# Learning Based on Conceptual Distance

YVES KODRATOFF AND GHEORGHE TECUCI

*Abstract*—**In this paper we present a new approach to concept learning from examples and concept learning by observation, which is based on an intuitive notion of conceptual distance between examples (concepts) and combines symbolical and numerical methods. Our approach is supported by the observation that very different examples generalize to an expression that is very far from each of them, while identical examples generalize to themselves. Therefore, a generalization of two examples, as well as the process of obtaining this generalization, represents indications of the conceptual distance between the examples. Following this idea we propose some domain independent and intuitively justified estimates for the conceptual distance. Usually however, a set of examples may be characterized by several generalizations, each suggesting a certain conceptual distance. The minimum of these is taken as the estimation of the real conceptual distance. Moreover, the corresponding generalization is recommended as the one to be made by the learning system because this generalization has the desirable property of reflecting the greatest number of common features of the examples. We also present a hierarchical conceptual clustering algorithm which groups objects so that to maximize the cohesiveness (a reciprocal of the conceptual distance) of the clusters. We further show that conceptual clustering may improve learning from complex examples describing objects and the relations between them. The idea is that learning good generalizations of such examples requires matching the most similar objects which, in turn, requies a clustering of these objects. Finally we present a methodology of learning hierarchies of prototype objects which is a step toward automating the construction of knowledge bases for expert systems.**

*Index Terms*—**Conceptual clustering, conceptual cohesiveness, conceptual distance, dissimilarity, generalization, knowledge acquisition, knowledge bases, preferable generalization, prototypes, robotics.**

## I. Introduction

Machine learning may be defined as any process by which a computer increases its knowledge and improves its skills. One of the basic types of learning is inductive learning, that is, learning by generalizing specific facts or situations. Inductive learning has received considerable attention in artificial intelligence [3], [4], [9], [10], [11], [15]. Two different kinds of inductive learning are learning from examples and conceptual clustering.

In *concept learning from examples*, the learning system is presented with independent instances representing a certain class, and the task is to induce a general description of the class. The instances can be specific physical objects, actions, processes, images, etc. Let us suppose that they are different cars (CITROEN, RENAULT, OPEL, etc.). In this case, the system's task is to learn the concept of car, represented by what is common to all the given examples (objects with four wheels, used to transport people, etc.). Having formed such a concept, the system will be able to recognize other objects as being or not being cars, as they have or have not the properties of the car concept.

Manuscript received December 8, 1986; revised February 24, 1988. This work was supported by PRC-GRECO "Intelligence Artificielle" and the Romanian CNST.

Y. Kodratoff is with the Laboratoire de Recherche en Informatique, Université de Paris-Sud, 91405 Orsay Cedex, France.

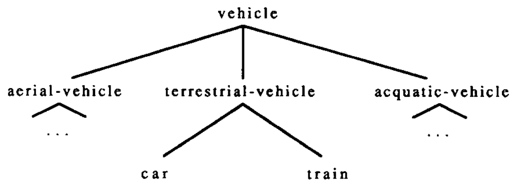
G. Tecuci is with the Research Institute for Computers and Informatics, 71316, Bd. Miciurin 8–10, Sector 1, Bucharest, Romania.

IEEE Log Number 8823854.

In *conceptual clustering*, the learner is also presented with a set of examples, but these examples are no longer said to represent the same class. In this case, the learner has to solve two problems:

- the *aggregation problem* of distinguishing classes (defined as extensionally enumerated sets of objects) into which the examples can be grouped;
- the *characterization problem* of inducing an intentional description for each class.

The examples presented to the learner could be, for instance, descriptions of specific cars, ships, airplanes, or trains, and the system would learn the following concepts:



As defined, the characterization problem is very similar with the problem of learning from examples. Conceptual clustering processes must address this problem since the quality of the clustering is dependent on the description of the clusters (the simplicity of these descriptions, the map between the descriptions and the clusters they cover, etc. [12]).

Although nobody is claiming that the aggregation and characterization problems should be independent, the present conceptual clustering algorithms [5] first solve the aggregation problem, and then use the methods of learning from examples to obtain a description for each cluster. The so obtained descriptions are further used to estimate the quality of the clustering and may suggest to search for another clustering.

In this paper we present a new approach to concept learning from examples and concept learning by observation, which is based on a intuitive notion of *conceptual distance* [12] between examples (concepts) and combines symbolical and numerical methods.

Our approach is supported by the observation that very different examples generalize to an expression that is very far from each of them, while identical examples generalize to themselves. Therefore, a generalization of two examples, as well as the process of obtaining this generalization, represent indications of the conceptual distance between the examples. Following this idea, in Section III, we propose some domain independent and intuitively justified measures for the conceptual distance.

However, the process of obtaining a generalization of a set of examples is not a deterministic one. Several generalizations are possible, and each suggests a certain conceptual distance between them. Therefore, we propose to estimate the real distance by the minimum of these distances. Moreover, the corresponding generalization is recommended as the one to be made by the learning system since this generalization has the desirable property of reflecting the most commonalties between the examples.

In Section IV, we present a conceptual clustering algorithm which is based on the reciprocal of the conceptual distance, called *conceptual cohesiveness* [12], and on a partial ordering defined on conceptual cohesiveness. The main feature of the conceptual cohesiveness is that it takes into consideration not only the properties of the individual objects, but also their relationship to other objects and, most importantly, their relationship to some predefined concepts characterizing object collections.

To cluster a set of examples $E1, \cdots, En$, our algorithm first looks for the two examples $Ei$, $Ej$ for which the conceptual cohesiveness is maximum. These examples form the seed of a cluster. A new example $Ek$ is added to this cluster only if the conceptual cohesiveness of the set $\{Ei, Ej\}$ is not greater than the conceptual cohesiveness of the set $\{Ei, Ej, Ek\}$.

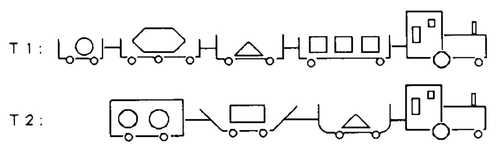
While, in general, only conceptual clustering is based on learning from examples, in our approach learning from *complex* examples is also based on conceptual clustering. Here by a complex example we mean an example describing several objects and the relations between them. The idea is that learning good generalizations of complex examples requires matching the most similar objects which, in turn, requires a clustering of these objects. This method is presented in Section V. For instance, it should be of use in scene analysis where the recognition of each individual scene component and the recognition of the whole scene are dependent of each other.

We consider that the approach presented in this paper is also significant to automatic knowledge acquisition for expert systems and, in Section VI, we present a methodology for generating hierarchies of prototype objects.

## II. Concept Learning from Examples

*Concept learning from examples* means forming a general description (concept) of a class of objects given a set of objects (examples) from this class.

We assume that both the examples and the concepts are described in the same representation language, as conjunctions of literals. For instance, Fig. 1 represents two



(a)

T1:
(infront C1 C2) (infront C2 C3) (infront C3 C4) (infront C4 C5)
(length long C1) (length long C2) (length short C3) (length long C4) (length short C5)
(car-shape machine C1) (car-shape open-rectng C2) (car-shape sloping-top C3)
(car-shape open-rectng C4) (car-shape open-rectng C5)
(contains C2 L2) (contains C3 L3) (contains C4 L4) (contains C5 L5)
(load-shape square L2) (load-shape triangle L3) (load-shape hexagon L4)
(load-shape circle L5) (nrpts-load 3 L2)
(nrpts-load 1 L3) (nrpts-load 1 L4) (nrpts-load 1 L5)
(nr-wheels 2 C1) (nr-wheels 2 C2) (nr-wheels 2 C3) (nr-wheels 3 C4) (nr-wheels 2 C5)

T2:
(infront C6 C7) (infront C7 C8) (infront C8 C9)
(length long C6) (length short C7) (length short C8) (length short C9)
(car-shape machine C6) (car-shape U-shape C7) (car-shape open-trapeze C8)
(car-shape closed-rectng C9)
(contains C7 L7) (contains C8 L8) (contains C9 L9)
(load-shape triangle L7) (load-shape rectangle L8) (load-shape circle L9)
(nrpts-load 1 L7) (nrpts-load 1 L8) (nrpts-load 2 L9)
(nr-wheels 2 C6) (nr-wheels 2 C7) (nr-wheels 2 C8) (nr-wheels 2 C9)

(b)

Fig. 1. The first two trains from Michalski's train problem.

examples of toy trains. They are taken from the famous Michalski's train problem [13], which consists in finding a common characterization of a set of such trains.

As can be seen in Fig. 1, each train is described as a conjunction of literals of the form $(p\ a1 \cdots an)$, where $p$ is a predicate and $a1, \cdots, an$ are the arguments of the predicate. For instance:

(car-shape open-rectng $C2$)

means that the shape of the car $C2$ is an open rectangle, and

(length long $C2$)

means that the length of $C2$ is long.

The argument of a predicate, also called term, may be a constant, a variable, or $f(t1 \cdots tn)$, where $f$ is a function and $t1, \cdots, tn$ are terms.

For each variable a domain is defined, containing all possible values the variable can take. As in [12], we distinguish among nominal (categorical), linear (quantitative), and structured variables, whose domains are unordered, totally ordered, and graph-oriented sets, respectively. Structured variables represent generalization hierarchies of related values as, for instance, the one shown in Fig. 2.

The predicates may be related by theorems as, for instance, the following one:

$$\forall x\ \forall y\ \forall z,\ (\text{contains}\ x\ y)\ \&\ (\text{contains}\ y\ z) \Rightarrow (\text{contains}\ x\ z)$$

We shall use the predicates from Michalski's train problem to present our approach. For instance, two simple examples are shown in Fig. 3.

The first example describes the car A1 as having the shape open rectangle and being short. The second exam-

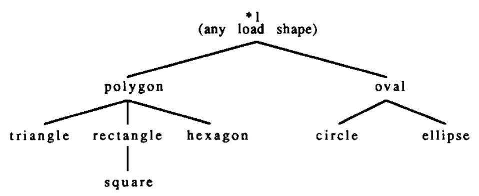


Fig. 2. A generalization hierarchy for the load shapes in Fig. 1.

E1: (car-shape open-rectng A1) (length short A1)
E2: (car-shape open-trapeze A2) (length short A2)

Fig. 3. Two simple examples.

ple describes the car A2 as having the shape open trapeze and being also short.

A *generalization* of an example is an expression which "describes" a set containing the example. That is, by replacing the variables of the generalization by suitable constants, one finds back the example (see precise definition below). For instance, the following is a generalization of $E1$:

$G$: (car-shape open-rectng $x$) (length $y$ $x$)

It describes a set of open rectangle cars of any length. One finds back $E1$ by replacing "$x$" by "A1" and "$y$" by "short."

A *generalization* of several examples $E1, \cdots, En$ is an expression which describes a set containing all these examples. For instance, the following is a generalization of $E1$ and $E2$:

$G1$: (car-shape $z$ $x$) (length short $x$)

It describes a set of short cars of any shape.

A key characteristic of the concept learning from example problem is that there is an important structure inherent to the language used to represent the concepts. This structure is based on the relation *less-general-than*, defined as follows.

Given two generalizations, $G1$ and $G2$, one says that $G1$ is *less-general-than* $G2$ if and only if the set of instances of $G1$ is included into the set of instances of $G2$.

Notice however that the above definition is extensional, based upon the instance sets that the generalizations represent. In order for the *less-general-than* relation to be practically computable by a computer program, it must be possible to determine whether $G1$ is less-general-than $G2$ by examining the descriptions of $G1$ and $G2$, without computing the (possibly infinite) sets of instances that they match. In our approach, this computational definition is based on the notion of substitution, as defined in the following.

A *substitution* has the following form:

$$\sigma = (x1 \leftarrow t1, \cdots, xn \leftarrow tn)$$

where each $xi$ $(i = 1, \cdots, n)$ is a variable and each $ti$ $(i = 1, \cdots, n)$ is a term.

If "$l$" is a literal then "$\sigma \circ l$" is the literal obtained by substituting each "$xi$" from "$l$" with "$ti$."

Using substitutions one may compare the generality of logical formulas [7].

One says that the term $t1$ is *less general than* the term $t2$ if and only if there exist $t1'$, $t2'$, and a substitution $\sigma$ such that:

$$t1' = t1$$
$$t2' = t2$$
$$\sigma \circ t2' = t1'$$

That is, one uses the theorems and the properties of the representation language to rewrite the terms so that to become directly comparable.

Let us consider, for instance, the following terms:

$$t1 = \text{VOLUM-CUBOID } (5\ 5\ 7)$$
$$t2 = \text{VOLUM-CUBOID } (x\ y\ x)$$

Using the property of commutativity of the arguments of the function VOLUM-CUBOID one may rewrite $t2$ as:

$$t2' = \text{VOLUM-CUBOID } (x\ x\ y)$$

Next one finds the substitution

$$\sigma = (x \leftarrow 5, y \leftarrow 7)$$

such that $\sigma \circ t2' = t1$. Therefore, one may say that $t1$ is less general than $t2$.

The literal $l1 = (p1\ t11 \cdots t1n)$ is *less general than* the literal $l2 = (p2\ t21 \cdots t2n)$ if and only if there exist $l1'$, $l2'$, and $\sigma$ such that:

$$l1' = l1$$
$$l2' = l2$$
$$\sigma \circ l2' = l1'$$

Let us now consider two conjunctive formula

$$X = X1 \ \&\ X2 \ \&\ \cdots \ \&\ Xn$$
$$Y = Y1 \ \&\ Y2 \ \&\ \cdots \ \&\ Ym$$

where $Xi$ $(i = 1, \cdots, n)$ and $Yj$ $(j = 1, \cdots, m)$ are literals.

$X$ is *less general than* $Y$ if and only if there exist $X'$, $Y'$, and $\sigma$ such that:

$$X' = X, \quad X' = X1' \ \&\ X2' \ \&\ \cdots \ \&\ Xp'$$
$$Y' = Y, \quad Y' = Y1' \ \&\ Y2' \ \&\ \cdots \ \&\ Yq'$$
$$\forall i, \quad 1 \le i \le q \ \exists j, \quad 1 \le j \le p \text{ such that}$$
$$\sigma \circ Yi' = Xj'.$$

Otherwise stated, one transforms the formula $X$ and $Y$, using the theorems and the properties of the representation language, so that for each literal from $Y'$ there exists a literal in $X'$ which is less general.

For instance

$E1$: (car-shape open-rectng $C1$)

is less general than

$G$: (car-shape $z$ $x$) (length $y$ $x$)

Indeed, one may use the theorem that any car has a length

$$\forall v \, \exists u \, (\text{length } u \; v) = \text{TRUE}$$

and may rewrite $E1$ as

$E1$: (car-shape open-rectng $C1$) (length $u$ $C1$).

Now, there exists the substitution

$$\sigma = (z \leftarrow \text{open-rectng}, x \leftarrow C1, y \leftarrow u)$$

such that "$\sigma \circ G = E1$". Therefore, $E1$ is less general than $G$.

Let us consider again the examples in Fig. 3. Some of their generalizations are the following ones:

$G1$: (car-shape $z$ $x$) (length $y$ $x$)
$G2$: (car-shape $z$ $x$) (length short $x$)
$G3$: (car-shape $z$ $x$) (length $y$ $t$)
$G4$: (car-shape $x$ $y$)
$G5$: (length short $x$)
$G6$: (length $x$ $y$)

A learning system will always have the problem of choosing among the competing generalizations. We define the notion of *preferable* generalization as being the generalization which is less general than all the other generalizations.

In our case, the *preferable* generalization is $G2$ since it is less general than all the other generalizations. For instance, $G2$ is less general than $G1$ because there is the substitution $\sigma = (y \leftarrow \text{short})$ such that $\sigma \circ G1 = G2$.

The notion of *preferable* generalization is relative to the knowledge about the learning universe and cannot be considered absolute. New knowledge may lead to an improvement. For instance, let us consider that we have acquired new knowledge about car shapes. Suppose that this knowledge is expressed by the hierarchy in Fig. 4.

In this case

$G7$: (car-shape open-top $x$) (length short $x$)

is also a generalization of $E1$ and $E2$. Since $G7$ is less general than $G2$ (there is $\sigma = (z \leftarrow \text{open-top})$ such that "$\sigma \circ G2 = G7$"), $G7$ is less general than all the other generalizations. Therefore $G7$ is the preferable generalization in the new context.

Given a set of generalizations, it is most probable that there is no generalization that is the least general. For instance, the set $\{G1, G3, G4, G5, G6\}$ does not contain a least general expression.

Given two expressions $G1$ and $G2$, if neither $G1$ *is less general than* $G2$ nor $G2$ *less general than* $G1$, then $G1$ and $G2$ are said to be *incomparable* from a generalization point of view.

In a given learning situation, there are many generalizations which are incomparable, and the main difficulty

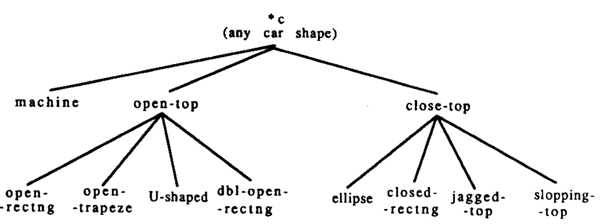


Fig. 4. A generalization hierarchy for the shapes of the cars in Fig. 1.

of learning is to choose the right one [6]. Therefore, we have to look for another, more relaxed, definition of the *preferable* generalization.

Let us notice that, if $Gi$ is the *preferable* among $G1, \cdots, Gn$, then the set of instances of $Gi$ is included into the set of instances of each of these generalizations. It follows that the number of instances of $Gi$ is less than the number of instances of any other generalization. Based on this observation we may compare two generalizations $G1$ and $G2$ even if they are incomparable from a generalization point of view. We shall say that $G1$ is preferable if the number of instances of $G1$ is less than the number of instances of $G2$.

But this definition is still unsatisfactory for the simple reason that learning is not an isolated aim. One is learning in a given universe, with a well-defined goal in that universe. As a consequence, a generalization has also to point to that essential common properties of the examples. Therefore, a *good* generalization is not one which represents many of the properties common to the examples, but that one which represents many of the *important* properties common to the examples.

Let us consider, for instance, the case of a robot which learns concepts from examples representing physical objects. Each object is described by specifying the actions which could be performed on it, the relations which could be established between this object and other objects, the shape of the object, its color, etc. Although all these properties are relevant for a robot their relative importance depends on robot goals. If the robot intends to use the learned concepts for action planning, then the action and relation properties are to be considered more important than the color, for instance. On the other hand, if the robot intends to use the learned concepts for recognizing objects, then the color property should be considered more important. This problem is treated in detail in Section VI.

In our approach, the relative importance of the predicates must be defined by the teacher. One way to do it is to associate a weight to each predicate. Based on the weights of the predicates we could estimate the relevance (value) of a generalization as the sum of the weights of the predicates included into the generalization. We may therefore consider that the *preferable* generalization is the one which maximizes the relevance and minimizes the number of instances.

We may now informally define the *preferable* generalization of the examples $E1, E2, \cdots, En$, as follows:

- if there is a generalization $Gi$ which is less-general-than the other generalizations of the examples then *Gi* is the *preferable* generalization;

• if *Gi*1, $\cdots$, *Gim*, are all the incomparable generalizations of the examples, then consider the *preferable* generalization the one which is the *most relevant* (contains the most important predicates) and has the *least number of instances.*

In the following section we shall propose a more computational definition of the *preferable* generalization, definition based on domain-independent and intuitively justified heuristics.

## III. Conceptual Distance and Conceptual Cohesiveness

### A. *The General Approach*

Given two descriptions, we could notice similarities and dissimilarities. For instance, the descriptions:

$E1$: (car-shape open-rectng $B1$)
$E2$: (car-shape U-shaped $B2$)

are similar because both are characterized by the predicate *car-shape* and each car-shape is *open-top* (see Fig. 4), but the first shape is *open-rectng*, while the second one is *U-shaped.*

If we could estimate the similarity $S(E1, E2)$ and the dissimilarity $D(E1, E2)$ between the descriptions $E1$ and $E2$, then we could estimate the conceptual distance between $E1$ and $E2$ by a function of $S$ and $D$. This function would quantify the contribution of $S$ and $D$ to the conceptual distance.

Since a generalization of two examples is able to reveal subtle commonalities between these examples it seems to be a suitable means of estimating their conceptual distance. Indeed, let us notice that very different examples generalize to a *general* expression that is very far from each of them, while identical examples generalize to themselves. Moreover, we want to take into account that the fewer changes are made to the examples in order to obtain a generalization, the greater the similarities and the less the dissimilarities are.

Here we shall propose an estimation of the conceptual distance which is based on the learning algorithm developed at LRI [7]. This algorithm uses the principle of structural matching: the examples are successively transformed until they acquire approximately the same form. Then the generalization is obtained by retaining only the common features.

To illustrate this algorithm let us consider the following two examples:

$E1$: (car-shape open-rectng $C1$) (contains $C1$ $L1$)
(car-shape open-trapeze $C3$)

$E2$: (car-shape U-shaped $C2$) (length short $C2$)

The first example represents two cars, an open rectangle one containing an object and an open trapeze one. The second example represents a short U-shaped car (see Fig. 4).

The algorithm first rewrites the examples revealing their common features:

$E1$: (car-shape open-top $X1$) (contains $X1$ $L1$)
(car-shape open-trapeze $C3$)
$(X1 \leftarrow C1)$

$E2$: (car-shape open-top $X1$) (length short $X1$)
$(X1 \leftarrow C2)$

Next, it will use the theorems of the representation language in order to reveal in one example features exhibited by the other. Such a theorem expresses, for instance, the fact that any object has a length:

$$\forall z \, \exists t \, (\text{length } t \; z) = \text{TRUE}$$

Using this theorem one rewrites the two examples as follows:

$E1$: (car-shape open-top $X1$) (length $Y1$ $X1$) (contains $X1$ $L1$)
(car-shape open-trapeze $C3$)
$(X1 \leftarrow C1,\ Y1 \leftarrow t)$

$E2$: (car-shape open-top $X1$) (length $Y1$ $X1$)
$(X1 \leftarrow C2,\ Y1 \leftarrow \text{short})$

If one example exhibits a certain feature more times than the other, one uses the idempotency of the AND operator to make the feature appear the same number of times:

(car-shape open-top $X1$)
= (car-shape open-top $X1$) &(car-shape open-top $X1$)

Therefore, the two expressions are farther rewritten as follows:

$E1$: (car-shape open-top $X1$) (length $Y1$ $X1$) (contains $X1$ $L1$)
(car-shape open-top $X2$)
$(X1 \leftarrow C1,\ Y1 \leftarrow t,\ X2 \leftarrow C3)$

$E2$: (car-shape open-top $X1$) (length $Y1$ $X1$)
(car-shape open-top $X2$)
$(X1 \leftarrow C2,\ Y1 \leftarrow \text{short},\ X2 \leftarrow C2)$

When no other common features may be revealed, one simply drops the differences between the two expessions and obtains the generalization of the initial examples:

$G(E1,E2)$: (car-shape open-top $X1$) (length $Y1$ $X1$)
(car-shape open-top $X2$)
(may-be-the-same $X1$ $X2$)

Let us notice that the operations made in order to obtain a structural matching and a generalization of the examples are indications of similarities and dissimilarities between these examples.

The predicates of $E1$, $E2$, and $G(E1, E2)$ could be classified in four categories (thus obtaining four lists of predicates), as follows:

*Common:* Predicates from $G(E1, E2)$ which were initially present in $E1$ and $E2$:

{ (car-shape open-top $X1$) }.

*Theorems:* Predicates introduced in $G(E1, E2)$ by using the theorems of the representation language:

{ (length $Y1$ $X1$) }.

*Idempotency:* Predicates introduced in $G(E1, E2)$ by using the idempotency of the **AND** operator:

{ (car-shape open-top $X2$) }.

*Dropped:* Predicates dropped from $E1$ and $E2$, in order to obtain $G(E1, E2)$:

{ (contains $X1$ $L1$) }.

The general intuition is that each such type has a specific influence to the estimation of the similarities and dissimilarities between the examples.

In the following sections we shall propose and justify measures for each type of predicates.

## B. Common Predicates

Let us consider the following four examples:

$E1$: (car-shape open-rectng $C1$)
$E2$: (car-shape U-shaped $C2$)
$E3$: (car-shape open-rectng $C3$)
$E4$: (car-shape ellipse $C4$)

Since each of these four descriptions is characterized by the same predicate, the conceptual distance between them is exclusively determined by the distance between their arguments.

Intuitively:

$$\text{distance}(E1, E3) < \text{distance}(E1, E2) < \text{distance}(E1, E4)$$

Let us also consider the following generalizations (see Fig. 4):

$G(E1,E3)$: (car-shape open-rectng $X1$)
$G(E1,E2)$: (car-shape open-top $X2$)
$G(E1,E4)$: (car-shape $*c$ $X3$)

Let us notice that *open-rectng, open-top,* and $*c$ are all values from the structured domain in Fig. 4, and that *open-rectng* is less general than *open-top* which in its turn is less general than $*c$. While *open-rectng* is an instance, *open-top* is a generalization with 4 instances and $*c$ is a generalization with 9 instances.

We could define the *degree of generality* of an argument, as the ratio of the number of argument's instances to the total number of instances from argument's domain, that is:

$$g(a) = \frac{\text{number of instances of ``}a\text{''}}{\text{number of instances of the domain of ``}a\text{''}}.$$

For example:

$$g(\text{open-rectng}) = 0. \quad g(\text{open-top}) = 0.44 \quad g(*c) = 1.$$

This definition applies also to the so-called linear (quantitative) variables [12].

Let us notice that there is no dissimilarity between $E1$ and $E3$. Indeed, $C1$ and $C3$ are just different names for the same entity (i.e., the car) and $X1$, in the expression $G(E1, E3) = (\textit{car-shape open-rectng } X1)$, is just another name for the car. $X1$ denotes a definite object and, therefore, $g(X1) = 0$.

Intuitively, the more similar two descriptions containing only common predicates are, the less general are the arguments of their generalizations. Also, the more dissimilar two such descriptions are, the more general are these arguments.

All the arguments of a predicate being *a priori* of the same importance, one should propose an estimation for the similarity (dissimilarity), between two examples $E1$ and $E2$, by a function of the mean degree of generality of the arguments of the generalization $G(E1, E2)$.

Let us now consider the following three examples:

$E5$: (color red $C5$)
$E6$: (color blue $C6$)
$E7$: (size big $C7$)

and the generalization:

$G(E5,E6)$: (color $*d$ $X8$)

It is quite obvious that $E5$ is more similar to $E6$ than to $E7$, in spite of the fact that the arguments of $G$ are variables. The simple fact that $E5$ and $E6$ are characterized by the same predicate makes them similar. Therefore, the similarity estimation function should also indicate a certain similarity between two examples even when all the arguments of $G$ are variables, but both examples are characterized by the same predicate.

To sum up, let us consider two examples

$E1$: $(P\ a1\ a2 \cdots an)\ \& \cdots$
$E2$: $(P\ b1\ b2 \cdots bn)\ \& \cdots$

and their generalization

$G(E1,E2)$: $(P\ c1\ c2 \cdots cn)\ \& \cdots$

Also let $g(ci)$ be the generality degree of the argument $ci$.

Then we propose to estimate the contribution of $P$ to the dissimilarity and similarity between $E1$ and $E2$ by the following functions on the degree of generality of the arguments of $P$:

$$D(E1, E2, P) = 0.5\left(\sum g(ci)\right)/n$$

$$S(E1, E2, P) = 1 - D(E1, E2, P).$$

That is, we take the total contribution of a predicate $P$ to the conceptual distance between $E1$ and $E2$ as being equal to 1, and we distribute it between similarity and dissimilarity in accordance with the generality degree of its arguments.

### C. Dropped Predicates

Let us consider two examples and their generalization:

$E1$: (car-shape open-rectng $C1$) & (contains $C1$ $L1$)
$E2$: (car-shape U-shaped $C2$)
$G(E1,E2)$: (car-shape open-top $X1$)

In order to obtain a generalization of $E1$ and $E2$ one has to drop the predicate *contains* because it represents a feature of $E1$ which has nothing in common with any feature of $E2$. Therefore, this predicate is an indication of dissimilarity between the two examples.

Therefore we propose to estimate the contribution of a dropped predicate $P$ to the estimation of the dissimilarity and similarity between $E1$ and $E2$ as follows:

$$D(E1, E2, P) = 1$$

$$S(E1, E2, P) = 1 - D(E1, E2, P) = 0.$$

### D. Predicates Introduced by Idempotence

We consider that the necessity of using idempotency of the logical "AND" (for computing a generalization of two descriptions) is an indication of dissimilarity. But this dissimilarity has to be considered less than in the case of dropping predicates. Indeed, the predicate involved is present in both descriptions (that is, both descriptions have the property expressed by the predicate) but a different number of times.

Let us consider the following examples:

$E1$ = (length short $C1$)

$E2$ = (length short $C2$) (length short $C3$)

$E3$ = (length short $C4$) (length long $C5$).

We could obtain the following generalizations (by applying idempotency in the first example):

$G(E1, E2)$ = (length short $X1$) (length short $X2$)

$G(E1, E3)$ = (length short $X3$) (length *$l$ $X4$)

Intuitively, one sees that distance$(E1, E2) <$ distance$(E1, E3)$.

$G(E1, E2)$ and $G(E1, E3)$ differ only by the predicate which was introduced by idempotency. The only significant dissimilarity between (length short $X2$) and (length *$l$ $X4$) consists in the generality degree of the first argument: $g(\text{short}) = 0$, $g(*l) = 1$.

As in the case of the common predicates, we could estimate the dissimilarity, due to a predicate introduced by idempotency, by the mean of the degree of generality of its arguments.

For intuitive, but also for formal reasons, we cannot accept any contribution of idempotency to the similarity estimation. Indeed, if idempotency would contribute to similarity estimation, then the similarity of two descriptions would be undefined and arbitrary since idempotency can be applied any number of times.

Therefore, the contribution to the estimation of the dissimilarity and similarity between $E1$ and $E2$ of the predicate $P$, introduced in $G(E1, E2)$ by idempotency, is taken as follows:

$$D(E1, E2, P) = 0.5\left(\sum g(ci)\right)/n$$

$$S(E1, E2, P) = 0.$$

The generalization algorithm will always prefer idempotency to dropping, but will also try to use idempotency as few times as possible.

### E. Predicates Introduced by Theorems

In principle, these predicates should be treated as the common predicates. Let us consider, for instance, the examples:

$E1$: (on $A$ $B$)
$E2$: (near $C$ $D$).

We may use the theorem "$\forall x\ \forall y$, (on $x$ $y$) $\rightarrow$ (near $x$ $y$)" and rewrite the first example as:

$E1$: (on $A$ $B$) (near $A$ $B$).

In this way we have revealed a common feature of $E1$ and $E2$: both represent two objects which are *near* one another.

However, we must take care avoiding counting several times the contribution of a predicate to the similarity and dissimilarity estimation, as shown by the following example.

Let us consider, for instance, the examples

$E1$: (on $A$ $B$)
$E2$: (on $C$ $D$)

and their rewritten form

$E1$: (on $A$ $B$) (near $A$ $B$)
$E2$: (on $C$ $D$) (near $C$ $D$).

In this case we should consider the *on* predicate as the only predicate *common* to the examples. Adding *near* would almost mean counting several times the *on* predicate.

### F. Relative Importance of the Predicates

A system learning concepts from examples is supposed to have its own goals that are intended to be achieved by using the learned concepts. Since these goals are also known by the teacher supplying the examples, it is reasonable to suppose that the examples specify only the properties relevant to the system's goals. Even in such a case however some properties may be regarded as more important than others.

We assume that the relative importance of the properties is given by the teacher in the form of a weight associated to each predicate. Therefore, the previously estimated contributions of a predicate, to the similarity and the dissimilarity of two descriptions will be multiplied by the weight of the predicate. For instance, if $P$ is a common predicate with the weight $\omega$, then its contribution to the dissimilarity and similarity estimation will be taken as:

$$D(E1, E2, P) = 0.5\omega\left(\sum g(ci)\right)/n$$
$$S(E1, E2, P) = 1 - D(E1, E2, P).$$

## G. Estimation of Conceptual Distance and Conceptual Cohesiveness

Let us consider two examples $E1$, $E2$, and one of their generalizations $G(E1, E2)$. As shown in the previous sections, one may estimate the contribution to similarity and dissimilarity of each involved predicate. By adding these estimations, one obtains the total estimation of similarity $S(E1, E2, G)$ and the total estimation of dissimilarity $D(E1, E2, G)$. These estimations depend of course on the generalization $G(E1, E2)$. Another generalization $G'(E1, E2)$ would produce different estimations $D'(E1, E2, G')$ and $S'(E1, E2, G')$.

Having estimated the similarity and the dissimilarity between $E1$ and $E2$ (corresponding to $G(E1, E2)$), one is able to estimate the conceptual distance between $E1$ and $E2$ (corresponding to $G$), as a function of $D$ and $S$. Hereafter we shall consider the following distance function:

$$f(E1, E2, G) = D(E1, E2, G)/S(E1, E2, G).$$

Using $G'(E1, E2)$ instead of $G(E1, E2)$, one obtains another estimation of the conceptual distance between $E1$ and $E2$:

$$f(E1, E2, G') = D(E1, E2, G')/S(E1, E2, G').$$

We take, as the conceptual distance between $E1$ and $E2$, the minimum of $f(E1, E2, G)$ over all possible generalizations of $E1$ and $E2$:

$$\text{conceptual-distance}(E1, E2) = \min_{G(E1,E2)} \left\{ f\big(E1, E2, G(E1, E2)\big) \right\}.$$

Moreover, *the generalization for which f is minimum is recommended as the concept to be learned* from $E1$ and $E2$ since this generalization has the desirable property of revealing the greatest number of common features between the examples and of being the least general among the generalizations revealing the same amount of common features.

Since the definition of the conceptual distance is based on the generalizations of the examples, this definition applies for any number of examples.

Let us consider $n$ examples $E1, E2, \cdots, En$ and $G(E1, E2, \cdots, En)$, one of their generalizations. Based on this generalization one can compute the four lists:

*CM:* Predicates from $G(E1, \cdots, En)$ which were initially present in $E1, \cdots, En$.

*TH:* Predicates from $G(E1, \cdots, En)$ which were introduced in at least one example, by using the theorems of the representation language (but not idempotency).

*ID:* Predicates from $G(E1, \cdots, En)$ which were introduced in at least one example, by using the idempotency of the **AND** operator;

*DR:* Predicates dropped from $E1, \cdots, En$ in order to obtain $G(E1, \cdots, En)$.

Further on, using the above four lists, one can estimate the similarity $S(E1, \cdots, En, G)$, the dissimilarity $D(E1, \cdots, En, G)$, and the conceptual distance $f(E1, \cdots, En, G)$. The minimum of $f$ taken over all possible generalizations of $E1, \cdots, En$ is the conceptual distance between $E1, \cdots, En$:

$$\text{conceptual-distance}(E1, \cdots, En) = \min_{G(E1,\cdots,En)} f(E1, \cdots, En, G).$$

The reciprocal of the conceptual distance is called the *conceptual cohesiveness* of the set $\{E1, \cdots, En\}$. The more similar and less dissimilar are the examples, the greater is their conceptual cohesiveness.

In the next section we shall present a hierarchical clustering algorithm based on the above notion of conceptual cohesiveness.

## IV. Clustering by Generalizing

Conceptual clustering was introduced by Michalski and Stepp [12] as an extension of processes of numerical taxonomy (a collection of methods used to form classification schemes over data sets) [19]. The main quality of the conceptual clustering is that it is able to capture the "Gestalt properties" of object clusters, that is, properties that characterize a cluster as a whole and are not derivable from properties of individual entities.

To illustrate this idea, let us consider the example taken from [12] shown in Fig. 5.

A person considering this figure would typically describe the observed points as representing two diamonds. Thus, the points $A$ and $B$, although closer to each other than to other points, are placed into different clusters.

CLUSTER/2 [12] is a conceptual clustering algorithm able to make such classifications. In this section we present another clustering algorithm which is based on the conceptual cohesiveness defined in the previous section.

The goal of our clustering algorithm is to group the examples in such a way so that to maximize the conceptual cohesiveness of the clusters.

Our algorithm is based on the following observations: if $\{E1, E2, \cdots, En\}$ is a cluster with *high* conceptual cohesiveness and $\{Ei, \cdots, Ek\}$ is a subset of this cluster, then the conceptual cohesiveness of $\{Ei, \cdots, Ek\}$ is a good approximation of the conceptual cohesiveness of $\{E1, E2, \cdots, En\}$. Let us now suppose that we add a new example $Ea$ to this cluster. If the conceptual cohesiveness does not decrease significantly, then $Ea$ be-

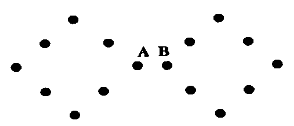


Fig. 5. An illustration of conceptual clustering.

longs to the same concept as $\{E1, E2, \cdots, En\}$. Otherwise, $\{E1, E2, \cdots, En\}$ and $Ea$ belong to different concepts.

To illustrate this idea, let us consider that each example represents either a man or a woman. The conceptual cohesiveness of each subset of women is approximately the same with the conceptual cohesiveness of the set of all women. However, adding a man to such a set would significantly decrease the conceptual cohesiveness of the set.

To make this approach operational, one has to be able to determine when a conceptual cohesiveness decreases significantly. This is analogous to the definition of what we call *resolution*, a measure that indicates the minimum distance that has to exist between two elements to be perceived as distinct. Our claim is that, in a given domain, one could experimentally determine the resolution by measuring the distances between concepts thought as distinct. We propose to express this resolution as a threshold $\mu$, where $0 < \mu \leq 1$, and to state that the conceptual cohesiveness of two sets $S1$ and $S2$ are different if and only if

- cohesiveness$(S1)$ is less than $\mu$*cohesiveness$(S2)$

or

- cohesiveness$(S2)$ is less than $\mu$*cohesiveness$(S1)$.

Let $\{E1, \cdots, Et\}$ be the examples to be clustered. The clustering algorithm first determines the pair of examples $\{Ep, Eq\}$ for which the conceptual cohesiveness is maximum and takes it as the *seed* of a cluster. A new example is introduced into this cluster only if it does not decrease the cohesiveness of the cluster below the cohesiveness of $\{Ep, Eq\}$.

Once a cluster is completed, it replaces, in the set of examples, all the examples it contains, and the process is restarted with the new examples.

In greater detail the clustering algorithm is the following one:

*Step 1: Ask for the resolution of the application domain.*

The resolution of the application domain is defined by the user as a threshold $\mu$, where $0 < \mu \leq 1$.

Given two sets $S1$ and $S2$, cohesiveness$(S1) <$ cohesiveness$(S2)$ if and only if cohesiveness$(S1)$ is less than $\mu$*cohesiveness$(S2)$.

If neither "$c(S1) < c(S2)$" nor "$c(S2) < c(S1)$" we say that $c(S1)$ and $c(S2)$ are incomparable, were by $c(Si)$ we denoted the cohesiveness of $Si$.

*Step 2: Compute the conceptual cohesiveness of each pair of examples.*

Let $E = \{E1, \cdots, En\}$ be the set of examples.

For each pair $\{Ep, Eq\}$ compute its cohesiveness by using the generalizations $G(Ep, Eq)$ and the corresponding quadruples

$$L(Ep, Eq) = (CO(Ep, Eq),\ TH(Ep, Eq),\ ID(Ep, Eq),\ DR(Ep, Eq)),$$

as presented in Section III.

*Step 3: Choose a seed of the clustering.*

Determine the pair $\{Ep, Eq\}$ for which $c(Ep, Eq)$ is maximum. If several such pairs exist, choose one of them.

Let $\{Ep, Eq\}$ be the chosen pair. It is called the *seed* of the clustering.

$G(Ep, Eq)$ is one of the *most* relevant concepts among those represented by pairs of examples.

Let $M = \{Ep, Eq\}$.

We shall discover a first cluster by introducing in $M$ other elements from $E$.

*Step 4: Determine the examples which could be members of the cluster represented by the chosen seed.*

That is, determine the set:

$$T = \{Ek \mid \text{"}c(Ep, Eq) \text{ and } c(Ek, Ep) \text{ are incomparable"}\ \text{"}c(Ep, Eq) \text{ and } c(Ek, Eq) \text{ are incomparable"}\}.$$

Let us suppose that $c(Ek, Ep) < c(Ep, Eq)$. In this case also $c(Ep, Eq, Ek) < c(Ep, Eq)$ so $Ek$ may not be member of the concept represented by $\{Ep, Eq\}$.

*Step 5: Introduce examples into the cluster.*

For each $Ek$ from $T$, if $c(Ep, Eq)$ and $c(M, Ek)$ are incomparable, then introduce $Ek$ into $M$.

Initially $M = \{Ep, Eq\}$. Therefore, $c(M, Ek)$ means $c(Ep, Eq, Ek)$.

If $M = \{Ep, Eq, \cdots, Et\}$ then $c(M, Ek)$ means $c(Ep, Eq, \cdots, Et, Ek)$.

At the end of this step one has discovered the cluster represented by the seed $\{Ep, Eq\}$:

$$M = \{Ep, Eq, \cdots, Es\}.$$

*Step 6: Replace the examples contained in the cluster with the cluster.*

Remove from the set of examples $E$, the elements of the discovered cluster $M = \{Ep, Eq, \cdots, Es\}$.

Consider $M$ as a *complex* example $Em$ and introduce it into $E$.

That is, $E \leftarrow (E - M) \cup \{Em\}$.

If $Ei$ is an initial example and $Em = \{Ep, Eq, \cdots, Es\}$, is a *complex* example, then we consider that

$$G(Ei, Em) = G(Ei, Ep, Eq, \cdots, Es)$$

and

$$c(Ei, Em) = c(Ei, Ep, Eq, \cdots, Es).$$

If $Ms$ is a cluster containing the *complex* example $Em$ then $Ms$ represents a super-concept of the concept $Em$.

*Step 7. Rerun the algorithm.*

Repeat from step 1 with the new set of examples until $E$ is reduced to one element $(Ek)$ or to two elements $(Ek1, Ek2)$. $Ek$ [respectively $G(Ek1, Ek2)$] is the concept representing all the examples.

The presented algorithm is only a basic one. Several improvements are obvious. For instance, let us consider again step 2. If $E = \{Ei1, \cdots, Eit\} \cup \{Em\}$, $Em$ being the last *complex* example formed, then we have to consider only the pairs $\{Eik, Em\}$ since the other pairs have already been considered in the previous step 1.

One could also modify step 3 of the above algorithm by working with several seeds simultaneously. Indeed, instead of choosing one seed (among the incomparable ones) and to determine the corresponding cluster, one could consider all the competing seeds at the same time and determine simultaneously the corresponding clusters.

Let us also notice that although the discovered clusters are disjoint with respect to the clustered examples (or one is included into the other) their descriptions are not guaranteed to be disjoint. That is, they may have common instances.

Since an example of using this algorithm is presented in Section VI, here we will only make clear its differences with CLUSTER/2 [12]. Both these algorithms are able to discover hierarchies of concepts and each concept is represented as a conjunction of predicates. Even more, they are both based on a notion of seed. However, in CLUSTER/2 a seed is an example, while in our algorithm a seed is represented by a pair of examples.

CLUSTER/2 requires as input the number of clusters to be determined. The analog in our algorithm is the "resolution." CLUSTER/2 may successively consider different numbers, until it finds the right one. Similarly, our algorithm may do experiences with different resolutions.

Another difference between these algorithms is that CLUSTER/2 is looking for nonoverlapping concepts that optimize pre-defined criteria, while our algorithm is looking for the most relevant concepts (as defined in Section III), be these overlapping or not.

Finally, we may contrast the presented algorithm with the other clustering algorithm developed in our team [1]. While both algorithms combines symbolical and numerical methods, the presented algorithm is more symbolically oriented while the other one relies more on the numerical approach. The two approaches complement each other in a natural way. While the clustering algorithm presented in this paper tends to discover more relevant concepts, it also requies more processing resources.

## V. Generalizing by Clustering

In this section we present an interesting relation which exists between generalization and clustering, in our approach.

Recall, from the previous section, that our clustering algorithm uses generalizations in order to cluster. We shall show that computing *good* generalizations of complex examples requires in turn a clustering phase.

By complex examples we mean examples containing objects. For instance, the examples in Fig. 1 contain car and load objects. The description of an object "O," from such an example, consists of all the predicates containing "O" as an argument. For instance, the description of the first car of the first train is the following one.

$C1$: (Car-shape machine $C1$) (length long $C1$) (infront $C1$ $C2$) (nr-wheels 2 $C1$).

The description of the load in the second car is:

*L2:* (Contains $C2$ $L2$) (load-shape square $L2$) (nrpts-load 3 $L2$).

One may easily notice that the description of a train is the conjunct of the descriptions of the objects it contains, except that, in the latter, some predicates are duplicated. For instance, (infront $C1$ $C2$) appears both in the description of $C1$ and in that of $C2$.

Provided that the two trains in Fig. 1 are examples of a general train concept, the objects from these examples are instances of the objects from the general train description. Therefore, one way to determine the general train concept is to match and generalize the descriptions of the objects from the examples.

Our claim is that for *computing good generalizations of complex examples one should match the most similar objects.*

We shall illustrate this generalization strategy by applying it to compute a generalization of the two trains in Fig. 1.

First of all we extract from each of the two train examples the descriptions of the objects. We find 9 objects (5 cars and 4 loads) in the first example and 7 objects (4 cars and 3 loads) in the second one.

Next we look for the two objects (one from $E1$ and the other from $E2$) for which the conceptual distance is minimum. These objects will be matched, that is, they are supposed to represent the same object in the generalization of $E1$ and $E2$. This matching will of course influence the following matchings. For instance, if two cars *Ci* and *Cj* are matched, and are represented in the generalization by $X1$, then the loads contained into these cars (*Li* and respectively *Lj*) become more similar to each other. This is represented by replacing, in the descriptions of *Li* and *Lj*, the predicates (contains *Ci Li*) and (contains *Cj Lj*) by (contains $X1$ *Li*) and (contains $X1$ *Lj*), respectively.

Having established the matching between the most similar objects, we look for the other two objects which are the most similar. We continue this way, matching each object from $E1$ with an object from $E2$ (using idempotency, if needed).

We find the generalization of $E1$ and $E2$ as the union of the generalizations of the corresponding objects (eliminating of course the identical predicates):

(car-shape machine $X1$) (length long $X1$) (infront $X1$ $X5$) (nr-wheels 2 $X1$)
(car-shape * $X2$) (length short $X2$) (contains $X2$ $Y1$) (infront $X2$ $X4$)
(infront * $X2$) (nr-wheels 2 $X2$) (load-shape triangle $Y1$) (nrpts-load 1 $Y1$)
(car-shape * $X3$) (length short $X3$) (contains $X3$ $L2$) (infront $X4$ $X3$)

(nr-wheels 2 $X3$) (load-shape circle $Y2$) (nrpts-load * $Y2$)
(car-shape open-top $X4$) (length * $X4$) (contains $X4$ $Y3$) (nr-wheels * $X4$)
(load-shape polygon $Y3$) (nrpts-load 1 $Y3$)
(car-shape open-top $X5$) (length * $X5$) (contains $X5$ $Y4$) (infront $X5$ *)
(nr-wheels 2 $X5$) (load-shape polygon $Y4$) (nrpts-load * $Y4$)

This description represents a type of train having the following features:

• The first car is a two wheel machine. It is followed by a two wheel open car containing polygons.

• The last car is a two wheel short one, containing circles. It is preceded by an open car containing one polygon which, in its turn, is proceeded by a two wheel short car containing a triangle.

When there are more than two examples to generalize, one needs to cluster the objects and to match objects belonging to the same cluster.

The main advantage of this approach is that it reduces a process of finding a generalization of complex descriptions to several processes of finding generalizations of much simpler descriptions.

The clustering of the objects may raise, however, complex combinatorial problems. In such a case, one may use heuristics (or a very simple clustering algorithm) to limit the objects to be clustered. For instance, it should be easy to establish that one must try to match cars with cars and loads with loads. Moreover, one does not need to cluster all the examples, but only to find out one cluster containing an object from each example.

## VI. Acquiring Object Knowledge

In this section we shall illustrate the relevance of our learning approach to the automation of knowledge base construction for expert systems.

A commonly used method of representing knowledge in artificial intelligence systems is to use prototypes [2], [8], [14], [16]. Each prototype represents a class of objects which is relevant for the system's application domain. The prototype is a parameterized representation of the properties common to the objects in the class. These prototypes are ordered in a class–subclass hierarchy in which each prototype inherits the properties of its superclass prototypes. The main feature of such a representation is that knowledge is organized around conceptual entities, in a memory efficient manner.

Therefore, automated construction of hierarchies of prototypes is an attempt towards automated construction of knowledge bases for expert systems.

The clustering algorithm presented in Section IV is able to discover a hierarchy of concepts characterizing a set of examples. The only thing which remains to be done, for building a hierarchy of prototypes, is to fill up this structure, by computing a description for each concept (node in the tree). Each such description has to be in terms of its ancestors in the tree, inheriting and particularizing their descriptions.

We shall illustrate this problem in the robot world [17] presented in Fig. 6. It consists of mechanical parts to be used in assembling tasks.

The robot is told the description of each part (AXLE1, AXLE2, WHEEL1, $\cdots$) and is asked to learn the general concepts represented by these examples. Such concepts are, for instance, axle, wheel, graspable-object, etc. The goal is to use the learned concepts in planning assembling tasks (for instance, planning the assembly of a car).

For instance, the following is the description of AXLE1:

(RELATION ATTACHED $R1$) (RELATION ATTACHED $R3$) (RELATION THRU $R2$)
(ACTION GRASP $R2$) (ACTION MOVE $R2$) (ACTION INSERT $R1$)
(ACTION INSERT $R3$) (POSITION $P3$) (GRASPING $G2$) (APPROACHING $L2$)
(SUBPART SOLID CYLINDER(5 1) $R1$) (SUBPART SOLID CYLINDER(20 4) $R2$)
(SUBPART SOLID CYLINDER(5 1) $R3$) (ALIGNED $R1$ $R2$ $R3$)

The symbols $R1$, $R2$, and $R3$ are the names of AXLE1's subparts. $P2$, $G2$, $L2$, are constants representing spatial positions. $R1$ and $R3$ could be in the relation ATTACHED with other entities (ATTACHED to a WHEEL, for instance) and $R2$ could be in the relation THRU with other entities (THRU a CARBODY HOLE, for instance). The actions which could be performed on AXLE1 are GRASP (by grasping $R2$), MOVE (by moving the grasped subpart), and INSERT (by inserting $R1$). AXLE1 is also characterized by a position ($P2$), a grasping point ($G2$) and a corresponding approaching point ($L2$). The three subparts $R1$, $R2$, $R3$, are solid cylinders, two of them ($R1$ and $R3$) having the same dimensions (height and diameter). The parts are aligned.

The robot is also given that the predicates describing the possible relations between parts or the actions that may be performed with these parts are to be considered more important than the predicates describing the shape of the parts. This information is given by the teacher in the form of the following weights associated with the predicates:

$$\omega(\text{RELATION}) = \omega(\text{ACTION}) = 4$$

$$\omega(\text{GRASPING}) = \omega(\text{APPROACHING}) = \omega(\text{POSITION}) = 3$$

$$\omega(\text{SUBPART}) = 2$$

$$\omega(\text{ALIGNED}) = \omega(\text{INSIDE}) = \omega(\text{PARALLEL}) = 1.$$

Running our clustering algorithm with these examples we obtained the hierarchy of concepts shown in Fig. 7.

Notice that the robot discovered the concepts that are relevant to its goal (planning assembling tasks).

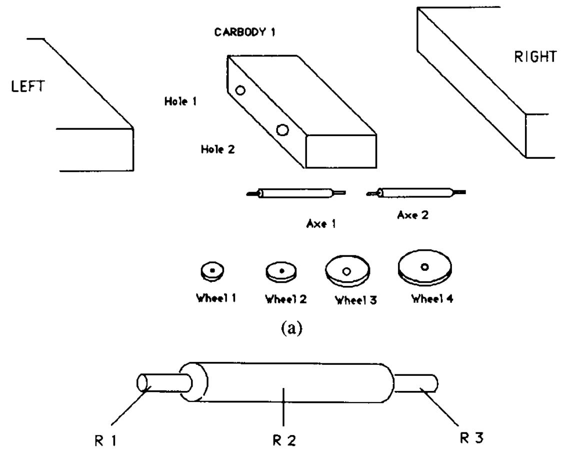


Fig. 6. (a) A robot assembly world. (b) Details of an axle.

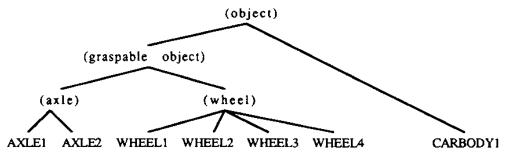


Fig. 7. Concepts learned from the robot world in Fig. 6.

Let us now suppose that the robot goal is to recognize objects. In this case, the teacher will have to state that the most important predicates are those describing the shape of the parts and the robot may discover the concepts shown in Fig. 8.

Let us consider again the hierarchy in Fig. 7. We want to transform it into a hierarchy of prototypes [18]. In such a hierarchy, each prototype is defined in terms of its ancestors. For instance, one says that *graspable object* is an object that has specific properties and that *axle* is a *graspable object* that has specific properties. The description that is actually associated with a prototype consists of its specific properties. Therefore, the description of *object* consists of the features common to all the objects from Fig. 6. Also, the description of *graspable object* consists of the features common to the axles and wheels, except those that are already present in the description of *object* because they are automatically inherited.

Our clustering algorithm has already computed a description of each concept. Therefore, to transform the hierarchy in Fig. 7 into a hierarchy of prototypes one has only to remove, from the description of each concept, the features which are already present into the descriptions of its ancestors. Acting this way one obtains the hierarchy of prototypes shown in Fig. 9.

The prototype *object* contains the properties common to all objects in the robot world presented. These properties express the fact that an object (be it a graspable one or not) could be in certain relations with other objects (there is no relation common to all objects and this is the

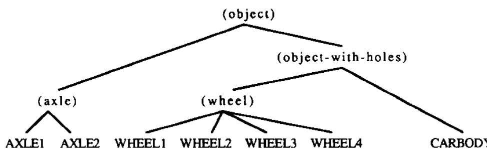


Fig. 8. Another set of concepts learnable from Fig. 6.

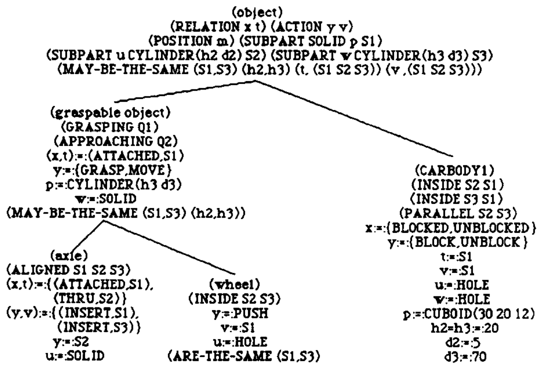


Fig. 9. A hierarchy of prototypes learned from examples.

reason for the presence of the variables in the description), that certain actions can be performed on the object, that the object is characterized by a certain spatial position and has cylinder subparts of the same height. Implicitly, different names means different entities. Therefore the MAY-BE-THE-SAME predicate indicates which variables can take the same value. For instance "$v$" could take the same value as $S1$ or $S2$ or $S3$.

The prototype *graspable object* contains the properties common only to graspable objects (has a GRASPING point and a corresponding APPROACHING point). It also inherits the properties of *object* and establishes values for some of the variables in the inherited properties.

$y :=: \{$GRASP, MOVE$\}$ means that the property (ACTION $y$ $v$), which is inherited from *object*, has to be instantiated to (ACTION GRAP $v$) & (ACTION MOVE $v$).

$(x, t) :=:$ (ATTACHED, $S1$) means that any inherited property containing the tuple $(x, t)$ has to be instantiated by replacing $x$ with ATTACHED and $t$ with $S1$.

Similarly, *axle* defines the properties common only to axles, but not common to all graspable objects or general objects. It also inherits the properties of its ancestors.

The significance and the advantages of the above description are those generally mentioned in connection with the hierarchies of prototypes: knowledge is organized around relevant conceptual entities (prototypes) in a memory efficient manner (the inheritance mechanism allows for a unique representation of a property common to some objects as the property of a prototype of those objects). Moreover, the generalization techniques are able to reveal subtle features that are common to the objects.

One disadvantage of the above descriptions is that they are somehow complicated. Therefore they need to be simplified by removing certain facts that may be proven to be useless for the application domain.

Once learned, these prototypes may be directly referred to by other pieces of knowledge [17]. For instance, a rule may indicate to the robot that, for moving a graspable object, it has to move the hand to the object's approaching point, open the hand, move the hand to the object, close the hand, and move the hand. This rule may be used for each graspable object instance.

## VII. Conclusions

One of the most critical problems of inductive learning is that of choosing among competing generalizations. In this paper we proposed and justified a solution to this problem which is based on the notion of conceptual distance and consists in enhancing the symbolical method of generalization with some numerical estimations.

A distinctive feature of our approach is that learning from examples and learning by observation are seen as complementary learning paradigms:

- learning by observation uses learning from examples to determine the examples to cluster;
- learning from examples uses learning by observation to determine the objects to match.

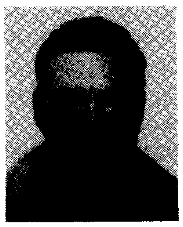
These relationships allowed the definition of a recursive learning method in which a complex learning from examples task is reduced to a task of clustering the objects contained in the examples, which in turn is reduced to a task of learning from these objects.

## Acknowledgment

This paper was written while one of the authors (G. Tecuci) was at LRI, on leave from his institute. His expenses have been taken in charge through an agreement between the French CNRS and the Romaninan Academy of Sciences. We wish to express our gratitude to both these institutions. We also wish to thank the referees for their detailed and helpful comments.

## References

[1] N. Benamou and Y. Kodratoff, "Conceptual hierarchical ascending classification," LRI, Univ. Paris-Sud, Res. Rep. 305, 1986.
[2] D. G. Bobrow and W. Winograd, "An overview of KRL, a knowledge representation language," *Cognitive Sci.*, vol. 1, pp. 3-46, 1977.
[3] P. R. Cohen and F. A. Feingenbaum, Eds., "Learning and inductive inference," in *The Handbook of Artificial Intelligence*, vol. 3. Los Angeles, CA: Kaufmann, 1982, pp. 323-511.
[4] G. T. Dietterich and R. S. Michalski, "Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods," *Artificial Intell. J.*, vol. 16, pp. 257-294, 1981.
[5] D. Fisher and P. Langley, "Approaches to conceptual clustering," in *Proc. Ninth Int. Joint Conf. Artificial Intell.* Los Angeles, CA: Kaufmann, 1985, pp. 688-697.
[6] Y. Kodratoff, J. G. Ganascia, B. Clavieras, T. Bollinger, and G. Tecuci, "Careful generalization for concept learning," in *Proc. European Conf. Artificial Intell.*, Pisa, Italy, 1984, pp. 483-492; also available in *Advances in Artificial Intelligence*, T. O'Shea, Ed. Amsterdam, The Netherlands: North-Holland, 1985, pp. 229-238.
[7] Y. Kodratoff, and J. G. Ganascia, "Improving the generalization step in learning," in *Machine Learning: An Artificial Intelligence Approach*, vol. 2, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Los Angeles, CA: Morgan-Kaufmann, 1986, pp. 215-244.
[8] Y. Kodratoff and G. Tecuci, "Techniques of design and DISCIPLE learning apprentice," *Int. J. Expert Syst.*, vol. 1, no. 1, pp. 39-66, 1987.
[9] P. Langley, Ed., *Proc. Fourth Int. Workshop Machine Learning*, Univ. California. Los Angeles, CA: Morgan-Kaufmann, 1987.
[10] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach*, vol. 1. Palo Alto, CA: Tioga, 1983.
[11] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach*, vol. 2. Palo Alto, CA: Morgan-Kaufmann, 1986.
[12] R. S. Michalski, and R. Stepp, "Learning by observation," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Palo Alto, CA: Tioga, 1983, pp. 163-190 (now distributed in the U.S. by Kaufmann and in Europe by Springer-Verlag).
[13] R. S. Michalski, "Inductive learning as rule-guided transformation of symbolic descriptions: A theory and implementation," in *Automatic Program Construction Techniques*, A. W. Biermann, G. Guiho, and Y. Kodratoff, Eds. New York: Macmillan, 1984, pp. 517-552.
[14] M. L. Minsky, "A framework for representing knowledge," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975, pp. 211-277.
[15] T. M. Mitchell, J. C. Carbonell, and R. S. Michalski, Eds., *Machine Learning: A Guide to Current Research*. Amsterdam, The Netherlands: Kluwer, 1985.
[16] N. Sridharan and J. Bresina, "A mechanism for the management of partial and indefinite descriptions," Rutgers Univ., Tech. Rep. CBM-TR-134, 1983.
[17] G. Tecuci, D. Mandutianu, and S. Voinea, "A hierarchical system for robot programming," *Comput. Artificial Intell.*, vol. 2, 1983.
[18] G. Tecuci, "Learning hierarchical descriptions from examples," *Comput. Artificial Intell.*, vol. 3, 1984.
[19] J. Van Ryzin, Ed., *Classification and Clustering*. New York: Academic, 1977.



**Yves Kodratoff** received the French state doctorate degree in physical chemistry in 1967.

He has been working since then at the Centre National de la Recherche Scientifique where he is presently Director of Research. He started working on artificial intelligence at the University of Paris-6 in 1973. He is presently leading the "Inference and Learning" research group of the Department of Computer Science of the University of Paris-Sud. About 30 researchers work in this group on themes related to automatic programming and machine learning. His interests also lie in the epistemology of AI. He organized most of the earliest machine learning meetings in Europe and is currently working on several European projects.

**Gheorghe Tecuci** received the M.S. degree in computer science from the Politechnical Institute of Bucharest, Romania, in 1979 and the Ph.D. degree in computer science from the University of Paris-Sud in 1988.

Since 1979 he has been with the Research Institute for Computers and Informatics, Bucharest. During the summers of 1986, 1987, and 1988, he was a Visiting Scientist at Laboratoire de Recherche en Informatique, University of Paris-Sud. His current research interests include machine learning, expert systems, and artificial intellience.