CONCEPTUAL CLUSTERING: INVENTING GOAL-ORIENTED CLASSIFICATION OF STRUCTURED OBJECTS

by

Robert Stepp
Ryszard S. Michalski

Chapter in the book, "Machine Learning: An Artificial Intelligence Approach," Volume II, Morgan-Kaufman Publishers, Palo Alto, California, 1985.

Chapter in the book, MACHINE LEARNING: An Artificial Intelligence Approach, R. S. Michalski, T.Mitchell, and J. Carbonell (Eds.), TIOGA Publishing Co., Palo Alto, 1983.

For Machine Learning, Book II

Artificial Intelligence Laboratory
Department of Computer Science
University of Illinois

CONCEPTUAL CLUSTERING:
Inventing Goal-Oriented Classifications of Structured Objects

Robert E. Stepp
University of Illinois
at Urbana-Champaign

Ryszard S. Michalski
Massachusetts Institute
of Technology[1]

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 based on *conceptual clustering*, i.e., 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 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 various inference rules. The goals and goal-related descriptive concepts are organized into a *Goal Interaction Network* (GIN). 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.

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

1. Introduction

Creating a classification[2] is typically the first step in understanding and formulating 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 classifications 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 intra-class and low inter-class similarity, according to the given measure. 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 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 as might be indicated by background knowledge.

As the result, classifications obtained by traditional methods are often difficult to interpret conceptually. The problem of interpreting the results has remained a challenging task of 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. One form of the method is illustrated by applying it to a sample problem for which the

2. Creating or building a classification involves two subprocesses: generating an appropriate set of categories or conceptual classes, and then classifying all given entities according to the generated categories.

classifications produced by machine are similar to those produced by people.

2. The Goal of This Research

The idea of conceptual clustering leads to an entirely new approach to the problem of creating classifications [Michalski, 1980b; Michalski and Stepp, 1983a, 1982b; Stepp, 1984]. It is based on the assumption that objects should be arranged into classes that represent simple concepts rather than classes based solely on a predefined measure of similarity.

In the earlier work on conceptual clustering, objects or events are described by attribute-value sequences. The method arranges the objects into a hierarchy of classes described by conjunctive concepts. These concepts are described by logical products of relations on selected attributes. The generated sibling classes of any node in the hierarchy represent the most preferred (sub)classification from this node according to a given preference criterion. The background knowledge includes 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 directions:

- Objects and classes are described by structural descriptions, which are expressed in *Annotated Predicate Calculus* (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,[3]
- A general goal of the classification provides the means for identifying relevant descriptors and inference rules for deriving new descriptors. (This avoids the necessity for giving them a priori as in the previous method).

An important aspect of our approach is the stress we put on the role of *background knowledge* for constructing meaningful and useful classifications. In this method the background knowledge consists of goals of the classification, inference rules and heuristics for deriving new descriptors, definitions of attribute domains and types, and the classification preference criterion. A network of goals called the *Goal Interaction Network* (GIN) is used for guiding the search for relevant descriptors and inference

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

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 Chapter 3 in this book) who describes an incremental learning process in which the background knowledge contains relevant precedents, exercises, and *unless conditions* (which he calls "censors"). DeJong in Chapter 19 shows an interesting way of using background knowledge to acquire explanatory schemas for describing sequences of events presented as stories. Background knowledge also has been used by Mitchell and Keller [1983] to guide an inductive learning program for learning problem-solving heuristics in integral calculus. In a related form of learning (learning by analogy) described by Burstein in Chapter 13, a large body of causal knowledge has been used to "fill out" incomplete descriptions and guide analogical inference. Carbonell [1983] has discussed a method for acquiring problem solving strategies by analogy to solutions of similar problems. Also, Rendell's Probabalistic Learning System demonstrates the usefulness of clustering points in the solution space according to localized penetrance scores in order to reduce the amount of search required in problem solving [Rendell, 1983]. Various aspects of learning structural descriptions from examples are given in [Winston, 1984] and [Dietterich and Michalski, 1983].

To provide the necessary background, Sec. 3 presents a brief overview of our earlier method of attribute-based conjunctive conceptual clustering. Sec. 4 focuses on the role of background knowledge and goals in building classifications. Following that, Sec. 5 presents a sample problem involving building a classification of structured objects. Finally, Sec. 6 presents two methods for constructing classifications involving structured objects and using background knowledge.

3. Attribute-based Conjunctive Conceptual Clustering (Previous work)

This section briefly describes our earlier work on automatic construction of classifications using the method of *attribute-based conjunctive conceptual clustering* (AC^3). 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 selected object attributes. AC^3 is a special case of *conceptual clustering* in which collections

of objects are described by a form of semantic network (network of concepts). 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,
- A body of background knowledge, which includes the problem constraints, goals, 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, in which each class is described by a single conjunctive concept. Subclasses that are descendants of any parent class should have logically disjoint descriptions and optimize an assumed criterion (a *clustering quality criterion*).

As mentioned before, classes of objects are formulated in conventional data analysis on the basis of a measure of 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, i.e., 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 that correspond to certain concepts.

This idea is the basis of conceptual clustering. The *conceptual similarity* between two objects A and B, which we shall call the *conceptual cohesiveness* of A and B, depends not only on those objects and surrounding objects E (the *environment*), but also on a set of concepts C which 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 similarity function of two arguments f(A,B).

The conjunctive conceptual clustering method can be described by dividing the algorithm into two parts: a *clustering algorithm* and a *hierarchy-building algorithm*. The clustering algorithm arranges objects into a specified number of classes on the basis of conceptual cohesiveness such that the whole

clustering optimizes the given context-based clustering quality criterion. The hierarchy-building algorithm first builds a conceptual classification for all objects (at the *root* of the hierarchy). Then it recursively builds a conceptual classification for each sibling group of objects from the previous classification until the *stop growth* criterion is met [Michalski and Stepp, 1983b]. For each needed conceptual classification, the clustering algorithm is applied iteratively for a range of number of clusters (classes). The classification which scores best (according to a context-based clustering quality criterion) is selected and put into the developing hierarchy.

The clustering algorithm works by alternately selecting a set of *seed* objects (one per class) and then 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 (exhaustively applying various generalization transformations) such that it covers only one unique seed object (but many non-seed objects) and does not intersect with any other cluster description. The algorithm is described in detail in [Michalski and Stepp, 1983b]. Different seed objects are used over several iterations while the score obtained by applying the clustering quality criterion is monitored for improvement. The algorithm halts when the clustering quality criterion does not improve for a dynamically determined number of iterations.

4. The Use of Background Knowledge and Goals

It can be observed that when people create a classification of some objects, they consider ways of describing configurations of objects using various concepts and relationships from background knowledge that are relevant to the problem, measured by how well they contribute towards fulfilling the given classification goal. Let us consider a simple example. 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, etc., as illustrated in Fig. 1. Suppose a person is asked to build a meaningful classification of the objects on the table. One way to create a classification is to perform the following chains of inferences:

Figure 1. A typical restaurant table.

- salt and pepper are seasonings
  seasonings are used to add zest to food
  seasoned food is something to be eaten
  things which 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 which are to be eaten are edible
  salad is edible

A similar chain of inferences applied to "meat on a plate" or "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" vs. "non-edible".

One may observe that when the background knowledge base has many such rules of inference, many 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, processing, and packaging were available for the analysis of the items on a restaurant table, they could be used in the generation of additional classifications. One new classification might involve the two categories "perishable" and "non-perishable". The problem of whether to select a classification based on "edible" or one based on "perishable" can be resolved by assuming that the system is equipped with a general goal or goals for the classification. Consider, for example, a case where the general goal is to *survive*. Among other things, this goal dictates that a person has to ingest food and water and be safe. Furthermore, the subgoal *ingest* can be linked to the two modes of ingestion, i.e., consuming food and drinking water. In the context of the subordinate goals reached by links from the most general goal node, the important attributes are *is_edible*, and *is_potable*.

Thus, we have 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 a *Goal Interaction Network* (GIN). This network links goals, subgoals, and relevant attributes together. Part of a hypothetical GIN headed by the *survive* goal is shown in Fig. 2. In the illustration, main goals are denoted by double ellipses while 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.

Suppose that the goals for classification include not only "survive", but also "be healthy and beautiful". When both goals are involved, a GIN such as the one in Fig. 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* 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 GIN.

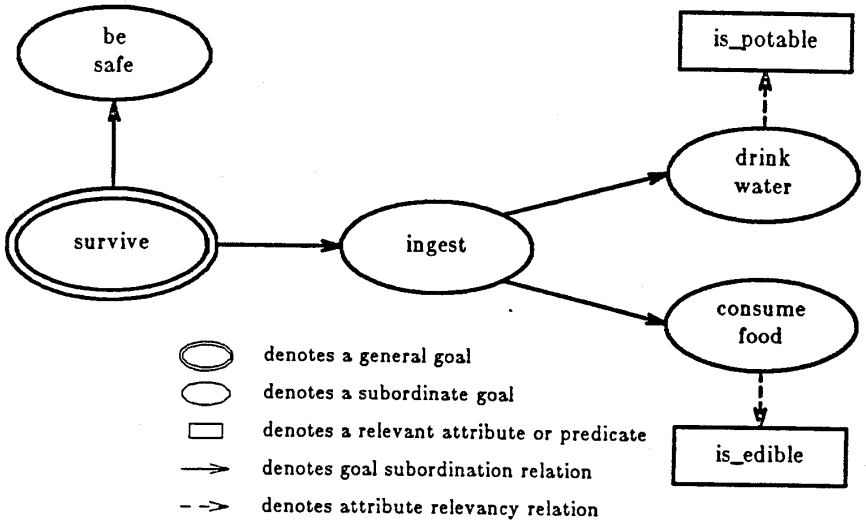

Figure 2. A Goal Interaction Network headed by the goal to survive.

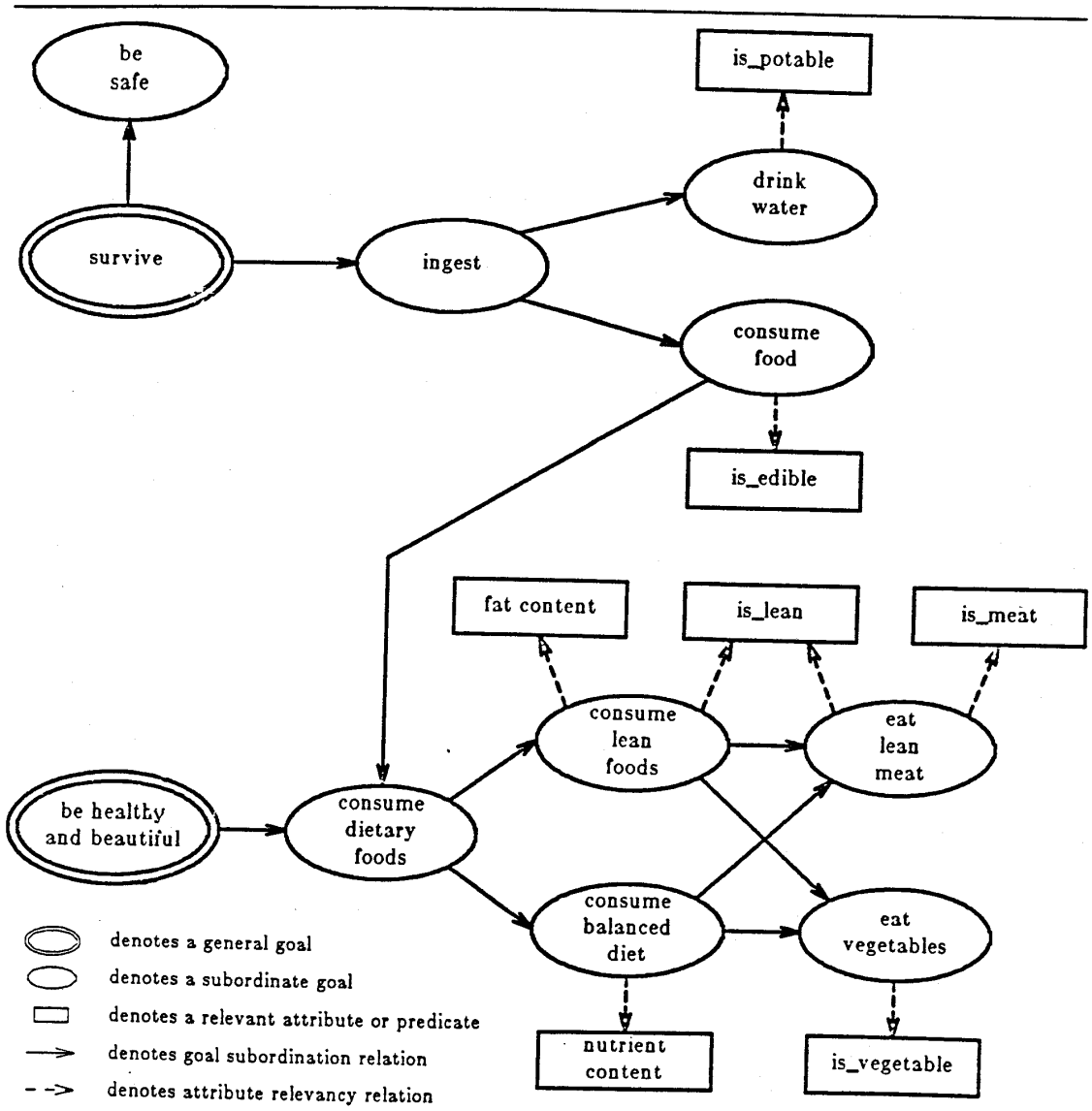

Figure 3. A GIN for the goals *survive* and *be healthy and beautiful.*

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 GIN for this last situation is illustrated in Fig. 4.

The first problem that we pose is this: suppose we are given descriptions of objects on the table in terms of their attributes (including structure) along with a general goal of the classification and we want to have the program create (on the basis of this information) the above classification into edible vs. inedible objects using the is_edible predicate.

Such a classification based solely on original attributes is quite unlikely, because objects that are in the same functional class (e.g., edible vs. inedible) can be vastly different in terms of their physical properties [Winston, 1984]. A program which could classify objects on a table as edible or inedible would have to be equipped with background knowledge consisting of the above inference rules and with the ability to use them in a goal-directed way.

Before we explain how to solve this problem, we first discuss the concept of background knowledge. Background knowledge can be divided into the categories: *general-purpose* and *domain-specific*. General-purpose background knowledge consists of fundamental constraints and criteria specifying desired properties of classifications. This includes definitions of the number of values in the value set of each descriptor, its scale of measurement (nominal, linearly ordered, or tree-structure ordered), and a sequence of elementary criteria to be applied lexicographically with tolerances to evaluate classifications (see *Lexicographical Evaluation Functional with tolerances* (LEF) in Sec. 6.2). The LEF is used to help select from among candidate classification schemes one classification that is appropriate for the problem at hand and which directs the algorithm towards solutions which meet a given goal.

Domain-specific background knowledge consists of inference rules for deriving values for new descriptors and a GIN to help infer which descriptors (initial or derived) are relevant to the given goal(s) of the classification. The GIN is used to suggest descriptors (attributes, functions, or predicates) that are likely to be relevant.

Event descriptors can be divided into *initial descriptors* and *derived descriptors*. Both kinds of descriptors can appear attached to goal nodes in the GIN. 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

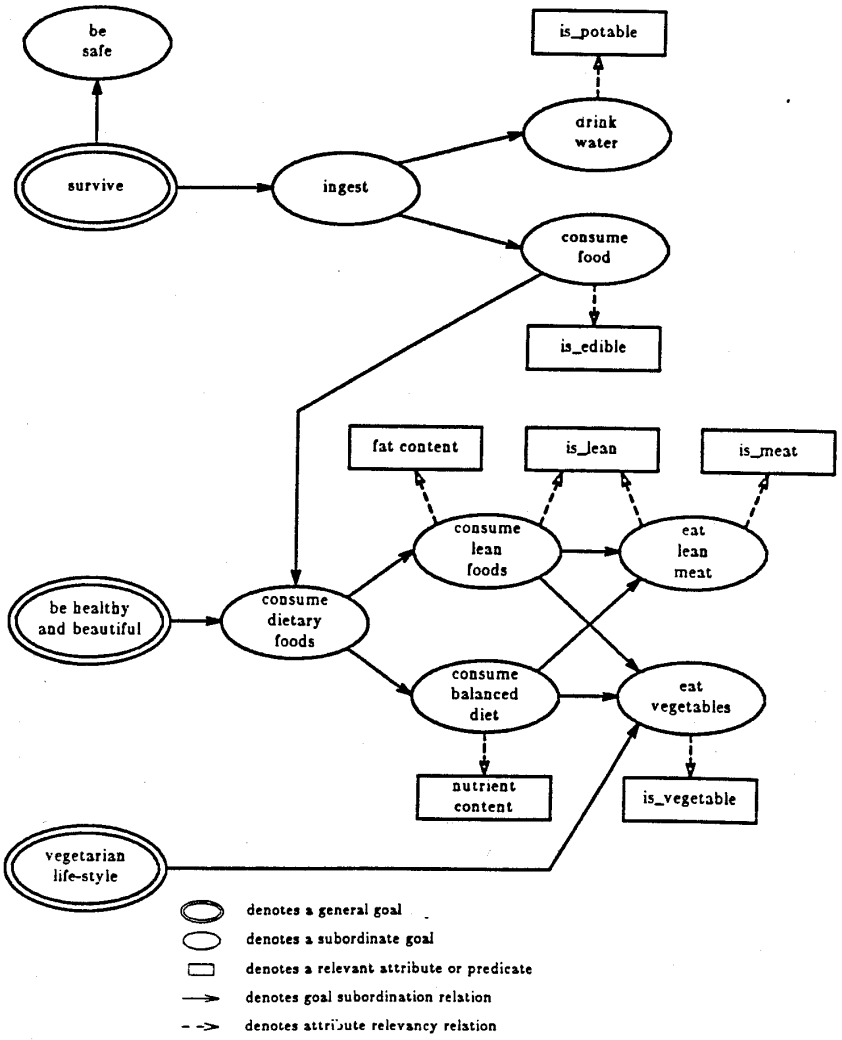

Figure 4. A GIN for the goals *survive*, *be healthy and beautiful*, and *vegetarian life-style*.

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 GIN is used to guide the application of inference rules towards descriptors that are likely to be relevant and thus worth the computational cost of descriptor value derivation for all events.

When considering derived descriptors, it is possible to distinguish two categories, as given below.

- 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 our 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 of new predicates and functions to be asserted or arithmetic expressions which are evaluated. In both cases, the new descriptors (unless already present) are appended to the object description and become available as attributes which 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, i.e., any procedure other than the application of ordinary 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 multi-place predicates to assert "same function value" for several parts (e.g., samecolor(p1,p2)), and single-place predicates to assert head and tail positions in a chain of properties, e.g., to assert most-ontop(p1) and least-ontop(p3) when given ontop(p1,p2) and ontop(p2,p3).

5. Building Classifications of Structured Objects

In order to explain the problem of building a classification of structured objects, let us consider the simple example of finding a classification of some trains[4] shown in Fig. 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

4. This example is based on rephrasing the problem known as "East- and West-bound trains" [Michalski and Larson, 1977]. In the original formulation, given were two collections of trains, those that are "East-bound" (A to E) and those that are "West-bound" (F to J), and the problem was to learn a simple descriptive rule to distinguish between the East-bound and West-bound trains. The original formulation is a problem of the type known as *learning from examples* or *concept acquisition*.

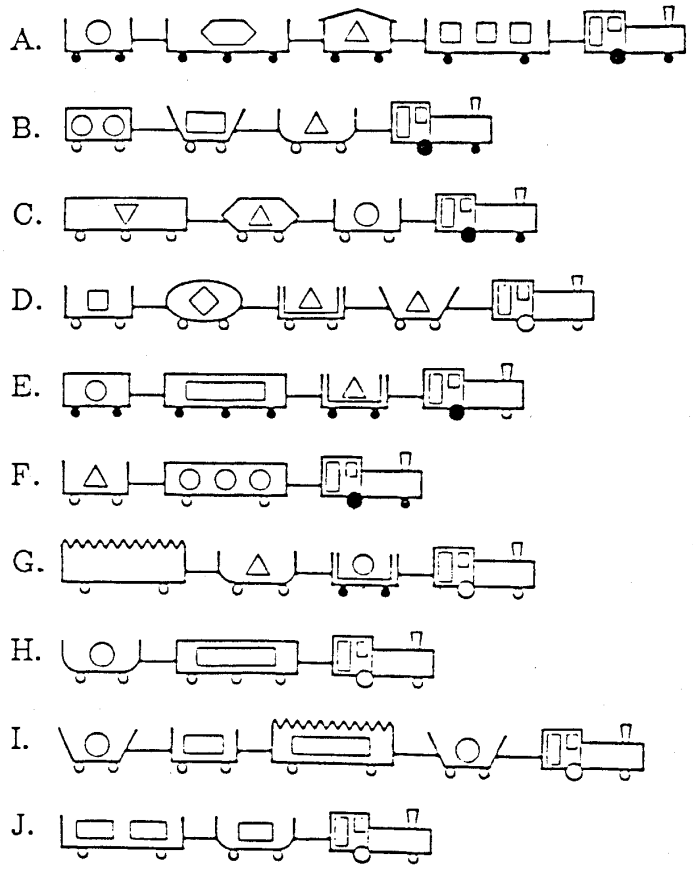

Figure 5. How would you classify these trains?

known as *learning from observation* or *concept formation*. It is interesting to both AI researchers and cognitive psychologists.

Human classifications of the trains shown in Fig. 5 have been investigated by Medin [Medin et al., 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 in the following ways and to state the rationale used.

1: Arrange the trains into any number of groups of conceptually similar objects.

2: Arrange the trains into two equal groups of similar objects.

3: Arrange the trains into any number of groups of conceptually similar objects plus an "other" category to hold any unusual or hard to classify trains.

The experiment was completed by 31 subjects who made a total of 93 classification schemes for partitioning the objects. The most popular basis for classification (17 repetitions) was the number of cars in the trains (a simple attribute that characterizes each train as a whole). The three clusters formed were: trains containing 2 cars, trains containing 3 cars, and trains containing 4 cars, respectively. The second most frequent classification of the trains was based on engine wheel color and occurred only 7 times. These two classifications are shown in Fig. 6. Of the 93 classifications produced, 40 of them were generated by only a single subject.

Although one cannot generalize from a single experiment, the results suggest that in the absence of explicit goals for a classification, there exists a strong pattern of uniformity in the form of a classification scheme and a wide spectrum of singleton solutions. It is possible that human classifications are generated to satisfy criteria that are acquired from experience which differs somewhat from person to person.

We now turn to classifications of the trains that were generated by machine. This problem is an example of a class of problems for which the 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 generally do not suggest intersecting descriptions and it is for this reason that we focus on methods which produce disjoint descriptions.

Classification problems such as this one occur when one wants to organize and classify observations that require structural descriptions, for example 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].

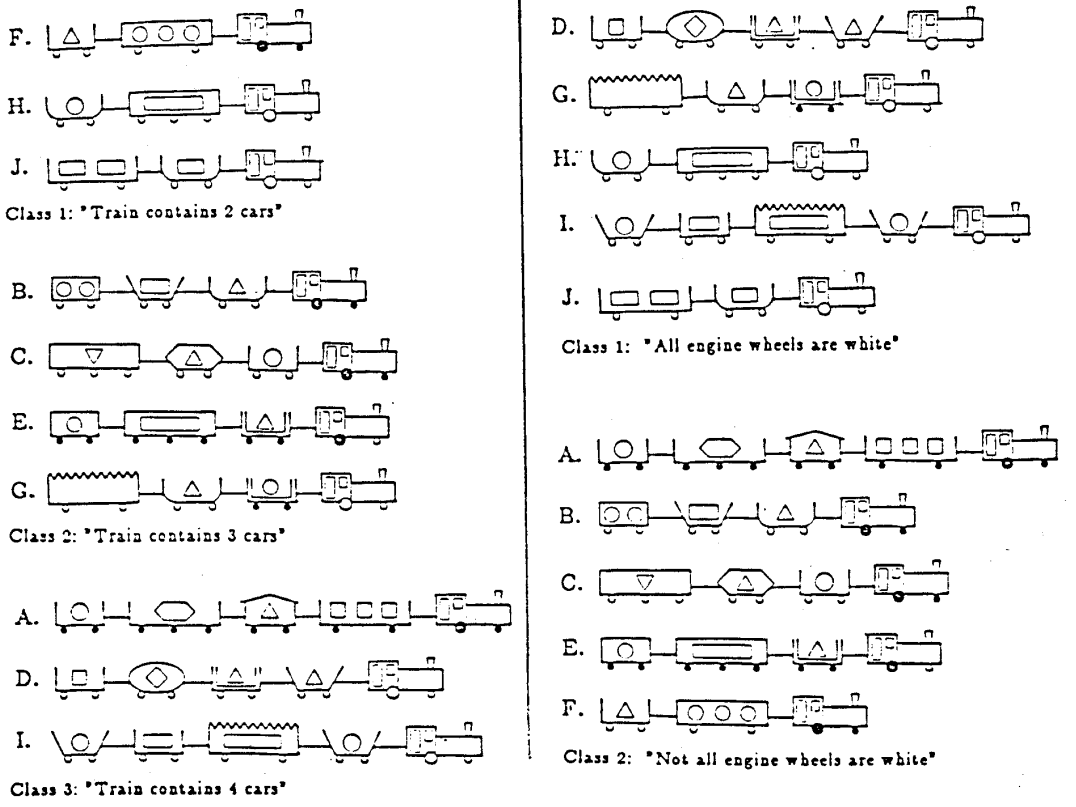

Figure 6. The two most popular classifications produced by people.

The problem of concern here is to develop a general method that when applied to the collection of trains could potentially generate the conjunctive concepts contained in human classifications or invent new concepts having similar appeal. In this kind of problem, there is only a general goal of the classification, such as *simplicity* or *good fit of the categories in the classification to the examples.* The method should be able to generate conceptual descriptions in which each class is described by a simple conjunction of predicates and attribute/value relations on object subparts with minimal overgeneralization in order to have good fit between each class description and the events in the class.

Fig. 7 shows a hypothetical GIN for a classification for which the general goal is to find simple visual patterns. A subordinate goal is to look for simple geometrical patterns in object descriptions. For the trains problem, this goal node leads to the relevant variables *number of cars, color of wheels, number of wheels, number of items carried,* etc. The *simple geometrical patterns* 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*

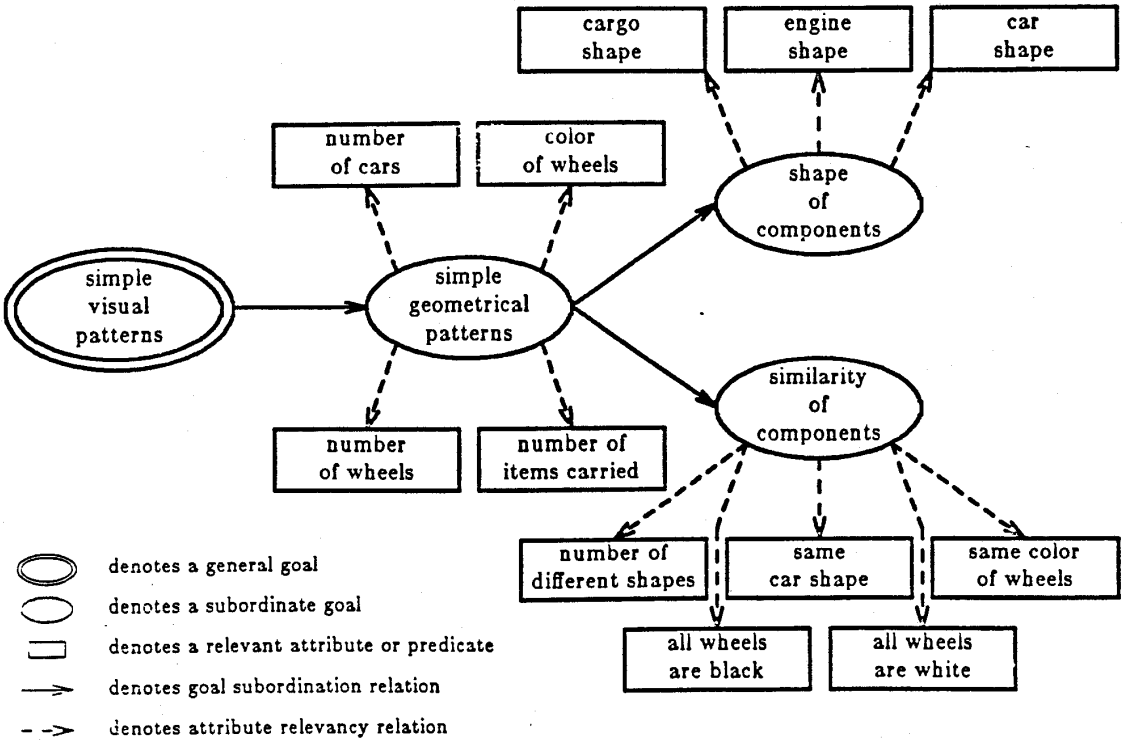

Figure 7. A GIN for the goal of finding simple visual patterns.

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* and *same color of wheels* attributes are predicates of two or more variables which 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* predicate such as *all wheels in the train are black* is a relevant attribute for describing the situation.

To give examples of solutions obtained by the method, we show in Fig. 8 two classifications created by this approach 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 GIN described above is used to help determine relevant variables. Some of the variables not in the initial descriptions (e.g., counting different shapes, same-shape predicates, same-color-of-wheels predicates) are derived by the program.

The generated attribute vectors were processed using a classification evaluation criterion which minimizes the number of attributes used in a description, maximizes the number of attributes which singly discriminate between all classes, and maximizes the number of attributes which take different values in pairs of 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%) for the first criterion and zero tolerances for the second and third criteria using the *LEF* described below.

Classification A in Fig. 8 was generated by the program with two different sets of class descriptions. The top class ("there are 2 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 3 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 Fig. 8 is based on the derived predicate *samecolor.* Both classifications received the same evaluation criterion score and were considered to be alternative classifications. We find solutions of the kind shown in Fig. 8 appealing because the difference between classes is striking, yet not obvious by casual inspection.

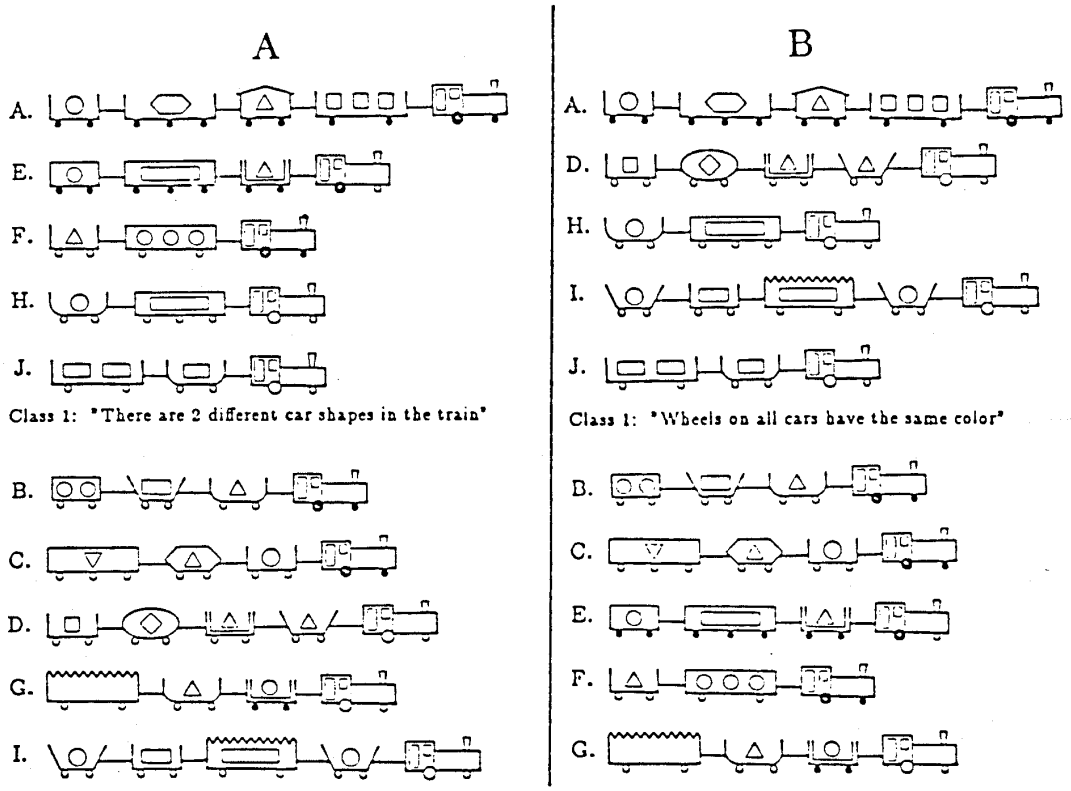

Figure 8. Two sample classifications found by method RD.

6. Two Methods for Building Classifications

Given that an algorithm for building classifications of objects that are representable by vectors of attribute/value pairs has been described and implemented (Conjunctive Conceptual Clustering [Michalski and Stepp, 1983b]), this section addresses the problems of extending that method in two directions:

(1) how to formulate and utilize background (or context) knowledge in creating classifications, and

(2) how to form classifications of structured objects.

This section describes two methods for solving problems of the kind posed in the preceding section, i.e., building a classification of an *unlabelled* collection of trains. One method is called RD for *Repeated Discrimination* and the other is called CA for a method using *Classifying Attributes*. The RD method is based on our 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).

The CA method is based on generating candidate *classifying attributes* either from the initially given pool of attributes or from algorithmically or inferentially derived attributes generated with the aid of inference rules and GIN provided by background knowledge.

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.

Before we describe the methods, we first discuss the description language used and then a criterion for measuring the quality of proposed classifications.

6.1. The Description Language: Annotated Predicate Calculus

In order to build a classification of structured objects, we must have an adequate language for describing such objects as well as classes of objects. Semantic networks or predicate calculus could provide an adequate representation scheme. Here we have chosen a language called *Annotated Predicate Calculus (APC)* [Michalski, 1983]. APC is an extension of predicate calculus that uses several novel forms and attaches to each predicate, variable, and function an *annotation*. The annotation is a store of information about the given predicate or atomic function such as the type and structure of its value set, related (more general or more specific) descriptors in some descriptor hierarchy, etc. Together with all

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 box1 *and* box 2 is red."

The meaning of the internal disjunction operator is defined by

$$[f(x)=a \vee b] \Longleftrightarrow [f(x)=a] \vee [f(x)=b])$$

and the meaning of the internal conjunction operator is defined by

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

Selectors can be combined by standard logical operators to form more complex expressions. For example, the descriptions of two classes[5] of the trains shown in Fig. 5 are written in APC as follows:

Class 1 (A-E): ∀(train) ∃(car) [contains(train,car)][length(car)=short][shape(car)=closed]
("A train is in Class 1 if it containe a short, closed car.")

Class 2 (F-J): ∀(train) [num-cars(train)=2] ∨
∃(car) [contains(train,car)][shape(car)=jagged top]
("A train is in Class 2 if there are two cars in the train or if there is a car with a jagged top.")

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, then CONSEQUENCE is asserted. To illustrate the implicative statement, consider the assertion "vegetables are food" from the example in Sec. 4. It can be expressed in APC by the following statement that says "if an object is a

5. The two classes were labelled East-bound and West-bound, respectively, in the original formulation of the problem.

vegetable then it is also a food."

$$[\text{is-vegetable(object)}] \Rightarrow [\text{is-food(object)}]$$

An alternative way to express this idea in APC is

$$[\text{object-type(object)} = \text{vegetable}] \Rightarrow [\text{object-type(object)} = \text{food}]$$

which says "if the type category of an object is *vegetable* then 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.* The domain of the attribute "object-type" should have a tree structure in order to apply this rule. Further details on the APC language are given in [Michalski, 1983].

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 unique answers was seen in the experiment with human classification-building presented in Sec. 5. The choice of which classification is best can be based on some perceived set of goals [Medin et al., 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, 1980a]. Each elementary criterion measures a certain aspect of the generated descriptions, such as the fit between the classification and the objects, the simplicity of the descriptions (the reciprocal of the number of selectors), the discrimination index (the number of attributes that singly discriminate between all classes), and the dimensionality (the number of attributes that are necessary to classify the objects). The LEF consists of an ordered sequence of elementary criteria along with tolerances which control to what extent different solutions are considered equivalent. The user may request the use of any of the several built-in elementary evaluation functions in a user-defined lexicographical order for evaluating classifications. If more than one evaluation function is

requested, all classifications are evaluated according to the first function. Those that score best or within a user-defined tolerance threshold of the best are retained. They are then evaluated according to the next evaluation function, etc. until either a single classification remains, or the list of evaluation functions in the LEF is exhausted. In the later 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 combinations of hypothetical class descriptions.

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* in Sec. 4 is such a high-level classifying attribute. The RD and CA methods (described in the next section) both use background knowledge rules in the search for such attributes. The Goal Interaction Network (GIN) is traversed to find the interactions between the classification goal(s) and 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 a classification in which each class has a different value (or value range) for the classifying attribute.

As described in Sec. 4, the background knowledge rules can represent both general-purpose knowledge available to all problems, and domain-specific knowledge provided by the data analyst for a particular type of problem. 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

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

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

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

or

$$[\text{mother}(a,b)] \;\&\; ([\text{mother}(b,c)] \vee [\text{father}(b,c)]) \Longleftrightarrow [\text{grandmother}(a,c)]$$

Each kind is associated with a condition indicating the situations to which the rule is applicable.

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). To begin, let us briefly describe the program INDUCE/2 which solves concept acquisition tasks involving structured objects.

Given a set of objects described by a subset of APC and arranged into two or more classes, INDUCE/2 generates a description of each class (in the form of annotated predicate calculus expressions) that covers all the objects in the described class and no objects in any other class. This is accomplished in the following manner.

The objects are divided into two sets: set F1 contains objects belonging to the class being described, and set F0 contains objects belonging to any other class (counter-examples to set F1). One object at a time is selected from set F1 (the *seed* object or "focus of attention") and a *star* is built that *covers* the seed object *against* all objects in set F0. The star is the set of all alternative descriptions that are maximally general, describe the seed object and possibly other objects from F1, and describe no objects from F0 [Michalski, 1983].

To control combinatorial explosion, a *trimmed star* is produced rather than a complete star. The process for building a trimmed star is outlined here. A *partial star* is generated for each counter-example in set F0. A partial star is a set of maximally general alternative descriptions which describe the seed object but not a particular counter-example. The combinatorial explosion takes place as the partial stars are logically multiplied together to generate the star. In a trimmed star, a parameter MAXSTAR restricts the number of alternative descriptions that are retained following a logical multiplication of partial stars. When the number of alternative descriptions exceeds MAXSTAR, the MAXSTAR best descriptions are selected by applying the LEF criterion and the others are discarded. When the trimmed star is complete, the best description in the star (according to a LEF criterion) is

selected and becomes all or part of the solution. The objects covered by the resulting description are removed from set F1 and if objects remain in set F1 the induction process is repeated by selecting another seed object (from among those not yet covered) and building a star for it. When all objects in the set F1 have been covered, the solution is complete and is the disjunction of the descriptions which were the best produced in each star.

This algorithm for learning from examples can be adapted for solving classification construction problems. Given a set of unclassified objects, k seed objects are selected 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 when either 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).

The control layer that provides the representatives to the concept acquisition process must be able to pick good representatives or else the resulting classifications will be arbitrary without revealing underlying patterns in the data. One technique that has been used in a variety of clustering situations is to pick representatives at random in the first step. In the next steps *central*[6] representatives (those objects that best represent a typical object in their respective classes) are selected following each iteration that yields an improved classification (measured by the LEF). After an iteration which does not generate an improved classification, *extreme*[7] representatives (those objects that are most atypical) are selected [Michalski and Stepp, 1983b].

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

---

6. An event is *central* to a class of events if it has the minimum sum of syntactic distances to other events. Syntactic distance between two events is the sum of differences in values of descriptors, where individual differences are normalized to the range 0 to 1 by dividing by the maximum possible difference in descriptor value. In such a scheme, differences in nominal values are either 0 or 1; differences in linear (ordered) values are between 0 and 1, inclusive. Thus the syntactic distance between two events is a number between 0 and n, where n is the number of descriptors (i.e., the dimensionality of the event space).

are used:

(1) varying the number of classes, and

(2) composing the classes hierarchically.

Since the classification to be formed should be easy to understand by humans, we assume an upper limit on the number of classes that stem from any node of the classification hierarchy. This limit is assumed to be in the range of 4 to 7. Since this limit is small, it is computationally feasible to repeat the whole process for each number of classes. The solution that optimizes the score on the LEF (with appropriate adjustment for the effects 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 is implemented in the program CLUSTER/2 for the subset of annotated predicate calculus representations involving only attributes (zero-order functions). Besides its relative computational simplicity, this approach has other advantages stemming from descriptions (for both objects and classes) which are quantifier free. We note specifically 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 is 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 objects and uses 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.

---

7. An event is *extreme* to a class of events if it has the maximum sum of syntactic distances to other events; opposite of *central*.

A characteristic generalization provides 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 which 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 multi-place 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 to 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 is 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 of the use of a GIN and problem-specific background knowledge. Suppose that the knowledge base includes an inference rule which can identify trains carrying toxic chemicals. Suppose also that the general goal *survive* has a subordinate goal to monitor dangerous shipments. This additional background knowledge can be used to help build a classification.

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

[contains(train,car)][car-shape(car)=opentop][cargo-shape(car)=circle][items-carried(car)=1]
$\Longleftrightarrow$ [has_toxic_chemicals(train)]

In the above inference rule, equivalence is used to indicate that the negation of the condition part is sufficient to assert the negative of the consequence part. After the application of this rule, 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 GIN 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,* and *has_toxic_chemicals,* etc. This GIN is illustrated in Fig. 9. The GIN signals the relevancy of these descriptors, although in this simplified case none are given initially and the background knowledge contains only a rule to derive the has_toxic_chemicals predicate. The GIN also signals that the evaluation LEF should give preference to the inclusion of these descriptors in evaluating candidate classifications. Classifications which make use of the has_toxic_chemicals descriptor in formulating conceptual classes would score better than those that do not. The classification produced for this example is shown in Fig. 10. The LEF used to generate this classification differs from the criterion used for the results in Fig. 8. Instead of minimizing the number of attributes used, the first criterion applied is maximizing the relevancy of the descriptions. The derived attribute "has_toxic_chemicals(train)" gives a large relevancy score as suggested by information in the GIN.

6.5. Concept Formation by Finding Classifying Attributes: Method CA

In contrast to the method RD described in the preceding section, we will now describe a different method we call CA for *Classifying Attributes* method. The underlying goal of this approach is to find one or more *classifying* attributes whose observed value sets can be split into ranges that characterize

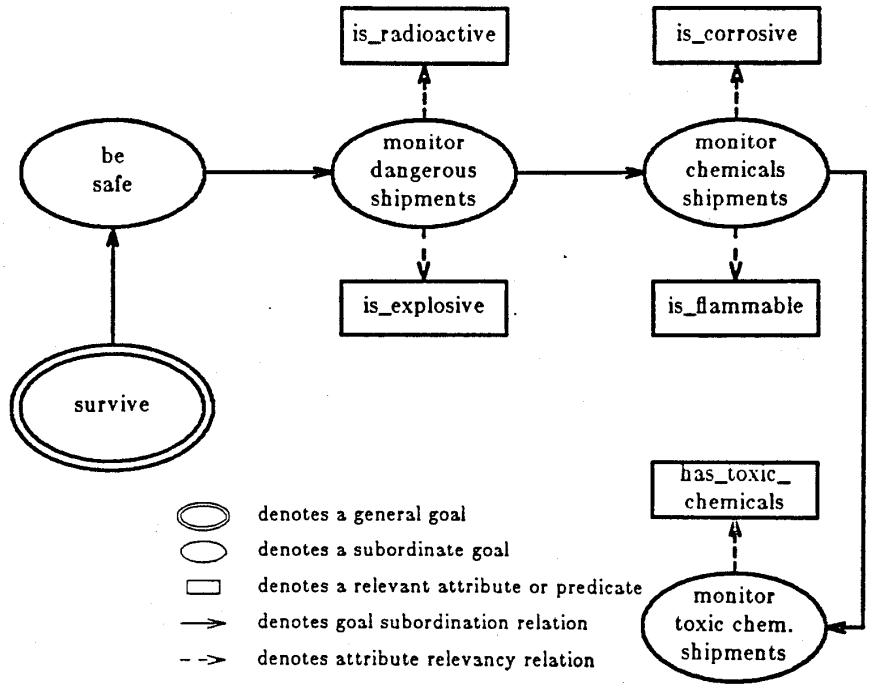

Figure 9. A hypothetical GIN for dangerous train shipments

each class. 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 during the course of constructing the classification. The classifying attributes sought are the ones that lead to classes of objects that are best according to the classification goal.

The importance of an descriptor can be determined either by consulting a GIN or by the fact that it implies many other descriptors. For example, if the goal of the classification is "finding food," the attribute "edibility" from Sec. 4 is the important classifying attribute. The second way of determining the importance 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 it implicatively leads to more secondary attributes than does the attribute

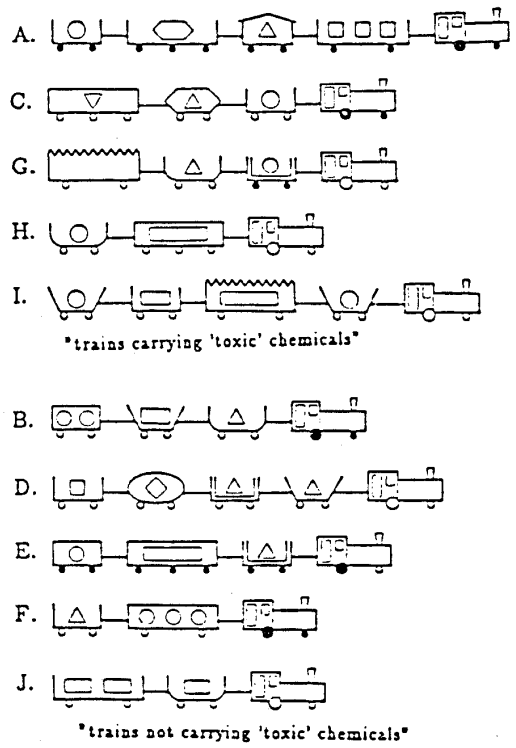

Figure 10. A classification produced using the 'toxic chemicals' inference rule.

"color" in a given attribute network (e.g., is-waterbird implies "can-swim," "has-webbed-feet," "eats-fish") [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 which can be inferred

are ordered by relevancy indicated by the GIN 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 be 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 described in Chapter 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

[cold-blooded(a1)][offspring birth(a1)=egg] ⇒ [animal-type(a1)=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.

7. Summary

This chapter has discussed approaches to building classifications of structured objects using goal-directed inferences from background knowledge. Two methods for performing this task were outlined. 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 for partitioning

the set of events in a way considered most appropriate according to a classification quality criterion (LEF).

The classifying attributes are either selected from the initially given ones or derived using background knowledge. The selection is aided by the use of a Goal Interaction Network which relates goals to subgoals and to relevant attributes which are to be preferred descriptors in the constructed conceptual categories that define object classes. The capability to incorporate domain-specific background knowledge in the form of inference rules and Goal Interaction Networks adds a new dimension to the tasks of concept formation and data analysis.

This work could be further extended through the investigation of alternative representations. These could include the used of additional logical operators such as implication, equivalence, and exception. The exception operator appears the most interesting because of its use by people. Exception clauses in logical rules can be used to provide *unless* conditions to handle the cases that are not frequent or ordinary. Censor-based learning (see Chapter 3) is based on the logical operation of exception.

A second extension of this work could be to develop a system capable of characterizing a collection of observations (facts, events) by building a semantic network in which nodes represent conceptual classes and links represent relations between them. The APC language used here has the same expressive power of semantic networks but it may be of interest to investigate whether the style of representation system has a significant impact on the development of the algorithms used to manipulate its content.

ACKNOWLEDGEMENTS

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 Insitute of Technology. Support for the University of Illinois Laboratory is provided in

part by grants by 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, 1973.

Burstein, M.H., "Concept Formation by Incremental Analogical Reasoning and Debugging," Chapter 13 in this book, 1985.

Carbonell, J.G., "Derivational Analogy in Problem Solving and Knowledge Acquisition," Proc. Intern. Machine Learning Workshop, Monticello, Illinois, June 22-24, 1983.

Dietterich, T.G. and Michalski, R.S., "A Comparative Review of Selected Methods for Learning from Examples," in *Machine Learning,* R.S. Michalski, J. Carbonell and T. Mitchell (Editors), Tioga Publishing Company, 1983.

DeJong, G., "Generalizations Based on Explanations," *Proceedings of the 7th IJCAI,* Vancouver, Canada, August 24-28, 1981.

DeJong, G., "An Approach to Learning from Observation," Chapter 19 in this book, 1985.

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

Langley, P., Zytkow, J., Simon, H.A., Bradshaw, G.L., "The Search For Regularity: Four Aspects of Scientific Discovery," Chapter 16 in this book, 1985.

Lingle, J.H., Altom, M.W., 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, J, Hortwick (eds.), Earlbaum, 1983.

Medin, D.L, "Structural Principles in Categorization," *Developments of Perception and Cognition,* 1982.

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

Mitchell, T.M. and Keller, R.M., "Goal Directed Learning," Proc. Intern. Machine Learning Workshop, Monticello, Illinois, June 22-24, 1983.

Michalski, R.S. and Larson, J.B., "Inductive Inference of VL Decision Rules," *Workshop in Pattern-Directed Inference Systems,* Hawaii, May 23-27, 1977 and published in SIGART Newsletter, ACM, No. 63, pp. 38-44, June 1977.

Michalski, R.S., "Pattern Recognition as Rule-Guided Inductive Inference," *IEEE Transactions on Pattern Analysis ad Machine Intelligence,* Vol. PAMI-2, NO. 4., pp. 349-361, July, 1980a.

Michalski, R.S., "Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts," *Policy Analysis ad Information Systems,* Vol. 4, No. 3, pp. 219-244, 1980b.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," in *Machine Learning,* R.S. Michalski, J. Carbonell and T. Mitchell (Editors), Tioga Publishing Company, 1983.

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

Michalski, R.S. and Stepp, R.E., "Learning From Observation: Conceptual Clustering," chapter in *Machine Learning,* R.S. Michalski, J. Carbonell and T. Mitchell (Editors), Tioga Publishing Company, 1983b.

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

Stepp, R.E., "CONJUNCTIVE CONCEPTUAL CLUSTERING: A Methodology and Experimentation," Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1984.

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

Winston, P.H., "Learning by Augmenting Rules and Accumulating Censors," Chapter 3 in this book, 1985.