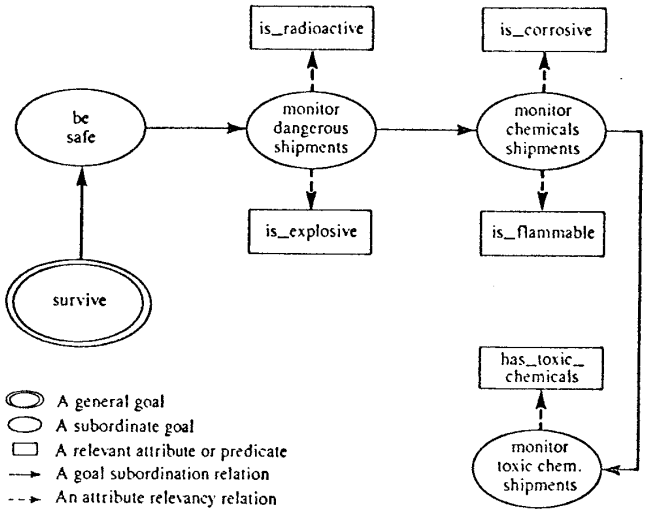
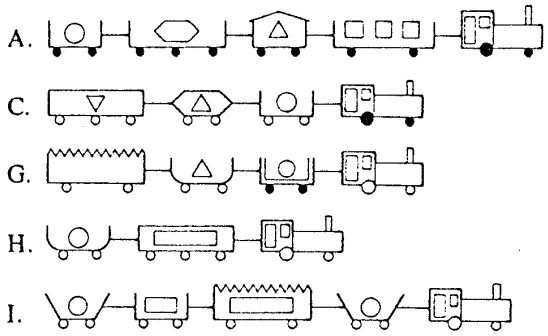
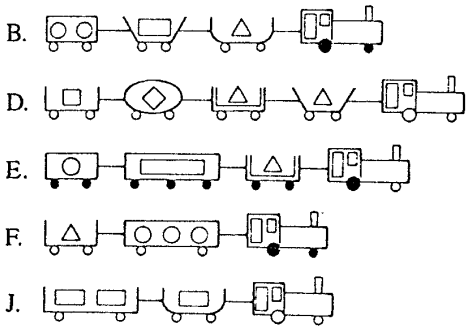


Figure 17-9: A hypothetical GDN for dangerous train shipments.



"These trains are carrying toxic chemicals."



"These trains are not carrying toxic chemicals."

Figure 17-10: A classification produced using the *toxic chemicals* inference rule.

classifying attributes sought are the ones that lead to classes of objects that are best according to the classification goal.

The promise of a descriptor to serve as a classifying attribute is determined by consulting the GDN and by considering how many other descriptors it implies. For example, if the goal of the classification described in section 17.4 is *finding food*, the attribute *edibility* might be a classifying attribute. The second way of determining the promise of an attribute can be illustrated by the problem of classifying birds. The question of whether *color* is a more important classifying attribute than *is_waterbird* is answered in favor of *is_waterbird*, because the latter implicatively leads to more implied attributes than does the attribute *color* in a given GDN network (e.g.,

is_waterbird implies *can_swim, has_webbed_feet, eats_fish,* and so on) (Medin, 1982).

There are two fundamental processes that operate alternately to generate the classification. The first process SEARCH searches for the classifying attribute whose value set can be partitioned to form classes such that the produced classification scores best according to the LEF. The second process GENERATE generates new descriptors by a chain of inferences using two forms of background knowledge rules: logical implicative rules (l-rules) and arithmetic rules (a-rules). Descriptors that can be inferred are ordered by relevancy indicated by the GDN and the goals of the classification.

SEARCH can be performed in two ways. When the number of classes to form (k) is known in advance, the process searches for attributes having k or more different values in the descriptions of the objects to be classified. These values are called the *observed values* of the attribute. Attributes with the number of observed values smaller than k are not considered. For attributes with observed value sets larger than k, the choice of the mapping of value subsets to classes depends on the resulting LEF score for the classification produced and the type of the value set. When the number of classes to form is not known, the above technique is performed for a range of values of k. The best number of classes is indicated by the classification that is best according to the LEF.

GENERATE constructs new attributes from combinations of existing attributes. Certain heuristics of attribute construction are used to guide the process. For example, two attributes that have linearly ordered value sets can be combined using arithmetic operators. When the attributes have numerical values (as opposed to symbolic values such as *small, medium,* and *large*), a trend analysis can be used to suggest appropriate arithmetic operators, as in the BACON system (see chap. 16). Predicates can be combined by logical operators to form new attributes through l-rules. For example, a rule that says an animal is a reptile if it is cold-blooded and lays eggs can be written in APC as

$$[\text{cold-blooded}(a1)][\text{offspring birth}(a1) = \text{egg}] \Rightarrow [\text{animal-type}(a1) = \text{reptile}].$$

The application of this rule to the given animal descriptions yields the new attribute *animal-type* with the specified value *reptile.* Using this rule and similar ones, one might classify some animals into reptiles, mammals, and birds even though the type of each animal is not stated in the original data.

17.7 SUMMARY

This chapter has discussed the problem of building classifications of structured objects using goal-directed inferences from background knowledge. Two methods

for performing this task were described. The first method, RD (repeated discrimination), transforms concept formation into a sequence of concept acquisition tasks. The second method, CA (classifying attributes), forms classes by generating new descriptors using a chain of inferences and testing them as candidate classifying criteria. The criterion selected is the one that partitions the set of events in the way most preferred according to a Lexicographical Evaluation Functional (LEF).

The classifying attributes are generated with the aid of a Goal Dependency Network, which relates goals to subgoals and to relevant attributes. The ability to incorporate domain-specific background knowledge in the form of inference rules and Goal Dependency Networks adds a new dimension to the process of concept formation and data analysis.

This work could be further extended through the investigation of alternative representations for describing classes in a classification. These could include the use of logical operators such as implication, equivalence, and exception in a class description. The exception operator appears to be especially interesting because of its frequent use by people. Exception clauses in logical rules can be introduced by using *unless* conditions to handle the cases that are not frequent or ordinary (see chap. 3).

Another extension of this work could be the development of a system capable of characterizing a collection of observations (facts, events, and so on) not just by a hierarchy of concepts but by a *concept network*, in which nodes represent conceptual classes and links represent various relations among them. In the kind of hierarchy considered here, any two generated concepts are related by the relation *is a generalization of* or *is a specialization of* or *is a disjoint of*. In a concept network (a form of semantic network) a much larger set of relations would be allowed.

ACKNOWLEDGMENTS

This research was done in part at the Department of Computer Science Artificial Intelligence Laboratory at the University of Illinois and in part at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. Support for the University of Illinois Laboratory is provided in part by grants from the National Science Foundation under grant No. NSF DCR 84-06801 and the Office of Naval Research under grant No. N00014-82-K-0185. Support for the Massachusetts Institute of Technology Laboratory is provided in part by the Advanced Research Projects Agency of the U.S. Department of Defense under the Office of Naval Research contract number N00014-80-C-0505.

The authors wish to thank Tom Mitchell at Rutgers University and Larry Rendell at the University of Illinois for remarks and criticisms on earlier versions of the manuscript. They also thank Peter Andreae at the Massachusetts Institute of Technology Artificial Intelligence Laboratory and Doug Medin at the University of Illinois Department of Psychology for useful discussions and valuable comments.

References

Anderberg, M. R., *Cluster Analysis for Applications*, Academic Press, New York, 1973.

Burstein, M. H., "Concept Formation by Incremental Analogical Reasoning and Debugging," chap. 13 of this volume, 1985.

Carbonell, J. G., "Derivational Analogy in Problem Solving and Knowledge Acquisition," *Proceedings of the International Machine Learning Workshop*, R. S. Michalski (Ed.), Allerton House, University of Illinois at Urbana-Champaign, pp. 12-18, June 22-24, 1983. (An updated version of this paper appears as chap. 14 of this volume.)

DeJong, G., "Generalizations Based on Explanations," *Proceedings of the Seventh IJCAI*, Vancouver, B.C., pp. 67-69, 1981.

———, "An Approach to Learning from Observation," chap. 19 of this volume, 1986.

Dietterich, T. G., and Michalski, R. S., "A Comparative Review of Selected Methods for Learning from Examples," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Tioga, Palo Alto, Calif., 1983.

Hoff, W., Michalski, R. S., and Stepp, R., "INDUCE/2: A Program for Learning Structural Descriptions from Examples," Technical Report No. UIUCDCS-F-83-904, Department of Computer Science, University of Illinois at Urbana-Champaign, 1983.

Langley, P.; Zytkow, J.; Simon, H. A.; and Bradshaw, G. L., "The Search for Regularity: Four Aspects of Scientific Discovery," chap. 16 of this volume, 1986.

Lingle, J. H., Altom, M. W., and Medin, D. L., "Of Cabbages and Kings: Assessing the Extendibility of Natural Object Concept Models to Social Things," in *Handbook on Social Cognition*, R. Wyer, T. Srull, and J. Hortwick (Eds.), Erlbaum, Hillsdale, N.J., 1983.

Medin, D. L., "Structural Principles in Categorization," in *The Developments of Perception and Cognition*, T. Tighe, B. Shepp, H. Pick (Eds.), Erlbaum, Hillsdale, N.J., 1982.

Medin, D. L., Wattenmaker, W. S., and Michalski, R. S., "Constraints in Inductive Learning: An Experimental Study Comparing Human and Machine Performance," submitted to *Cognitive Science*, 1985.

Michalski, R. S., "Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts," *Policy Analysis and Information Systems*, Vol. 4, No. 3, pp. 219-44, 1980a.

———, "Pattern Recognition as Rule-Guided Inductive Inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 4., pp. 349-61, July 1980b.

———, "A Theory and Methodology of Inductive Learning," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Tioga, Palo Alto, Calif., 1983.

Michalski, R. S., and Larson, J. B., "Inductive Inference of VL Decision Rules," Paper presented at Workshop in Pattern-Directed Inference Systems, Hawaii, May 1977. (Published in *SIGART Newsletter*, ACM, No. 63, pp. 38-44, June 1977.)

Michalski, R. S., and Stepp, R. E., "Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 4, pp. 396–410, July 1983a.

———, "Learning from Observation: Conceptual Clustering," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Tioga, Palo Alto, Calif., 1983b.

Mitchell, T. M., and Keller, R. M., "Goal Directed Learning," *Proceedings of the 2nd International Machine Learning Workshop*, R. S. Michalski (Ed.), Allerton House, University of Illinois at Urbana-Champaign, June 22–24, 1983.

Rendell, L. A., "Toward a Unified Approach for Conceptual Knowledge Acquisition," *AI Magazine*, Winter 1983.

Stepp, R. E., "Conjunctive Conceptual Clustering: A Methodology and Experimentation," Ph.D. diss., Department of Computer Science, University of Illinois at Urbana-Champaign, 1984.

Winston, P. H., *Artificial Intelligence*, Addison-Wesley, Reading, Mass., 1984.

———, "Learning by Augmenting Rules and Accumulating Censors," chap. 3 of this volume, 1985.