

CONCEPTUAL CLUSTERING OF STRUCTURED
OBJECTS: A GOAL-ORIENTED APPROACH

by

R. Stepp
R. S. Michalski

,In AI Journal v,pp 103-110 1986.

Conceptual Clustering of Structured Objects: A Goal-Oriented Approach

Robert E. Stepp

*Department of Electrical and Computer Engineering, University of Illinois,
Urbana, IL 61801*

Ryszard S. Michalski*

*Artificial Intelligence Laboratory, Massachusetts Institute of Technology,
Cambridge, MA 02139*

ABSTRACT

Conceptual clustering is concerned with problems of grouping observed entities into conceptually simple classes. Earlier work on this subject assumed that the entities and classes are described in terms of a priori given multi-valued attributes. This research extends the previous work in three major ways:

- entities are characterized as compound objects requiring structural descriptions,
- relevant descriptive concepts (attributes and relations) are not necessarily given a priori but can be determined through reasoning about the goals of classification,
- inference rules are used to derive useful high-level descriptive concepts from the initially provided low-level concepts.

The created classes are described using Annotated Predicate Calculus (APC), which is a typed predicate calculus with additional operators. Relevant descriptive concepts appropriate for characterizing entities are determined by tracing links in a Goal Dependency Network (GDN) that represents relationships between goals, subgoals, and related attributes.

An experiment comparing results from the program CLUSTER/S that implements the classification generation process and results obtained from people indicates that the proposed method might offer a plausible cognition model of classification processes as well as an engineering solution to the problems of automatic classification generation.

1. Introduction

The process of forming meaningful classifications of observed entities is a difficult intellectual task, and usually precedes the development of a theory about the entities¹. According to the machine learning

*Author is on leave of absence from the University of Illinois at Urbana-Champaign.

1. By the term *entities* we mean any objects, phenomena, processes, etc. that are of interest and that can be described by statements in our representation language.

terminology proposed by Carbonell, Michalski, and Mitchell [4], this process is a form of *learning from observation* (learning without a teacher). Until recently, problems of developing computational methods and a theory underlying classification construction procedures have not received much attention from the AI community. Classification was studied primarily under the headings of numerical taxonomy and cluster analysis [1]. Those methods utilize a mathematical measure of similarity between entities, defined over a finite, a priori given set of multi-valued or continuously-valued attributes. Generated classes are collections of objects with high intraclass and low interclass similarity. The methods assume that all relevant attributes for characterizing entities are provided initially and that they are 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. Likewise, they do not take into consideration possible goals of classification that seem to underly human classification processes.

As a result, classifications obtained by traditional methods are often difficult to interpret conceptually and may not reflect the goals a data analyst might have in mind. Moreover, the classes generated by a given method require interpretation since no conceptual description is provided..

Another aspect of traditional classification-building methods is that they describe objects by attribute value sequences and therefore are inadequate for creating classifications of complicated objects composed of various interrelated component parts. A description of such entities must involve not only attributes of whole entities but also attributes of their parts and relationships among them.

This research is concerned with the problem of automating the process of generating classifications of structured entities through *conceptual clustering*. In this approach, classes (clusters of entities) are generated by first formulating conceptual descriptions of these classes and then classifying the entities according to the descriptions.

2. The Goal of This Research

The idea of *conceptual clustering* leads to an entirely new approach to the problem of creating classifications [12, 17, 18, 21]. It is based on the assumption that entities 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, events (entities) were described by attribute value sequences. The method arranged the events into a hierarchy of classes described by *conjunctive concepts*. These concepts were described by logical products of relations on selected attributes. The generated sibling classes of any node in the hierarchy represented the most preferred classification from this node according to a given *preference criterion*. The background knowledge consisted of the definitions of the attributes used in event 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 that 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 put forth by Michalski [15]. Important work involving background knowledge has been done by Winston [23], who describes an incremental learning process in which the background knowledge contains relevant precedents, exercises, and *unless conditions*. DeJong [5]

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

presents a method of using background knowledge to acquire explanatory schemata that describe sequences of events presented as stories. Background knowledge is also used by Mitchell and Keller [19] to guide an inductive learning program for acquiring problem-solving heuristics in integral calculus. In learning by analogy described by Burstein [2], a large body of causal knowledge is used to "fill out" incomplete descriptions and guide analogical inference. Carbonell [3] presents a method for acquiring problem-solving strategies by analogy to solutions of similar problems. Rendell's Probabilistic Learning System [20] 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. Various aspects of the problem of learning structural descriptions from examples are discussed in [22] and [7].

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

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

This section briefly describes the authors' 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 events forms a class only if it can be described by a conjunctive concept involving relations on their attributes. AC^3 is a special case of general *conceptual clustering* that generates a network of concepts to characterize a collection of entities. The problem posed in the framework of AC^3 is defined as follows:

Given: A set of events (physical objects or abstract entities),

A set of attributes to be used to characterize the events, 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 event 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 a *clustering quality criterion*.

As mentioned before, in conventional data analysis classes of events are formulated solely on the basis of a measure of event similarity. The similarity between any two events is characterized by a single number: the value of a similarity function applied to symbolic descriptions of the events. These symbolic descriptions are vectors, whose components are scores on selected attributes in event descriptions. Such measures of similarity are *context free*; that is, the similarity between any two events (objects) A and B depends solely on the properties of the objects and is not influenced by any context (the *environment* surrounding the events). 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 but 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 events representing certain concepts.

This idea is the basis of conceptual clustering. Rather than group entities together on the basis of the similarity between two entities such as A and B , the method groups entities together on the basis of the *conceptual cohesiveness* between A and B . The conceptual cohesiveness between two events depends not only on those events and surrounding events 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 events 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 algorithm for conjunctive conceptual clustering can be described as consisting of two parts: a *clustering part* and a *hierarchy-building part*. The clustering part of the algorithm arranges objects into classes using conceptual cohesiveness, so that the obtained clustering optimizes the given context-based clustering quality criterion. The hierarchy-building part of the algorithm starts by building first-level conceptual classifications of all objects (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 (for example, until the height of the hierarchy reaches a certain level or the sizes of the clusters fall below a certain size).

The clustering 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 quality criterion is monitored. The algorithm halts when the clustering quality criterion does not improve for a dynamically determined number of iterations. The algorithm is described in detail by Michalski and Stepp [18].

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. 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:

- 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* 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* objects versus *inedible* objects.

When the background knowledge contains many such rules of inference, a large number of different but equally meaningful candidate classifications can be created. The problem is how to select the best or most appropriate classifications from among the candidate classifications. 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*.

One way to resolve the classification selection problem is by assuming a general goal of the classification. For example, assume that the general goal is to *survive*. Among other things, this goal 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, by consuming food and by drinking liquids. In the context of the subordinate goals reached by links from the most general goal node, the relevant attributes might be, for example, *is_edible*, *is_potable*, and *tastes_good*. The attribute *tastes_good* can be linked by an implication relation to the attributes *is_edible* or *is_potable* (if something tastes good then it is either edible or potable).

Thus, the classification building process can be guided by a general goal that leads 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 together goals, subgoals, and relevant attributes. Part of a hypothetical GDN headed by the *survive* goal is shown in figure 1. In the illustration, main goals are denoted by double 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 for a classification include not only *survive*, but also *be healthy and beautiful*. When both goals are involved, a GDN such as the one in figure 2 is used. Here, the links from the two top-level goals converge at the *consume dietary food* subgoal which links to the subordinate goals

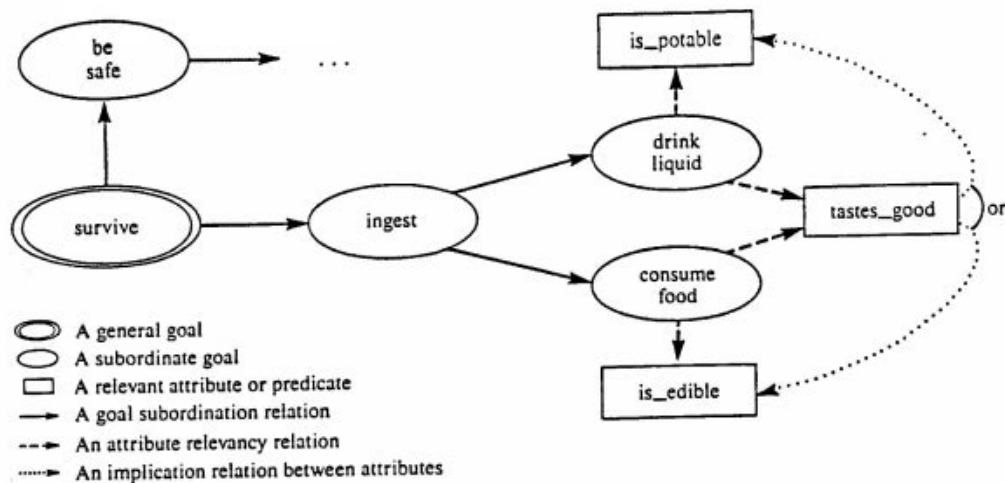


FIG. 1. A Goal Dependency Network headed by the goal to survive.

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 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 3.

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 the classification, and we want it to create the classification into *edible* versus *inedible* objects.

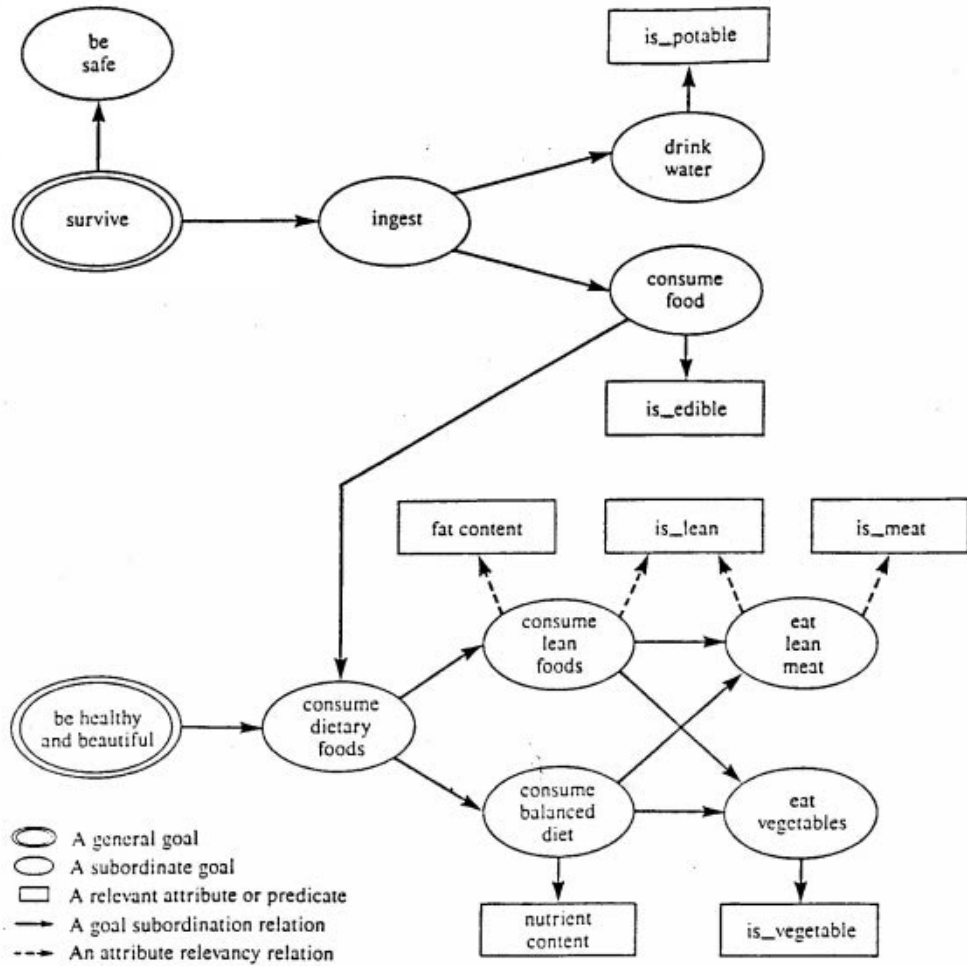


FIG. 2. A GDN for the goals *survive* and *be healthy and beautiful*.

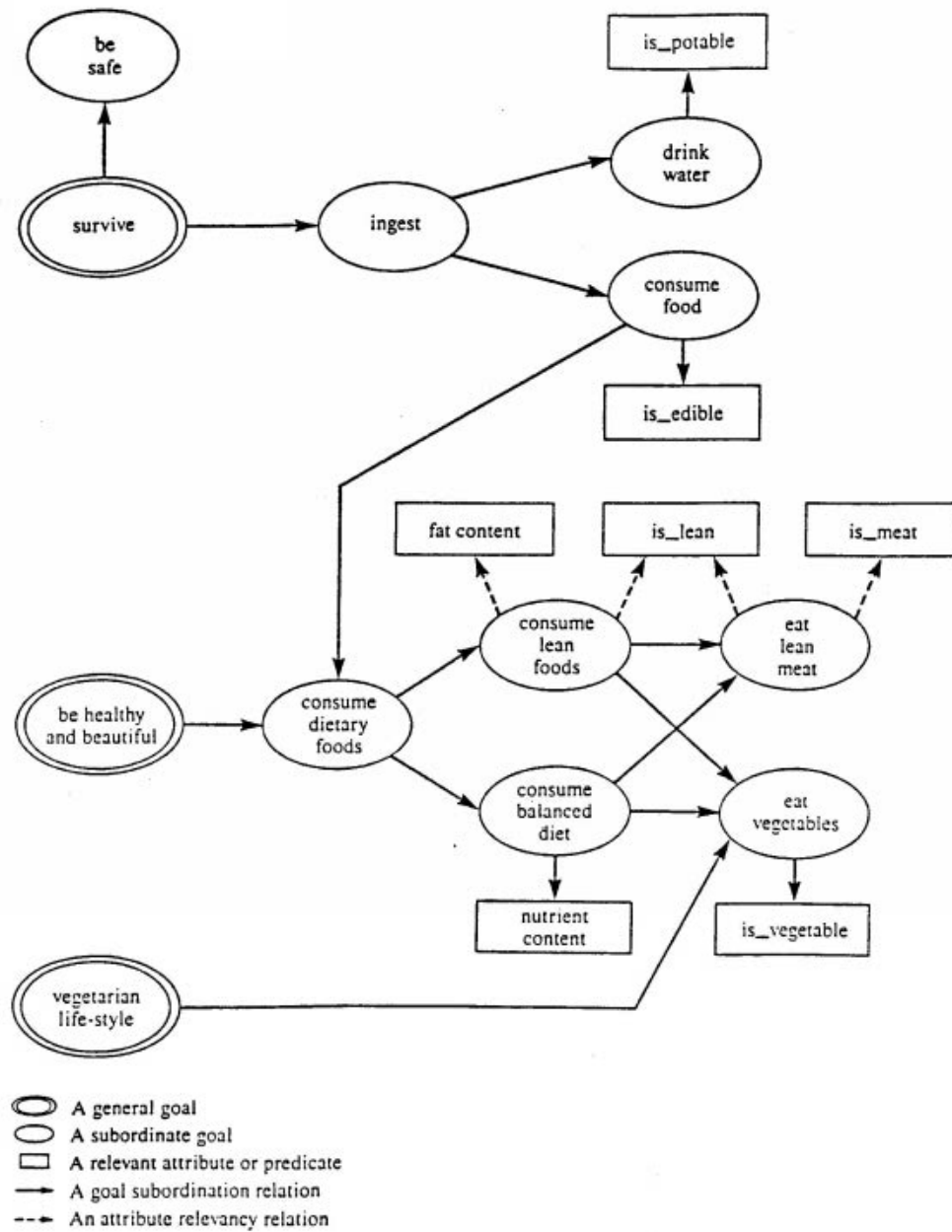


FIG. 3. A GDN for the goals *survive*, *be healthy and beautiful*, and *vegetarian life-style*.

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 [22] 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 classified as *general purpose* or *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 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* (LEF) (see section 6.2) is used to help select from among candidate classification schemes one classification that is appropriate for the problem at hand and that directs the algorithm towards solutions that meet a given goal.

Domain-specific background knowledge consists of inference rules for deriving values for new descriptors and a GDN to infer which descriptors (attributes, functions, or predicates) are likely to be 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 towards descriptors that are likely to be relevant and thus worth the computational cost of their

derivation.

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 [8] 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(p3)` when given `ontop(p1,p2)` and `ontop(p2,p3)`.

5. Building Classifications of Structured Objects

Let us turn now to the problem of classifying structured objects. Consider for example the problem of finding a classification of some trains,³ shown in figure 4. 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.

Human classifications of the trains shown in figure 4 have been investigated by Medin, Wattenmaker, and Michalski, [12]. 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:

3. This example is a reformulation of the problem known as "east- and west-bound trains" [16]. In the original formulation, two collections of trains were given— those that were east-bound (A to E) and those that were west-bound (F to J). The problem was to learn a simple rule to distinguish between the east-bound and west-bound trains. Thus the original problem was that of *learning from examples, or concept acquisition*.

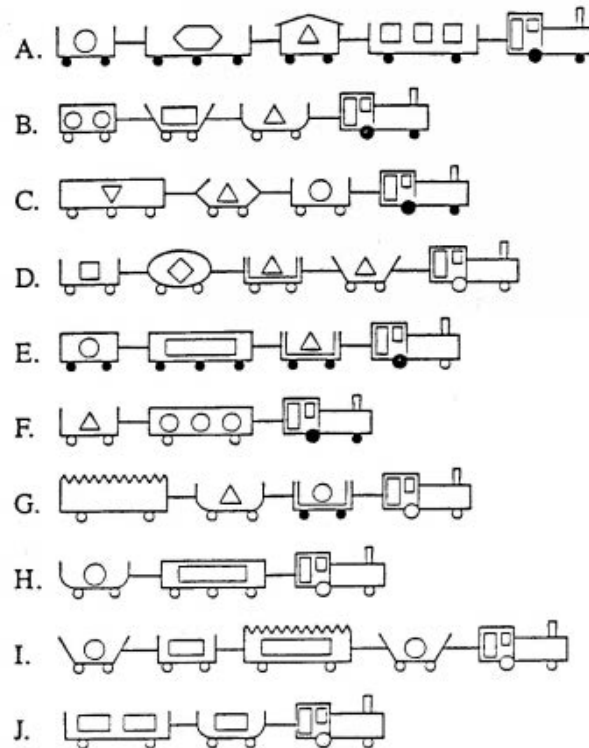


FIG. 4. How would you classify these trains?

- 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 formed a total of ninety three classifications of the trains. The most popular concepts used to form a classification (seventeen repetitions) involved the number of cars in the trains. The three classes formed were described by the following concepts: "trains containing 2 cars," "trains containing 3 cars," and "trains containing 4 cars." The second most

frequent classification (seven repetitions) was based on the engine wheel color. These two classifications are shown in figure 5. Of the ninety three classifications produced, forty of them were unique. Thus, although there was no explicit goal given for this classification, there was some pattern among the subjects. In this case, the pattern was not a very strong one, as indicated by the 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

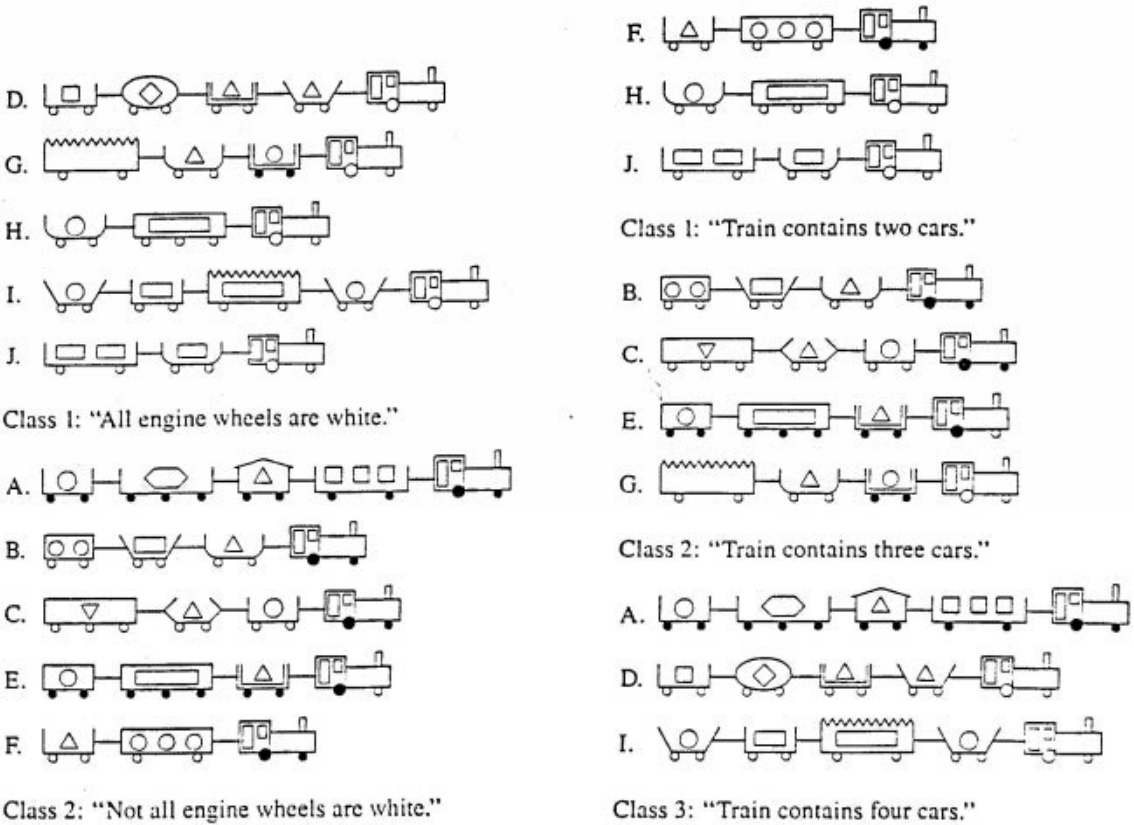


FIG. 5. The two most popular classifications produced by people.

by Medin. People typically do not suggest intersecting classifications, and it is for this reason that we focus on methods that produce disjoint descriptions.

The problem of classifying toy trains represents a general category of classification problems in which one wants to organize and classify observations that require structural descriptions, for example, classifying physical or chemical structures, analyzing genetic sequences, building taxonomies of plants or animals, characterizing visual scenes, or splitting 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 [5].

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 or 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 the observed events in the class so as to insure a good "fit" between each class description and the events.

Figure 6 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 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* and *same color of wheels* attributes are predicates of two or more variables that denote the

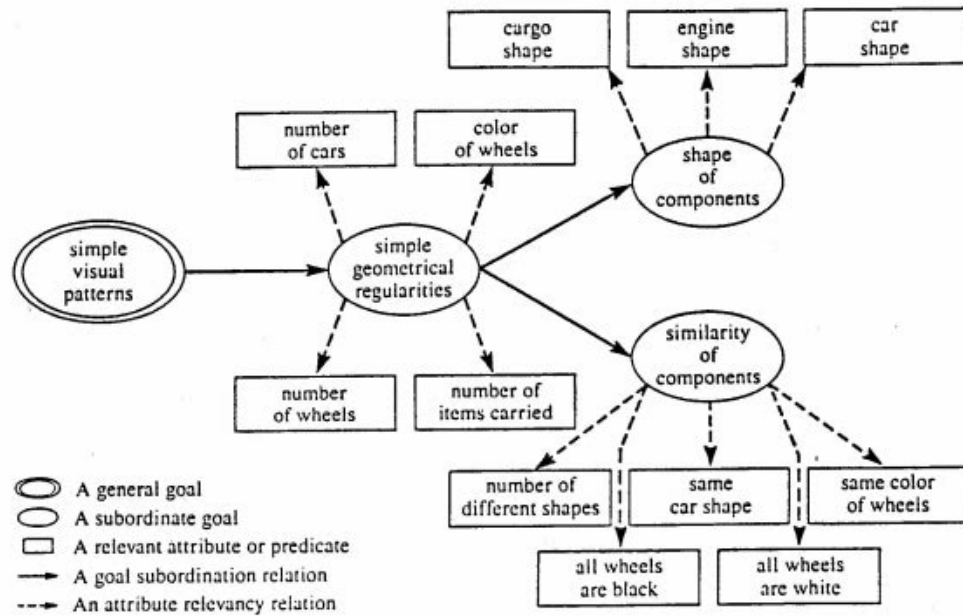


FIG. 6. A GDN for the goal of finding simple visual patterns.

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.

As examples of solutions obtained by the method, figure 7 shows two classifications created for the trains problem. Given structured descriptions of each train involving the descriptors *contains*, *in front*, *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.

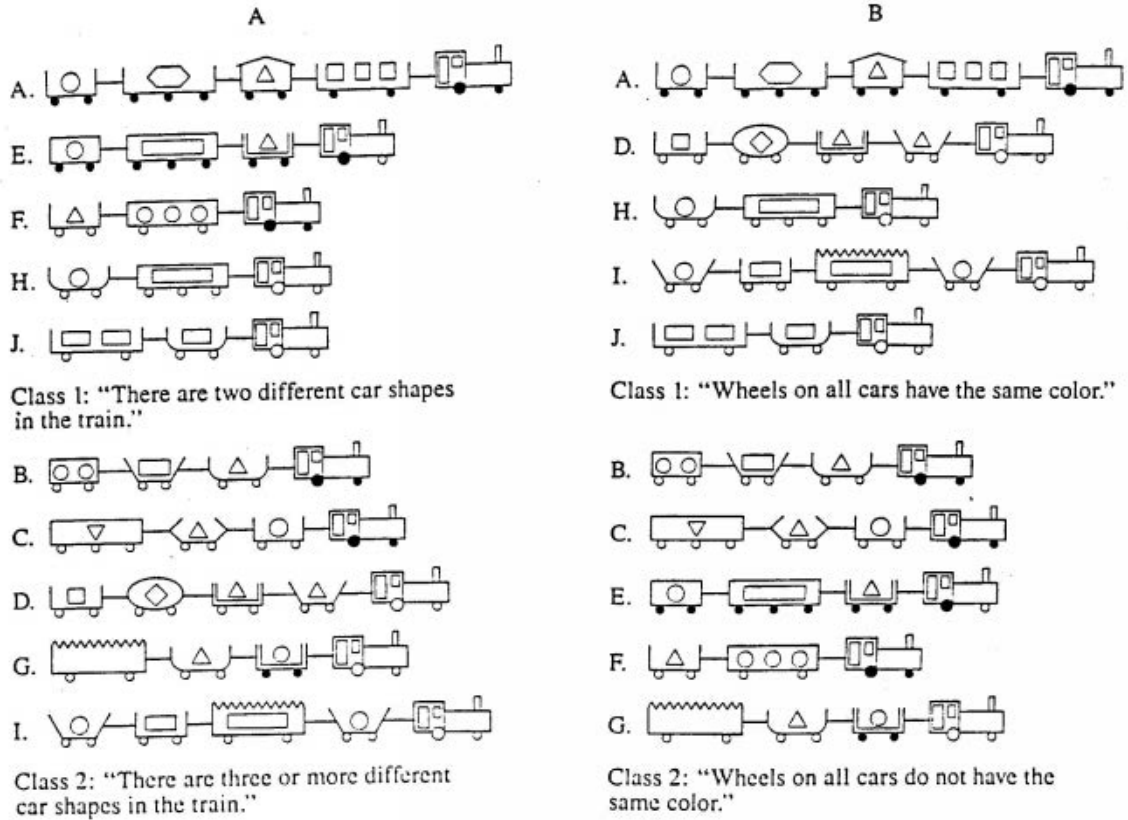


FIG. 7. Two sample classifications found by one of the two presented methods.

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 (ninety percent) for the first elementary criterion and zero tolerances for the second and third elementary criteria in the LEF evaluation criterion described in section 6.2.

The event partitioning into the two clusters shown in classification A of figure 7 was generated by the program for two different conceptual clusterings. The top class ("There are two different car shapes in

the train") was also described in a different clustering 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 7 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 7 are appealing because the difference between classes is striking yet not obvious from casual inspection.

6. Two Methods for Building Classifications

Conceptual clustering classification building problems such as those of the previous section can demand great computational resources. In this section two resource limited methods for building a classification of a collection of structured objects are outlined. One method is called RD for *repeated discrimination*. The other method is called CA 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 [18]. Method RD has been implemented in the program CLUSTER/S [21].

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 a Goal Dependency Network. An implementation of method CA is in progress.

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 also use the LEF as the general-purpose criterion for measuring the *quality* of generated candidate solutions. Of the two, method RD is more data-driven while method CA is more model-driven. Our discussion of the two methods begins with an explanation of the Annotated Predicate Calculus (APC) and the Lexicographical Evaluation Functional (LEF) in the next two sections,

respectively.

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 [15]. 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 a descriptor hierarchy, and other information. For example, the function *shape* can be annotated with its domain type (DOMAIN TYPE=UNORDERED) and the structure of values in the domain (SQUARE \vee TRIANGLE \vee DIAMOND \Rightarrow POLYGON; CIRCLE \vee ELLIPSE \Rightarrow CURVED). The form of annotations adapts to encode any auxiliary information that aids correct interpretation of the predicate, variable, or function.

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

$$[\text{atomic-function REL value-of-atomic-function}]$$

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

$$[\text{weight}(\text{box}) > 2\text{kg}]$$

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:

$$[\text{color}(\text{box}) = \text{red} \vee \text{purple}] \quad \text{"the color of the box is either red or purple."}$$

$$[\text{color}(\text{box1} \ \& \ \text{box2}) = \text{red}] \quad \text{"the color of box1 and box 2 is red."}$$

The meaning of the internal disjunction operator is defined by

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

and the meaning of the internal conjunction operator is defined by

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

Selectors can be combined by standard operators to form more complex expressions.

Background knowledge is expressed as a set of APC implicative rules:

$$\text{CONDITION} \Rightarrow \text{CONSEQUENCE}$$

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 understand the implicative statement, consider the assertion "vegetables are food" from the example in section 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":

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

An alternative way to express this idea in APC is

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

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*. Further details on the APC language are given by Michalski [15].

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 5. The decision about which classification to choose can be based on some perceived set of goals [12], a goal-oriented, statistic-based utility function [20], 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) [14]. 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 [18].

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 a 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.

The LEF can be used to select the best classifications from an exhaustively generated set of candidates, but that kind of exhaustive generate and test is not practical for most problems. Besides its use for final selection of classifications, the LEF is also used during the assembly of candidate classification descriptions as a heuristic for a variation of *beam search*. Classification descriptions are assembled by adding a few selected selector conjuncts at a time. The candidate classification descriptions are evaluated by the LEF and all but a user specified number of best descriptions are discarded prior to extending the descriptions with additional selectors. In this way the LEF provides a powerful heuristic for searching the huge space of hypothetical classifications using only a small fraction of the effort required for exhaustive generate and test. This latter application of the LEF is used to build *bounded stars*, described later in section 6.4.

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 4 is such a high-level classifying attribute. The repeated discrimination and classifying attributes methods described in section 6.4 and 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 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 4, the background knowledge rules can represent both 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

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

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

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

or

$$\forall p1, p2, p3, [\text{mother}(p1, p2)] \& \{[\text{mother}(p2, p3)] \vee [\text{father}(p2, p3)]\} \iff [\text{grandmother}(p1, p3)].$$

Each rule is associated with a condition defining the situations to which it 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). We start with a brief description of the program INDUCE/2, which solves concept acquisition tasks involving structured objects.

Given a set of events (symbolic descriptions of objects or situations) arranged into two or more classes, INDUCE/2 used inductive inference techniques to build a general description of each class in the form of an annotated predicate calculus expression. The generated class descriptions are consistent with the training data (the description for class *i* is satisfied by each training event in class *i* and no others) and optimizes a given evaluation criterion.

The descriptions are generated in the following manner. First, all events are divided into two sets: set *F1* contains events belonging to the class currently being considered and set *F0* contains 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 $F0$. (*Star generation* is described in [15, 18].)

To control combinatorial explosion, INDUCE/2 determines *bounded* stars rather than complete stars. A bounded star contains only a fixed number of most promising descriptions selected according to the LEF. The highest-rank description in the bounded star is chosen as a part of the solution and the events covered by that 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 description of the class 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 single class of events, k seed events 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 event in each newly formed class (defined as the set of events satisfying the class description). The representative events 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 events (corresponding to the number of classes). Since the best number of classes to form is usually unknown, two techniques are used: varying the number of classes and 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 is 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 events and classes) that are quantifier free. It should be noted that classifications normally have the property that they can unambiguously classify any event 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, 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 binding 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. The attributes are the original functions and predicates stripped of their arguments with the same assigned values as before. The descriptions of object cores thus need neither quantified variables nor multiplace predicates in their descriptions. At this point, the derived descriptions of object cores can be handled by the CLUSTER/2 program.

It is recognized that some structural differences between objects could be lost by the above approach because it focuses on the *common* substructure found in all given objects. To retain some unique structural features of individual objects that might otherwise be lost, an inspection is made of connecting relations between object subparts within the core and object subparts outside the core. New predicates are generated and added to object descriptions to denote the attachment to the core substructure of different kinds of object substructures that lie outside the core.

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 (for example, 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 of 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 *monitor dangerous shipments*. This 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

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

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.

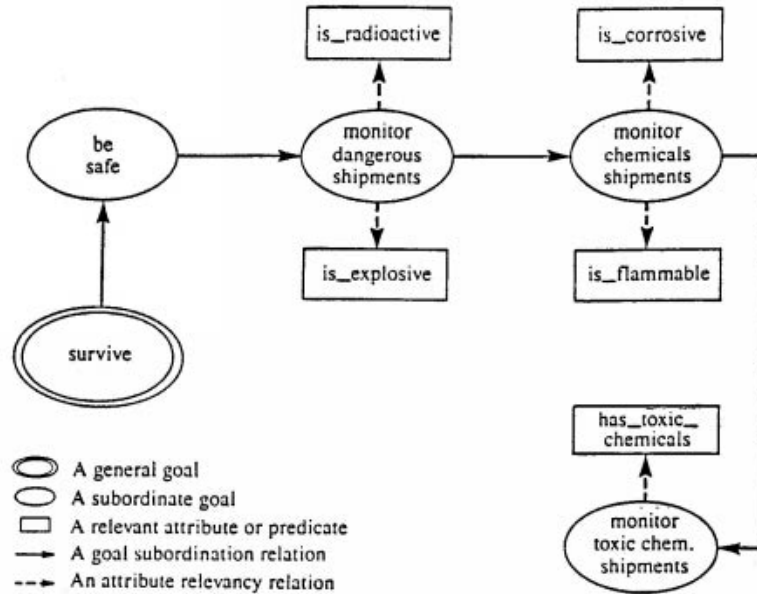


FIG. 8. A hypothetical GDN for dangerous train shipments

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. This GDN is illustrated in figure 8. 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 related to visual patterns. The classification produced in this case is shown in figure 9.

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 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 is *finding food*, the attribute *edibility* from section 4 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

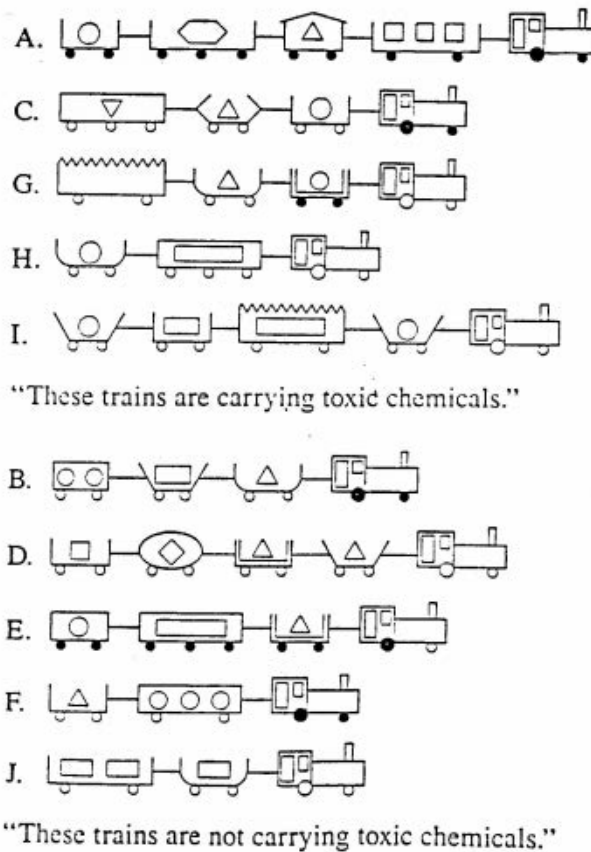


FIG. 9. A classification produced using the 'toxic chemicals' inference rule.

of *is-waterbird* because the latter implicatively leads to more implied attributes than does the attribute *color* in a given GDN (for example, *is-waterbird* implies *can-swim*, *has-webbed-feet*, *eats-fish*, and so on) [11].

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 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 [9]. 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.

7. Summary

This paper has discussed approaches to building classifications of structured entities 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 Dependency Network, that 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.

An extension of this work could be the development of a system capable of characterizing a collection of observations (facts or events) 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 disjoint from*. 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 number NSF DCR 84-06801 and the Office of Naval Research under grant number 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.

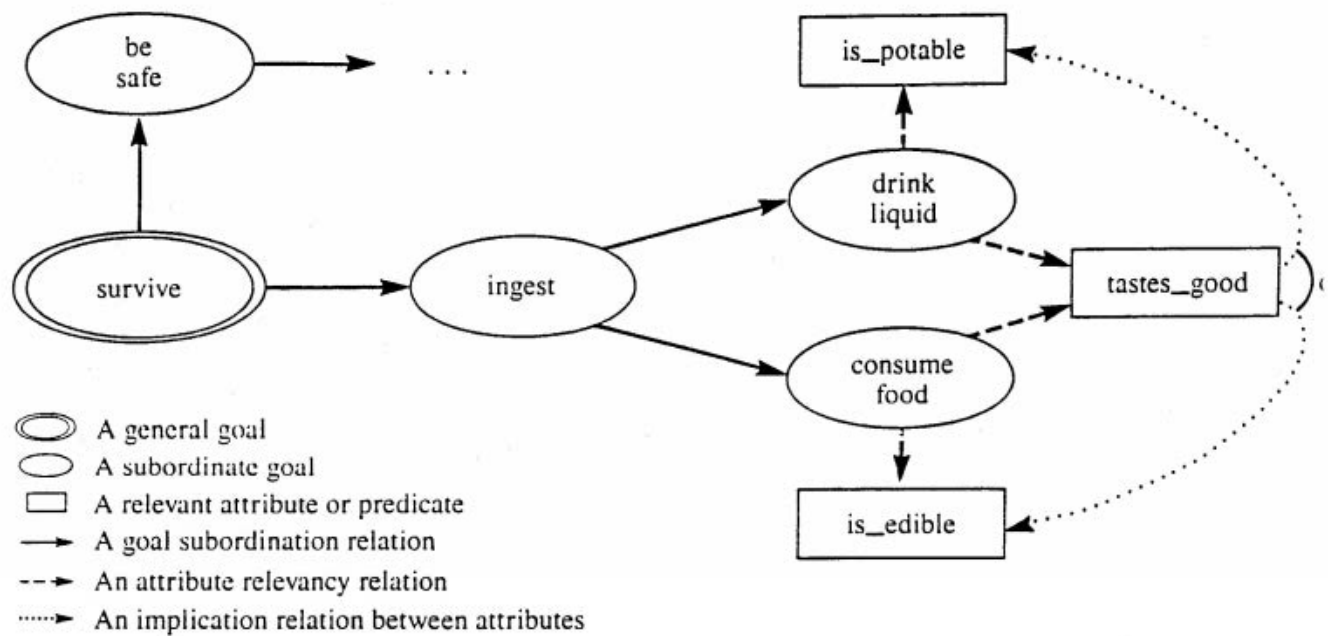
The authors wish to thank Tom Mitchell at Rutgers University and Larry Rendell at the University of Illinois for remarks and criticisms on earlier versions of the manuscript. They also thank Peter Andreae at the Massachusetts Institute of Technology Artificial Intelligence Laboratory and Doug Medin at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

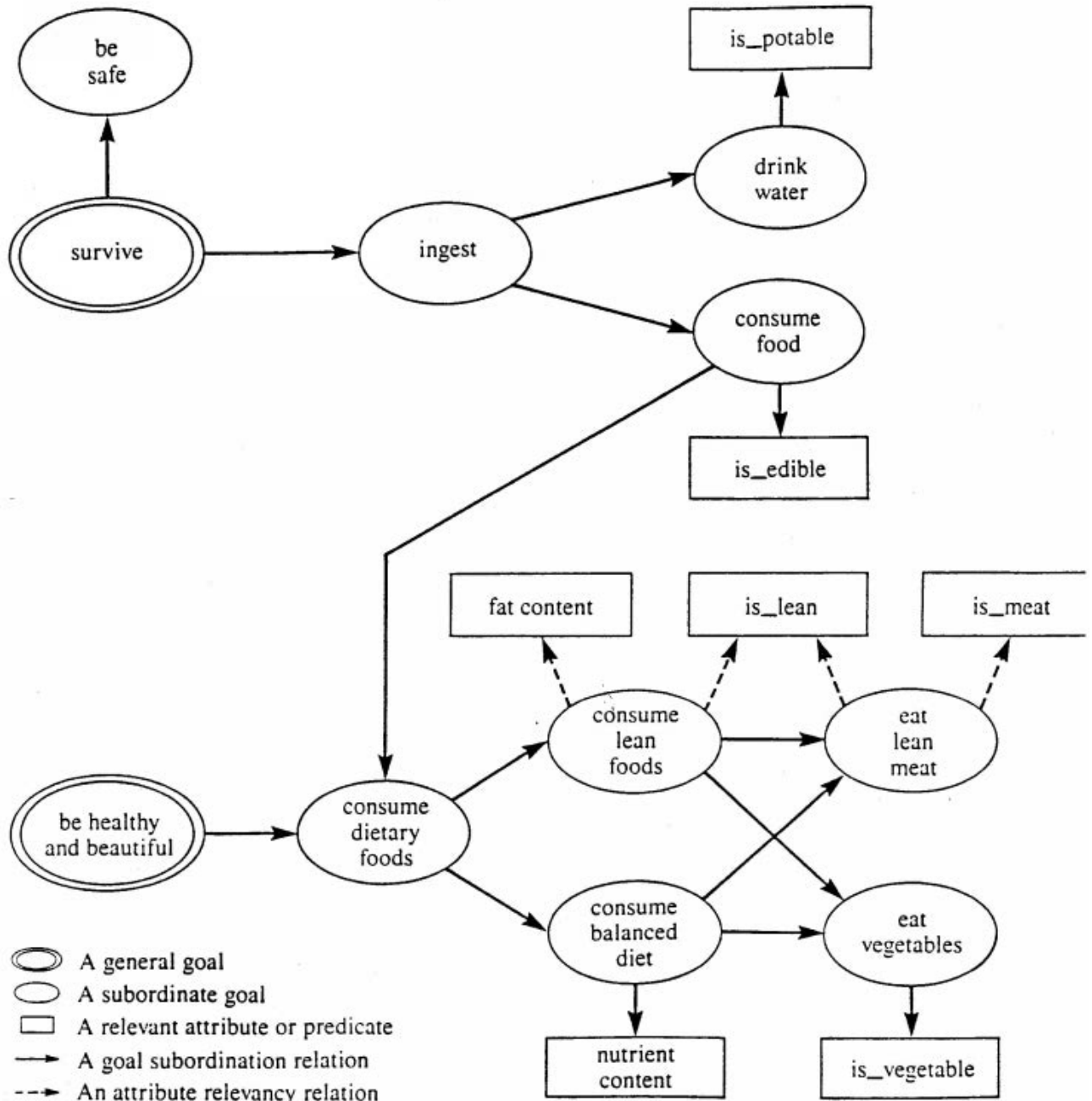
REFERENCES

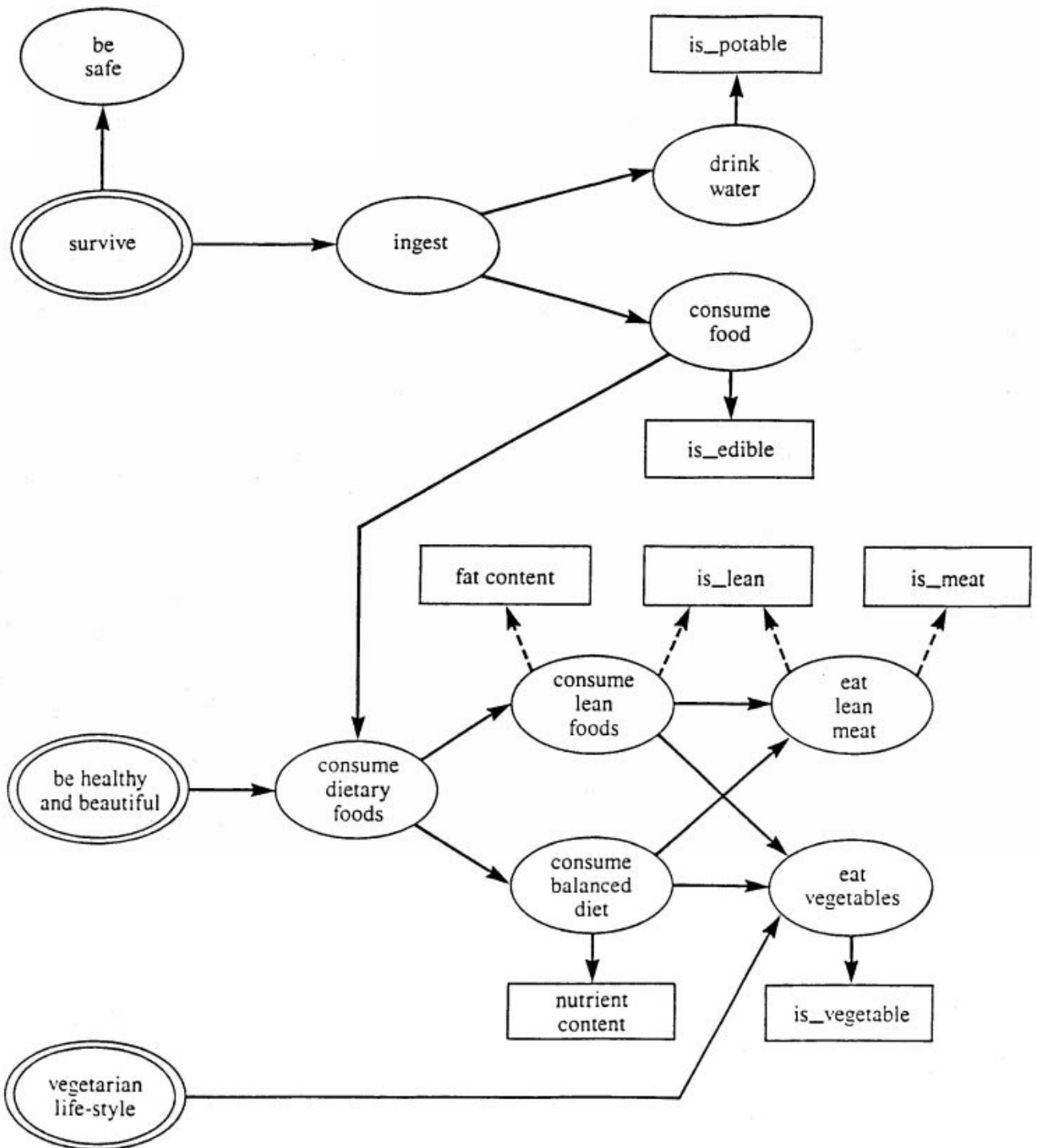
1. Anderberg, M. R., *Cluster Analysis for Applications*, Academic Press, New York, 1973.
2. Burstein, M. H., "Concept Formation by Incremental Analogical Reasoning and Debugging," to appear in *Machine Learning: An Artificial Intelligence Approach II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann, Los Altos, Calif., 1985.
3. 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.
4. Carbonell, J. G., Michalski, R. S., and Mitchell, T. M., "An Overview of Machine Learning," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Tioga, Palo Alto, Calif., 1983.
5. DeJong, G., "Generalizations Based on Explanations," *Proceedings of the Seventh IJCAI*, Vancouver, B. C., pp. 67-69, 1981.
6. -----, "An Approach to Learning from Observation," to appear in *Machine Learning: An Artificial Intelligence Approach II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann, Los Altos, Calif., 1985.

7. 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.
8. Hoff, W., Michalski, R. S., and Stepp, R. E., "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, January, 1983.
9. Langley, P., Zytkow, J., Simon, H. A., and Bradshaw, G. L., "The Search For Regularity: Four Aspects of Scientific Discovery," to appear in *Machine Learning: An Artificial Intelligence Approach II*, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), Morgan Kaufmann, Los Altos, Calif., 1985.
10. 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, J. Hortwick (Eds.), Earlbaum, Hillsdale, N.J., 1983.
11. Medin, D. L., "Structural Principles in Categorization," in *The Development of Perception and Cognition*, T. Tighe, B. Shepp, and H. Pick (Eds.), Earlbaum, Hillsdale, N.J., 1982.
12. 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.
13. 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* 4, no. 3, pp. 219-244, 1980a.
14. ———, "Pattern Recognition as Rule-Guided Inductive Inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2, no. 4., pp. 349-361, July, 1980.
15. ———, "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.
16. 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 *ACM SIGART Newsletter*, no. 63, pp. 38-44, June, 1977.
17. Michalski, R. S. and Stepp, R. E., "Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy," *IEEE Trans. on Pattern Analysis and Machine Intelligence* 5, no. 4, pp. 396-410, July, 1983.
18. ———, "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., 1983.
19. Mitchell, T. M. and Keller, R. M., "Goal Directed Learning," *Proceedings of the International Machine Learning Workshop*, R. S. Michalski (Ed.), Allerton House, University of Illinois at Urbana-Champaign, pp. 117-118, June 22-24, 1983.


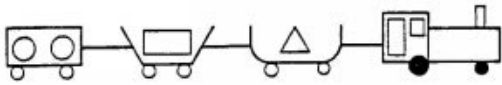
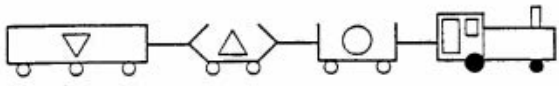

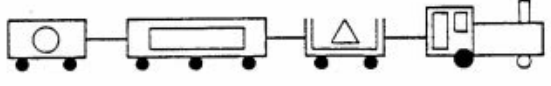
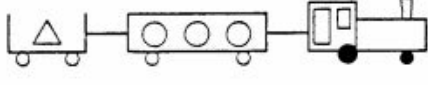
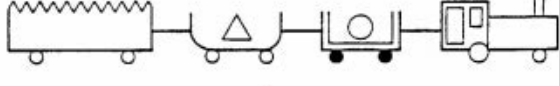
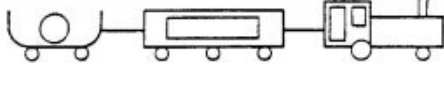
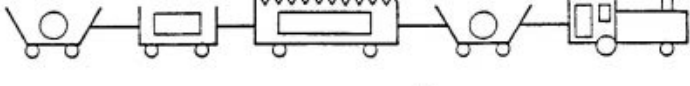
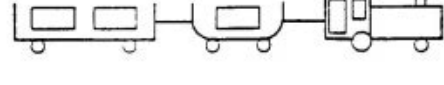
20. Rendell, L. A., "Toward a Unified Approach for Conceptual Knowledge Acquisition," *The AI Magazine*, Winter, 1983.
21. Stepp, R. E., "Conjunctive Conceptual Clustering: A Methodology and Experimentation," Ph.D. diss., Department of Computer Science, University of Illinois at Urbana-Champaign, 1984.
22. Winston, P. H., *Artificial Intelligence*, Addison-Wesley, Reading, Mass., 1984.
23. ———, "Learning by Augmenting Rules and Accumulating Censors," to appear in *Machine Learning: An Artificial Intelligence Approach II*, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), Morgan Kaufmann, Los Altos, Calif., 1985.

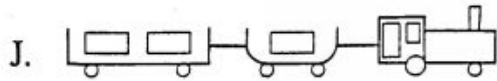
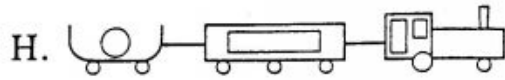
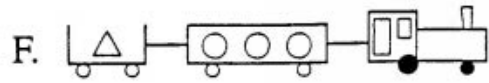




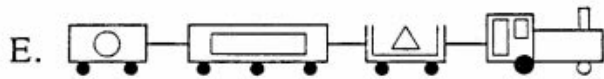


- A general goal
- A subordinate goal
- A relevant attribute or predicate
- A goal subordination relation
- - - An attribute relevancy relation

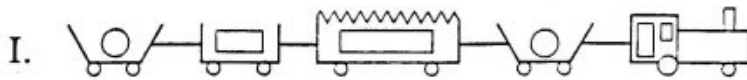
- A. 
- B. 
- C. 
- D. 
- E. 
- F. 
- G. 
- H. 
- I. 
- J. 



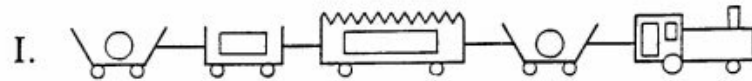
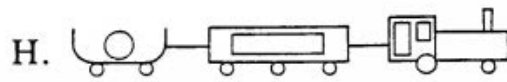
Class 1: "Train contains two cars."



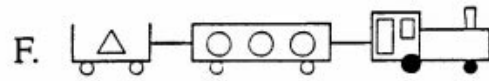
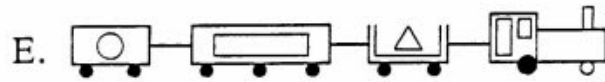
Class 2: "Train contains three cars."



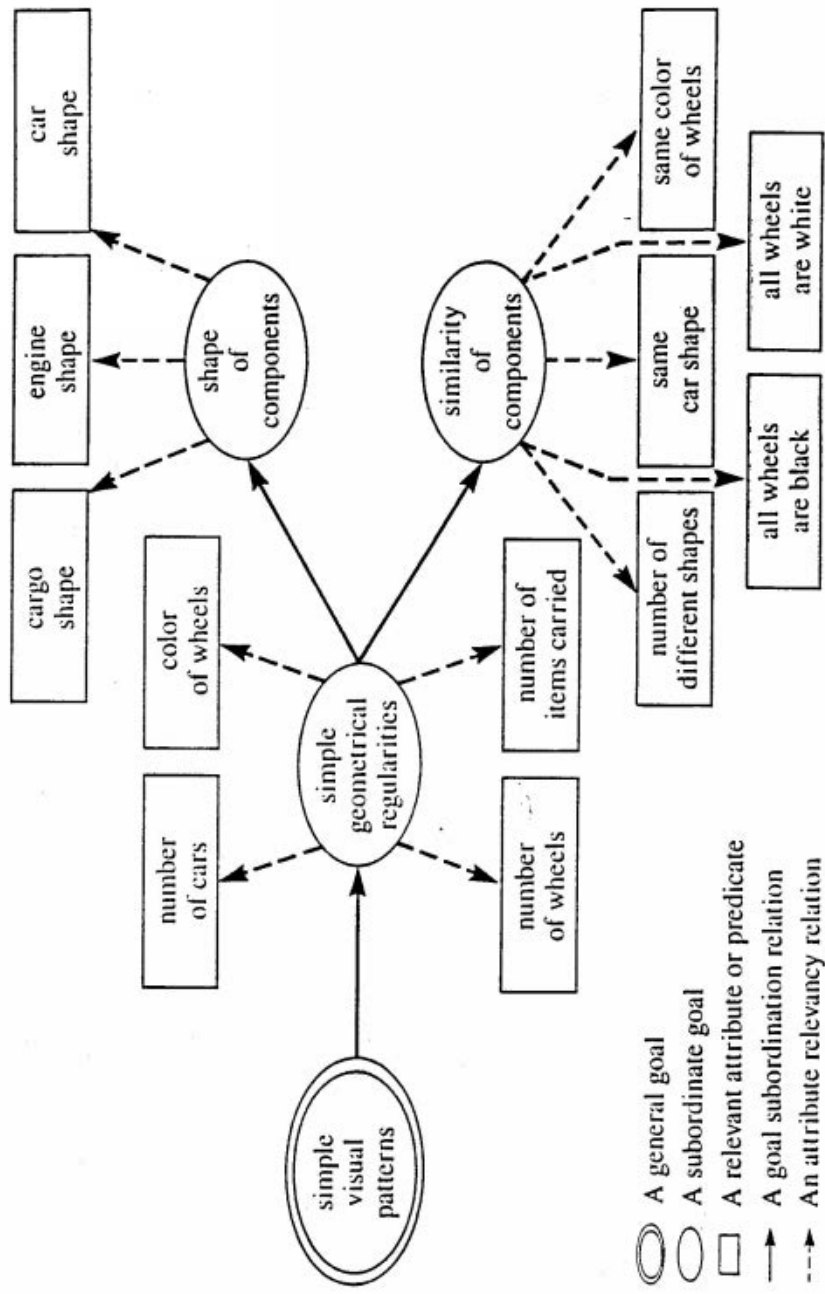
Class 3: "Train contains four cars."



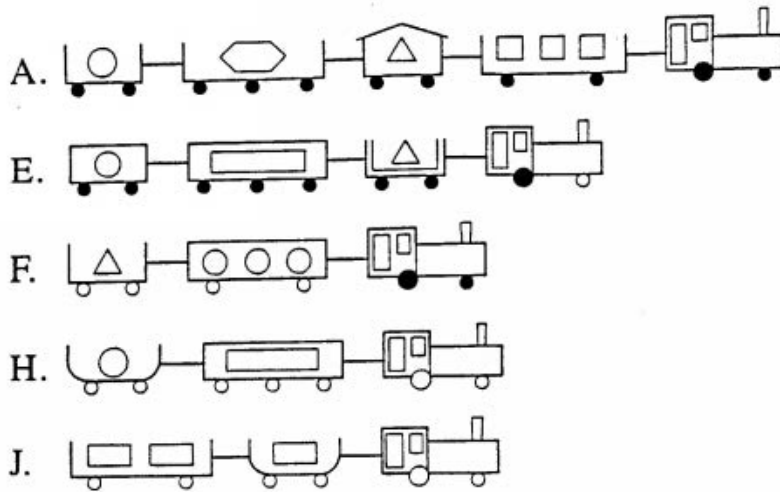
Class 1: "All engine wheels are white."



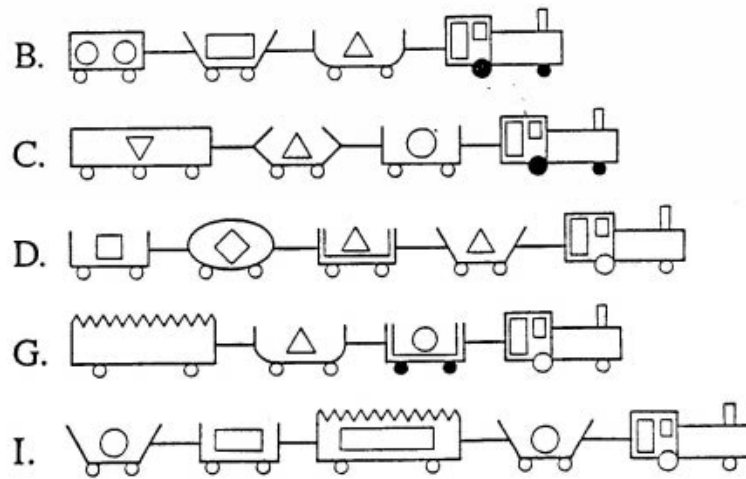
Class 2: "Not all engine wheels are white."



A

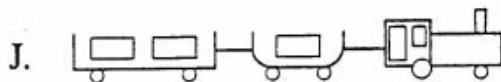
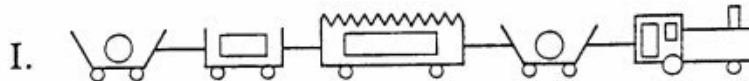
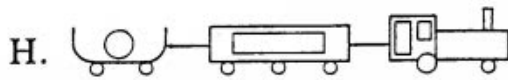


Class 1: "There are two different car shapes in the train."

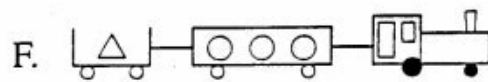


Class 2: "There are three or more different car shapes in the train."

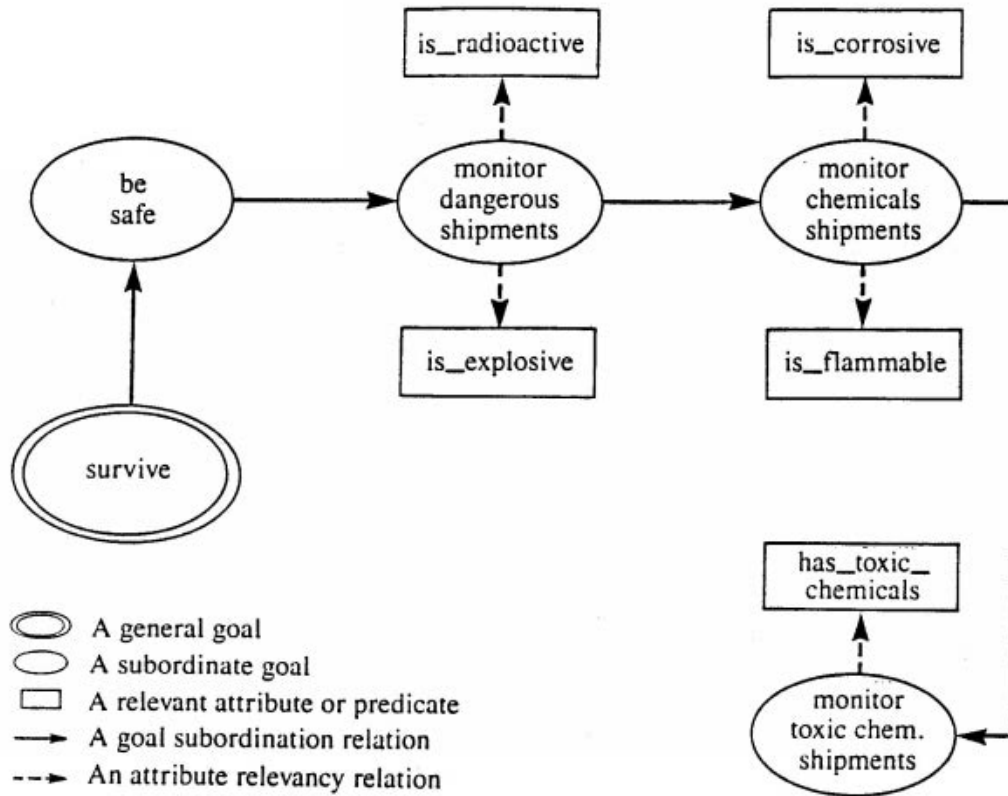
B

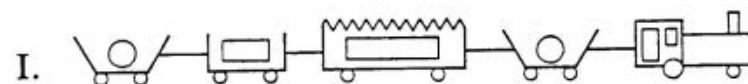
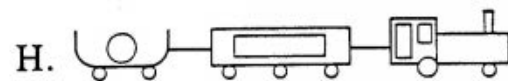


Class 1: "Wheels on all cars have the same color."

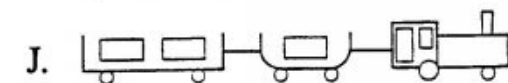
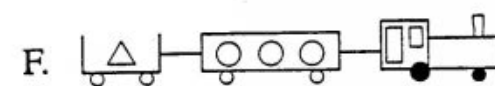


Class 2: "Wheels on all cars do not have the same color."





“These trains are carrying toxic chemicals.”



“These trains are not carrying toxic chemicals.”

