LEARNING TWO-TIERED DESCRIPTIONS OF FLEXIBLE CONCEPTS: THE POSEIDON SYSTEM

by

F. Bergadano
S. Matwin
R. S. Michalski
J. Zhang

Reports of the Machine Learning and Inference Laboratory,MLI 90-10, School of Information Technology and Engineering, George Mason University, Fairfax, VA, September 1990.

Submitted for publication in *Machine Learning Journal*, 1991

LEARNING TWO-TIERED DESCRIPTIONS OF FLEXIBLE CONCEPTS: The POSEIDON System

F. Bergadano[1], S. Matwin[2], R.S. Michalski and J. Zhang

Artificial Intelligence Center
George Mason University,
Fairfax, VA 22030
email: jzhang@gmuvax2.gmu.edu

ABSTRACT

Machine learning research has so far been primarily concerned with learning *crisp* concepts, that is concepts that are well-defined and context-independent. Most real-life concepts, however, are *flexible*, as their meaning is not precise and can change with the context in which they are used. To extend the applications of machine learning, it is therefore important to develop methods for learning flexible concepts. This paper describes such a method, which is based on a *two-tiered* concept representation. In the two-tiered representation, the first tier, called the *Base Concept Representation*, contains an explicit description of core concept properties, and the second tier, called the *Inferential Concept Interpretation*, defines allowable modifications of the explicit meaning and its dependence on the context of discourse.

The method generates a Base Concept Representation by first creating a complete and consistent concept description from supplied training examples, and then optimizing the description according to a general description quality criterion. The complete and consistent description is obtained by applying the AQ inductive learning methodology. The optimization process is done by a best-first search, that simplifies the initial concept description. The inferential concept interpretation consists of a user-specified procedure of *flexible matching* and a set of inference rules.

The method has been implemented in the POSEIDON[3] learning system, and experimentally tested on two different real-world problems: learning the concept of an acceptable union contract and learning the distinction between republicans and democrats in the U.S. Congress. For comparison, a few other learning methods were also applied to the same problems. These methods included simple variants of exemplar-based learning and the decision tree learning, based on the ASSISTANT program. The results have shown that the POSEIDON generated concept descriptions were easier to understand and slightly more accurate than those produced by the other methods.

Key words: learning from examples, learning imprecise concepts, flexible concepts, two-tiered concept representation, empirical learning, flexible matching, description quality measure, best-first search.

---

1 On leave from the University of Torino, Italy

2 On leave from the University of Ottawa, Canada

3 The system is named after POSEIDON, the Greek god of the waves, which represent fluidity and changing aspects of nature.

1. INTRODUCTION

Most current methods of machine learning, both empirical and analytic, assume that concepts are precise and context-independent entities, representable by a single symbolic description. A recognition of such concepts, which we call *crisp*, is therefore very simple: if an instance satisfies a concept description, then it belongs to the concept, otherwise it does not. Another common assumption is that concept instances are equally representative, that is, there is no distinction between typical and less typical instances.

In some methods, these assumptions are partially relaxed by assigning to a concept a set membership function (e.g., Zadeh, 1974) or a probability distribution (e.g., Cheeseman et al., 88). However, once such a measure is defined explicitly for a given concept, the concept again has a fixed, well-defined meaning. Moreover, these approaches remain unsatisfactory for coping with context-dependency, handling exceptional cases, or for capturing increases of knowledge about the concept properties.

When one looks at human concepts, however, one can see that most of them inherently lack precisely defined boundaries and have a context-dependent meaning. Our claim is that the imprecision of flexible concepts often has logical, rather than probabilistic character. This means that the classification of a non-typical instance of a flexible concept normally involves reasoning, rather than calculation of probabilities. In other words, to decide about the concept membership of an instance that is non-common, borderline, or irregular in any sense, one may reason deductively from one's background knowledge, draw an analogy or perform induction, instead of relying on a statistical analysis.

Examples of human concepts are usually not all equivalent. They may be characterized by a *degree of typicality* in representing the concept. For example, a robin is usually viewed as a more typical bird than a penguin or an ostrich. Moreover, in different contexts, the concept "bird" may apply to a live, flying bird, a sculpture, a chick hatching out of the egg, or even an airplane. Thus, human concepts are *flexible*, as they can modify or change their meaning in different situations or contexts. It is clear that in order to handle such flexibility of concepts, machine learning systems need to employ richer concept representations than are currently used. Developing methods for acquiring flexible concepts and reasoning with them emerges as an important goal for machine learning research.

Our approach to learning flexible concepts is based on the idea of *two-tiered (TT) representation*, proposed by Michalski (1987). In this representation, the meaning of a concept is defined by two components, the *base concept representation* (BCR) and the *inferential concept interpretation* (ICI). The BCR defines explicitly the properties of the concept, while the ICI describes allowed

modifications and extensions of the concept's properties in different situations.

In the general formulation of the TT representation, the "distribution" of the meaning between these two components is not fixed, but it depends on the properties of the reasoning agent and the criteria evaluating the quality of concept descriptions. For example, in the method described here, the first phase of a learning process produces a concept description in which BCR is a complete and consistent (CC) characterization of all training examples. Such a description, as shown experimentally, may be overly complex and perform poorly on new examples. The second phase optimizes this description according to a criterion measuring its "quality." Our experiments have shown that so optimized description, in addition to being substantially simpler, may also perform better in recognizing new concept examples than the original CC description. In the application of the TT method to modelling human concept representation, the BCR would describe the most typical, common, and intentional meaning of a concept, while the ICI handles the exceptional or borderline cases, and context dependency (Michalski, 1990).

Early ideas, experiments and the first method for learning two-tiered concept representations were presented in (Michalski et al,. 1986 and Michalski, 1988; see also Michalski, 1990). The method proposed there employed a simple form of description simplification, called TRUNC, which used only specialization as a description reduction operator. An intriguing result of that research was that a substantial reduction of the description's complexity was achieved without affecting its performance on new examples. The effect was obtained by removing the parts of the CC description that covered only a small fraction of examples (the so called *light complexes),* and by applying a *flexible matching* procedure for concept recognition.

This paper is an extension and continuation of these early ideas. One important advance is the development of a heuristic double-level search procedure, called TRUNC-SG, which explores the space of two-tiered descriptions in order to determine a globally optimized description. The search employs both generalization and specialization operators, and is guided by a new criterion. the *general description quality* measure *(GDQ).* This measure takes into consideration the description accuracy, the computational cost of both tiers - BCR and ICI, and its cognitive comprehensibility (Bergadano et al., 1988). By introducing such a general description quality measure one can view concept learning in a new way, namely, as a process of modifying the input concept description in order to maximize the description quality measure. The input description can be in the form of positive and negative examples, a CC concept description, a teacher-supplied tentative description, an abstract description (as in the explanation-based learning), or other.

Another difference between the present approach and previous research is that flexible matching is used during not only in the recognition process, like in (Michalski et al., 1986), but also in the learning process, i.e., in searching for high "quality" concept descriptions. This feature distinguishes the method also from the related method described in (Bergadano and Giordana,

1989), which does not involve deductive reasoning in the learning phase, and evaluates the performance of generated descriptions solely on the basis of the coverage of examples. These earlier approaches may be compared to using hands in learning how to row a boat, and then using oars in the performance phase. The idea is that learning is more effective if one uses the same instruments for learning and for the performance phases. Consequently, the presented approach represents an important advance over tree-pruning techniques (e.g., Quinlan, 1987), which do not use any form of deductive matching or flexible extension of the learned descriptions. Other advances include the ability of taking into consideration the typicality of training instances (when it is known), and the introduction of a rule base in the ICI. The paper presents also a body of experimental results comparing the performance of the proposed method with several other methods, such as variants of exemplar-based learning, decision tree learning, learning CC descriptions, and the earlier method based on using TRUNC procedure.

The method presented has been implemented in the POSEIDON system, and experimentally applied to two different real-world problems: learning the concept of an acceptable union contract, and learning voting patterns of republicans and democrats in the U.S. Congress. The experiments have confirmed the initial findings that the TT representation can lead to a substantial reduction of concept representation, and at the same time to an improvement of its predictive power. They also showed that in the applications considered, the method proposed produced simpler and more accurate concept descriptions than other learning methods, such as simple variants of exemplar-based learning and decision tree learning with pruning.

2. TWO-TIERED CONCEPT REPRESENTATION

2.1 Discussion and Definition

Traditional work on concept representation has assumed that the whole meaning of a concept resides in a single stored structure, e.g., a semantic network, a logic-based description or a decision tree. Such a structure is expected to capture all relevant properties of the concept(s) and define the concept borderline (e.g., Collins and Quillian, 1972; Minsky, 1975; Smith and Medin, 1981; Sowa, 1984). In domains with a significant amount of noise, the knowledge structure may be partially inconsistent and/or incomplete with regard to the known instances or facts about the concept. The latter is illustrated by the work on pruning decision trees (e.g., Quinlan, 1987), and the HILLARY system (Iba et al., 1988). The work on two-tiered representation (Michalski, 1987; and this paper) also points in this direction.

In traditional approaches, recognizing a concept instance typically involves a direct matching of the properties of the instance with the stored concept representation. Such matching may include comparing feature values in an instance with those in the concept description, or tracing links in networks of concepts, but has not been assumed to involve any complex inferential processes. More recently, researchers working on exemplar-based reasoning (e.g., Bareiss, 1989; Kolodner,

1988 and Hammond, 1989) have recognized the need for using advanced inference mechanisms for classifying new instances. In these methods, however, the concept representation consists of stored individual examples. Such a representation is memory-taxing, and makes it more complex to match concepts with instances. The same objection continues to apply to those approaches to exemplar-based learning that generalize the cases that are stored, because the number of cases may still be very large.

In contrast to the above, the two-tiered (TT) representation involves a combination of an explicit concept description (BCR), and an implicit one (ICI), represented by an inference mechanism for matching the description with instances. The BCR consistes of a description of the concept in terms of attributes observed in examples and/or higher level concepts employed during concept formation. The BCR can be viewed as a characterization of the central tendency of the concept. It captures the principle, the most relevant properties and basic the intention behind the concept.

The ICI handles special cases, anomalies, exceptions and context-dependencies[4]. It treats them either by extending the base concept representation (concept *extension)*, or by specializing it (concept *contraction*). This process involves the background knowledge and relevant inference methods contained in the ICI. Inference allows the recognition, extension or modification of the concept meaning according to a context.

When an unknown entity is to be recognized, it is first matched against the BCR. Then, depending on the outcome, the entity may be related to the concept's inferential extensions or contractions. A simple inferential matching can involve just a probabilistic inference based on some similarity measure, e.g., the method of *flexible matching* (Michalski et al., 1986). An advanced matching may involve any kind of inference - deductive, analogical or inductive.

Let us illustrate the idea of TT representation using the concept of "chair". A TT representation of that concept could be:

BCR: A piece of furniture.
Function: to seat one person.
Structure: a seat supported by four legs and a backrest attached from the side.
(Various physical properties of typical chairs are specified, such as a characterization of their shape, the range of heights, widths, used materials, etc.).

---

4 The terms such as anomalous, exceptional, and representative examples, as well as context, are used here in their colloquial meaning. They will be more precisely defined in sec. 2.4.

ICI: No-of legs can vary from one to four. The material of which the chair is made, the shape of the seat, the backrest and the legs are irrelevant as long as the function is preserved. Backrest may be missing. The legs may be replaced by any support.
If legs are replaced by wheels --> type(chair) is wheelchair
Chair without the backrest --> type(chair) = stool
Chair with the armrests --> type(chair) = armchair
Context = museum exhibit --> chair is not used for seating persons any more, but has all the physical characteristics of a chair.
Context = toys --> the size can be much smaller than typical. The chair does not serve for seating persons, but correspondingly small dolls.

This simple example illustrates several important features of the TT representation method. The commonly occurring chairs will strictly match the BCR, and the ICI may not even be involved. This reduces the recognition time for these common cases. For objects that do not satisfy the BCR of any relevant concept, or satisfy the BCR of more than one concept, the ICI rules are involved (which concepts are relevant is decided by the context of discourse). The ICI can be changed, upgraded or extended, without any change to BCR. The concepts used in BCR and ICI may themselves be flexible, and thus also characterized by a TT representation.

The ideas of TT representation are also supported by research on the so-called transformational model (Smith and Medin, 1981). In this model, matching object features with concept descriptions may involve transformations of object features into those specified in the concept description. Such a matching has an inferential character. Some recent work in cognitive linguistics also seems to support the ideas of TT representation. For example, Lakoff (1987), in his idealized cognitive models (ICM) approach, stipulates that humans represent concepts as a structure, which includes a fixed part and mappings that modify the fixed part. The fixed part is a propositional structure, defined relative to some idealized model. The mappings are metaphoric or metonymic transformations of concepts meaning.

As mentioned before, in the general TT approach, the distribution of the concept meaning between the BCR and ICI can vary, depending on the criterion of the concept description quality. A special case of the BCR can be just concept examples, and of the ICI - procedures for a certain kind of inferential matching. Consequently, the TT representation can be viewed conceptually as a generalization of the representation used in simple case-based reasoning systems.

2.2 Base Concept Representation

In POSEIDON, we use the attribute-based *variable-valued logic system* VL_1 (Michalski 1983) as a representational formalism. The BCR for a concept is a disjunctive normal form expression in VL_1. Such an expression corresponds to a cover, which is a set of complexes. A complex is a conjunction of selectors and can be viewed as a rule (Michalski, 1990). A selector is a relational expression (a condition):

$$[L \# R]$$

where the attribute L is called the *referee*. R, called the *referent*, is a set of values from the domain of L. Symbol # denotes one of the relational symbols $=$, $<$, $>$, $\leq$, $\geq$, $\neq$. A selector is satisfied by an event if the value of attribute L in this event is in relation # to R.
For example, the expression [shape = circle v square] & [length > 2] is a complex containing two selectors. The first selector is satisfied by an event, if the attribute "shape" in this event takes value "circle" or "square".

2.3 Inferential Concept Interpretation: Flexible Matching Function

A flexible matching function F is used as a part of the ICI and it is predefined. The function measures the degree of match between an event and a concept description. The specific F used in our implementation maps events from the set E, and concept descriptions from the set D, into the degree of match from the interval [0..1]:

$$F: \; E \times D \rightarrow [0..1]$$

The value of F of an event e and a cover, *Cov*, is defined as the probabilistic sum of F for its complexes. If *Cov* consists of two complexes, cpx_1 and cpx_2, we have:

$$F(e,Cov) = F(e,cpx_1) + F(e,cpx_2) - F(e,cpx_1) * F(e,cpx_2)$$

A weakness of the probabilistic sum is that it is biased toward covers that consist of many complexes. Specifically, if a cover *Cov* includes many complexes, $F(e,Cov)$ may be close to 1, even if each $F(e,cpx)$ is relatively small (see Table 3 in sec. 6). To avoid this effect, if the value of $F(e,cpx_i)$ is below a given threshold, then it is assumed to be 0. In POSEIDON, this problem does not occur because covers are typically reduced to only a few complexes (see the TRUNC-SG procedure in sec. 3).

The degree of match, $F(e,cpx)$, between an event *e* and a complex *cpx* is defined as the average of the degrees of fit for its constituent selectors, weighted by the proportion of positive examples covered by the complex:

$$F(e,cpx) = (\sum F(e,sel_i)/n) * \#cpxpos/(\#cpxpos + \#cpxneg)$$

where *n* is the number of the selectors in *cpx*, and *#cpxpos* and *#cpxneg* are the number of positive examples and the number of negative examples covered by *cpx* , respectively.

The degree of match between an event *e* and a selector *sel* is weighted by the coverage of positive and negative examples by the selector. Such a degree of match depends on the type of the attribute in the selector. An attribute can be one of four types: nominal, structured nominal, linear and structured-linear (Michalski and Stepp, 1983). Values of a structured nominal (linear) attribute are nodes of an unordered (ordered) generalization hierarchy. In an ordered hierarchy, children nodes stemming from any parent constitute a totally ordered set.

In a nominal or structured nominal selector, the referent is a single value or an *internal disjunction* of values, e.g., [color = red v blue v green]. The degree of match is 1 if such a selector is satisfied by an event, and 0 otherwise. In a linear or structured linear selector, the referent is a range of values, or an internal disjunction of ranges, e.g., [weight = 1..3 v 6..9]. A satisfied selector returns the value of match 1. However, if the selector is not satisfied, the degree of match is the ratio of the distance between the value of the attribute in the event and the nearest value in the referent (or, in the case of a structured linear attribute, in the set of referent's descendants), and the size of the referent. For example, suppose the event is (weight = 2, height = 5, ...), and the selector is [height = 6...8]. The degree of match is (6-5)/2 = .5. Suppose now, that the selector is [height = high], and the value "high" is a node (in an ordered generalization hierarchy) whose descendents are values 6..10. The degree of match between the same event and the above selector would also be .5.

The system is not forced to make a decision when the difference between the values of flexible matching function for two concepts is very small. If the difference is smaller than a preset threshold, the result will be "no match".

2.4 Inferential Concept Interpretation: Deductive Rules

In addition to flexible matching, the ICI includes a set of deductive rules, allowing the system to recognize exceptions and context-dependent cases. For example, flexible matching could allow us to recognize an old sequoia as a tree, although it does not match the typical size requirements. Deductive reasoning is required to recognize a tree without leaves (in the winter time), or to include in the concept of tree its special instance (e.g. a fallen tree). In fact, flexible matching is most useful to cover instances that are close to the typical case, while deductive matching is appropriate to deal with concept transformations necessary to include exceptions or context-dependencies.

The deductive inference rules in the ICI are expressed as Horn clauses. The inference process is implemented using the LOGLISP system (Robinson and Sibert, 1982). Numerical quantifiers and

internal connectives are also allowed. They are represented in the annotated predicate calculus (Michalski 1983).

2.5 Types of Matching

There can be three different types of match between an event and a two-tiered description:

1. *Strict matching*: the event matches the BCR exactly, in which case we say that the event is S-covered.
2. *Flexible matching*: the event is not S-covered, but matches the BCR through a flexible matching function. In this case we say that the event is F-covered.
3. *Deductive matching*: the event is not F-covered, but it matches the concept through deductive reasoning based on the ICI rules. The event is referred to as D-covered.

The sets of all S-covered, F-covered, and D-covered events are called Scov, Fcov, and Dcov, respectively.

We can now give a precise meaning to the term *exception* used throughout this paper. Events are called *exceptions*, if they are D-covered. Events that are S-covered are said to be *representative* examples, and events that are F-covered are said to be *nearly-representative* examples. One major advance of the presented method over the previous method using TT representation (e.g., Michalski et al, 1986) is that the ICI includes not only a flexible matching procedure, but also inference rules. Thus, the method is oriented toward handling not only representative or nearly representative examples, but also exceptions.

3. AN OVERVIEW OF POSEIDON

3.1. Basic algorithm

Based on the ideas presented above, we have implemented a system for learning TT descriptions, called POSEIDON (or AQTT-16). Table 1 specifies the input, output, and the function of the system. The BCR is determined in two phases. The first phase generates a general consistent and complete (CC) concept description, and the second phase optimizes the description according to a criterion of concept quality.

The CC description is determined using the AQ15 inductive learning program (Michalski et al. 86). The initial BCR consists of the CC description together with the flexible matching procedure described above.

Phase 1

Given:

Concept examples obtained from a source
Relevant background knowledge

Determine:

Complete and consistent (CC) description of the concept

Phase 2

Given:

CC description of the concept
A general description quality (GDQ) measure
Typicality of examples (if available)

Determine:

A two-tiered description that maximizes the GDQ

Table 1. Specification of the POSEIDON system

The second phase improves the initial description by conducting a "double level" best first search. This search is implemented in the TRUNC-SG procedure (SG symbolizes that the method uses both specialization and generalization operators). The first level search is guided by the *description quality measure* (GDQ), which ranks descriptions for consideration (see sec. 4). The second level search is guided by heuristics that specify the search operators to be applied to a given description. The search operators simplify the BCR of a description by removing some of its components, or by modifying the arguments or referents of some of the predicates. A general structure of the system is presented in Fig. 1.

The search process is defined by:

Search space:	A tree structure, in which nodes are two-tiered concept descriptions (BCR + ICI)
Operators:	Selector removal, Complex removal, Referent modification.
Goal:	To determine a description that maximizes the general description quality criterion (GDQ).

The goal of the search procedure is not necessarily to find an optimal solution, as this would require a combinatorial search. Rather, the system tries to maximally improve the given concept description by expanding only a limited number of nodes in the search tree, which are suggested by certain heuristics to be discussed in section 5.1.

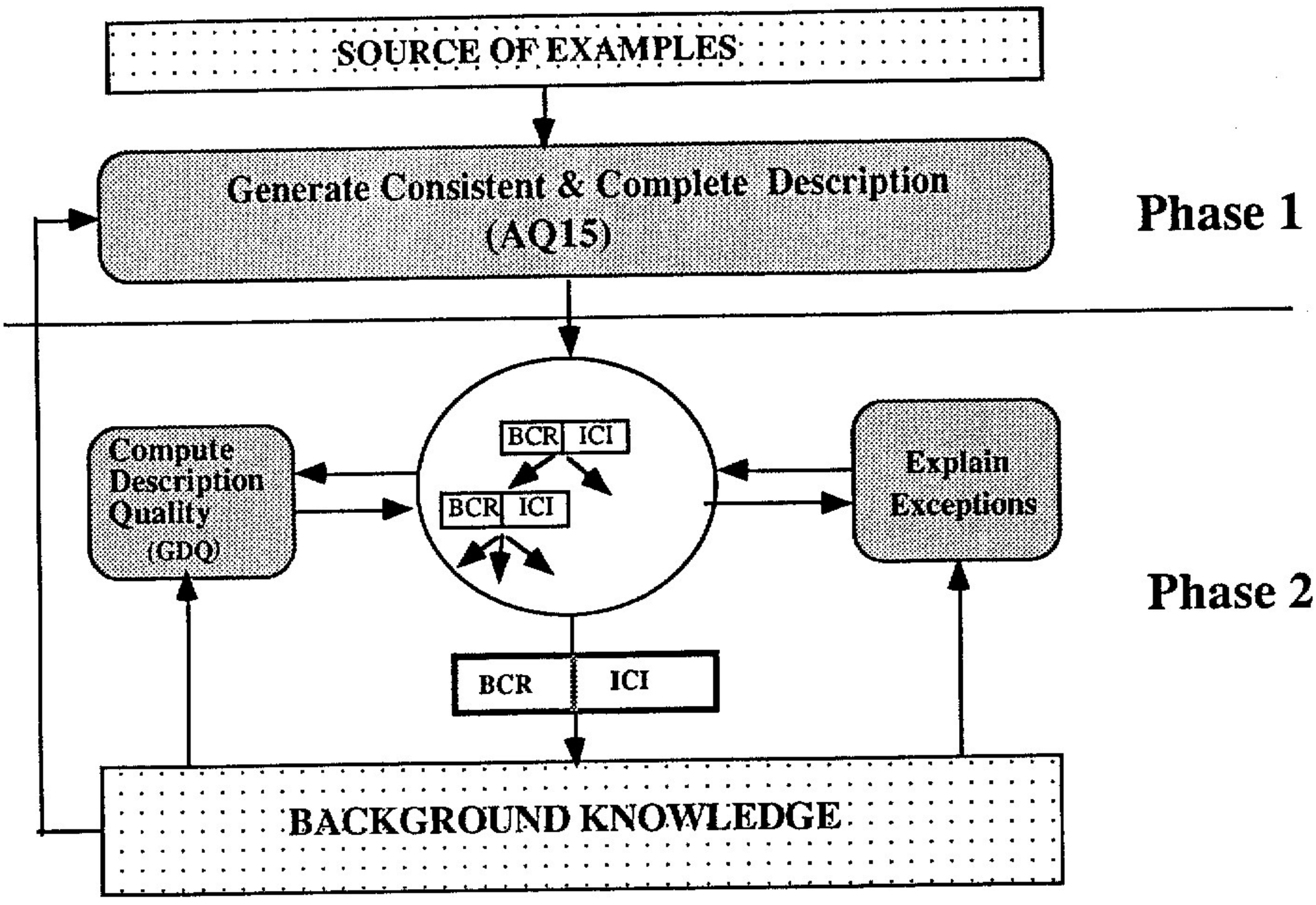

Figure 1. Learning in the POSEIDON system

The BCR is learned from examples and represented as a set of rules (a VL_1 cover, as described above). The ICI consists of two parts: a flexible matching function and a rule base. The rule base is used for explaining exceptional examples, and is acquired through an interaction with an expert.

3.2 Operators Used in Optimizing Base Concept Representation

A description can be modified in three basic ways: by complex (rule) removal, selector (condition) removal or referent (subset of the attribute range) modification. Rule removal is a specialization operator, because it leads to "uncovering" of some examples. It is the reverse of the "adding an alternative" generalization rule (Michalski, 1983). Condition removal is a generalization operator, as it is an instance of the "dropping condition" generalization rule. A referent modification can either generalize or specialize a description. Consequently, two operators are defined: *referent extension*, which generalizes a description, and *referent contraction*, which specializes a description. For illustration, consider the selector

[size = 1..5 v 7]

A referent modification that produces a generalization

[size = 1..7]

represents a referent extension operator. If in the above selector the relation is changed to <>

[size <> 1..5 v 7]

then the same referent modification, i.e., change to [size <> 1..7], represents a referent contraction operator. A referent modification that produces a specialization

[size = 1..5]

represents also a referent contraction operator. Table II lists search operators and specifies their effect on the description.

Search operator	**Type of knowledge modification**
Selector removal (SR)	Generalization
Complex removal (CR)	Specialization
Referent extension (RE)	Generalization
Referent contraction (RC)	Specialization

Table II. Search Operators

Thus, each search operator produces either a specialization or generalization of a description. A generalized (specialized) description covers potentially a larger (smaller) number of training examples (positive or negative). By measuring the changes in the coverage caused by applying an operator one can choose the operator at a given search step (section 5.2).

3.3 Learning the Inferential Concept Interpretation.

As indicated above, by applying a search operator (SR, CR, RE or RC) to the current BCR, one can make it either more general or more specific. If the modified BCR is more specific, some positive examples previously covered may cease to be S-covered. These examples may, however, be covered by existing ICI rules (so they become D-covered), or may be made D-covered by adding new rules to the ICI. On the other hand, if the modified BCR is more general than the original one, some negative examples, previously uncovered, may now be S-covered. As before, they may already be excluded by existing ICI rules, or can be excluded by adding new rules. Consequently, there are two types of rules in the ICI: those that cover positive examples left uncovered by the BCR ("positive exceptions"), and rules that eliminate negative examples covered by the BCR ("negative exceptions"). A problem then is how to acquire these rules.

The rules can be supplied by the expert, inherited from higher level concepts, or learned from other knowledge. If the rules are supplied by an expert, their form may not be operationally effective. In such a case, they can be made effective through some form of analytic learning (e.g., Mitchell et al., 86; Prieditis and Mostow, 1987). If the rules supplied by an expert are too specific or not

totally correct, they may be improved by induction (e.g., Dietterich and Flann 1988; Mooney and Ourston, 1989). Thus, in general, learning the ICI can be accomplished by different strategies, like learning the BCR.

In the implemented method, the system identifies exceptions (i.e., examples not covered by the BCR), and asks an expert for an explanation (see sec. 5.2). The expert is supposed to justify the identified exceptions, and to express this justification in the form of rules. The search procedure, shown in Fig. 1, guides the process by determining the examples that need explanation. This way, the role of the program is to learn the "core" part of the concept from the supplied examples, and to identify the exceptional examples.The role of a teacher is to provide concept examples, and to explain why the examples identified by the learner as exceptional also belong to the concept.

4. QUALITY OF CONCEPT DESCRIPTIONS

4.1 Factors Influencing the Description Quality

The learning methodology utilizes a general description quality measure (GDQ) that guides the search for an improved two-tiered description. The GDQ measure is influenced by three basic characteristics: the *accuracy*, the *comprehensibility*, and the *cost*. This section discusses these three components, and describes a method for combining them into a single measure.

The accuracy represents the description's ability to produce correct classifications. The numbers of positive and negative examples covered by a description determine its degree of completeness and consistency, and are indicative of its predictive power. In order to achieve a high degree of completeness and consistency when learning from noisy input examples, the system may produce overly complex and detailed descriptions. Such descriptions may be strongly dependent on the particular training set, and, consequently, may perform poorly in classifying future examples. For that reason it may be better, when learning from imperfect inputs, to produce descriptions that are only partially complete and/or consistent.

The comprehensibility of the acquired knowledge depends on subjective and domain-dependent criteria. If an AI system is supposed to supply advice to humans, knowledge used by such a system should be comprehensible to human experts. A "black box" classifier is not satisfactory in such situations. Therefore, knowledge acquired by a learning system should be related to terms, relations and concepts used by experts, and should not be syntactically too complex. This requirement is called the *comprehensibility* principle (Michalski, 1983). There is no well established measure of description comprehensibility. Therefore, we approximate it by a representational simplicity of a description. Such a simplicity is evaluated by counting the number of operators involved, and taking into consideration an estimated cognitive complexity of the operators in the description. In the case of two-tiered representations, the proposed approximation of the comprehensibility takes into account the operators occurring in both, the BCR and the ICI,

and weighs the relative contribution of each part to the comprehensibility of the whole description.

The third criterion, the description cost, captures the properties of the description related to its storage and its use (computational complexity). Other things being equal, descriptions which are easier to store and easier to use for recognizing new examples are preferred. When evaluating the description cost, two characteristics are of primary importance. The first is the cost of measuring values of variables occurring in the description. In some application domains, e.g., in medicine, this is a very important factor. The second characteristic is the computational cost (time and space) of evaluating the description. Again, in some real-time applications, e.g., in speech or image recognition, there may be stringent constraints on the evaluation time. The cost and the comprehensibility of a description are frequently mutually dependent, but generally these are different criteria.

The criteria described above need to be combined into a single evaluation measure that can be used to compare different concept descriptions. One solution is to have an algebraic formula that, given numeric evaluations for individual criteria, produces a number that represents their combined value. Such a formula may involve, e.g., a multiplication, weighted sum, maximum/minimum, or t-norm/t-conorm of the component criteria (e.g., Weber, 1983). Although these approaches are often appropriate, they also have significant disadvantages. First, they usually combine a set of heterogeneous evaluations into a single number, and the meaning of this final number is hard to understand for a human expert. Second, they may force the system to evaluate all the criteria for each description, even if it is sufficient to compare descriptions on the basis of just one or two most important ones. The latter situation occurs when one description is so much better than the other according to the most important criterion, that it is not worth to even consider the alternatives. To overcome these problems, we use a combination of an lexicographic evaluation and a linear function-based evaluation.

4.2 Combining Individual Factors Into a Single Preference Criterion

Given a set of candidate descriptions, we use the *general description quality* (GDQ) criterion to select the "best" description. In POSEIDON, the GDQ consists of two measures, the *lexicographic evaluation functional* (LEF), and the *weighed evaluation functional* (WEF). The LEF, which is computationally less expensive than WEF, is used to rapidly focus on a subset of the most promising descriptions. The WEF is used to select the final description. A general form of a LEF (Michalski, 83) is:

LEF: <(Criterion1,τ_1), (Criterion2,τ_2), ...(Criterionk,τ_k)>

where Criterion_1, Criterion_2, ... Criterion_k are elementary criteria used to evaluate a description, and τ_1, τ_2, ... τ_k are corresponding *tolerances* , expressed in %. The criteria are applied to a description in order from the left to the right (this order reflects their decreasing importance). At each step, all candidate descriptions whose score on a given criterion is within the tolerance range from the best scoring description on this criterion are considered equivalent with respect to this

criterion, and are kept on the CANDIDATE LIST; other descriptions are rejected. If after applying some criterion, there is only one description on the CANDIDATE LIST, this description is selected as the best. If after applying all criteria, the CANDIDATE LIST has more than one description, a standard solution is to chose the descritpion that scored highest on the first criterion. In POSEIDON, we chose another approach to the latter problem (see below).

The LEF evaluation scheme is not affected by the problems of algebraic evaluation functions mentioned above. The importance of a criterion depends on the order in which it is evaluated in the LEF evaluation scheme, and on its tolerance. Each application of an elementary criterion reduces the CANDIDATE LIST, and thus the subsequent criterion needs to be applied only to a reduced set. This makes the evaluation process quite efficient. In POSEIDON, the default LEF consists of the three elementary criteria discussed above, i.e., accuracy, comprehensibility and cost., specified in that order. The tolerances are parameters of the program, and set by the user.

While it is usually easy to determine the desired order of the criteria, it may be more difficult to decide the tolerance for them. If the tolerance for some criterion is too small, the chances of using the remaining criteria decrease. If the tolerance is too large, the importance of the criterion is decreased. For this reason, the LEF measure in POSEIDON is applied with relatively large tolerances, so that all the elementary criteria are taken into account. If after applying the last criterion the CANDIDATE LIST has still several candidates, a weighed evaluation functional (WEF) is used to make the final choice. The WEF is a standard linear function of the elementary criteria. The description with the highest WEF is selected. Thus, this approach allows one to use computationally very efficient LEF to reduce the set of candidates to a small set, and then apply a more complex measure to the remaining candidates.

4.3 The Role of Typicality of Examples

The accuracy is a major criterion for determining the quality of a concept description. Current machine learning methods usually assume that the accuracy depends only on the number of positive and negative examples covered by the description (training and/or testing). One can argue, however, that in evaluating accuracy one should take into consideration also the *typicality* of examples (Rosch and Mervis, 1975). If two descriptions cover the same number of positive and negative examples, the one that covers more typical positive examples and less typical negative examples can be considered more accurate.

For the above reason, we propose a measures of the degree of completeness and the degree of consistency of a description that takes into consideration the typicality of the examples. The typicality of examples can be approximated by the frequency of their occurence. Alternatively, the teacher supplying the examples can be asked to assign the typicality to the examples. If the typicality is not provided, the system makes the standard assumption that the typicality is the same

for all examples. The degree of completeness of a description is proportional to the typicality of the positive events covered. The consistency of the description is inversely proportional to the typicality of the negative events covered[5] .

In defining the completeness and consistency of a TT description, other factors may be taken into consideration. One may postulate that a description should cover the typical examples explicitly, and non-typical ones implicitly. Thus, the typical examples are covered by the BCR, and non-typical, or exceptional ones by the ICI. In POSEIDON, the BCR is inductively learned from examples provided by a teacher, and it is desirable that they are typical of the concept being learned. The ICI rules are provided by a teacher or deductively inferred from the background knowledge. They are assumed to handle special or rare cases. An advantage of such an approach is that the system learns a description of typical examples by itself, and the teacher needs to explain only the special cases.

In view of the above, the explicitly-covered (strictly-covered or S-COV) examples are assumed to contribute to the completeness of a description more than implicitly-covered, i.e., flexibly-covered (F-COV) or deductive inference rules-covered (D-COV). These assumptions are reflected by weights w_s, w_f, and w_d used in the definition of completeness and consistency described in the next section.

4.4 General Description Quality Measure

This section specifies the details of the GDQ measure implemented in POSEIDON. To do so, we first define the *typicality-based completeness*, T_COMPLETENESS, and the *typicality-based consistency*, T_CONSISTENCY of a two-tiered concept description:

$$\text{T\_COMPLETENESS} = \frac{\sum_{e^+ \in S\text{-cov}} w_s * \text{Typicality}(e^+) + \sum_{e^+ \in F\text{-cov}} w_f * \text{Typicality}(e^+) + \sum_{e^+ \in D\text{-cov}} w_d * \text{Typicality}(e^+)}{\sum_{e \in POS} \text{Typicality}(e)}$$

$$\text{T\_CONSISTENCY} = 1 - \frac{\sum_{e^- \in S\text{-cov}} w_s * \text{Typicality}(e^-) + \sum_{e^- \in F\text{-cov}} w_f * \text{Typicality}(e^-) + \sum_{e^- \in D\text{-cov}} w_d * \text{Typicality}(e^-)}{}$$

5 When negative examples are instances of another concept, as is often the case, their typicality is understood as the typicality of being members of that other concept.

$$\sum_{e \in NEG} \text{Typicality}(e)$$

where POS and NEG are sets of positive and negative examples, respectively, that are covered by the two-tiered concept description; and Typicality(e) expresses the degree of typicality of e as a representant of a given concept. Weights w_S, w_F, and w_D represent different significance of the type of coverage. They depend on certain thresholds reflecting the appropriateness of the type of coverage for the given degree of typicality:

w_S: if Typicality(e) $\geq t_2$ then 1, else w,

w_F: if $t_2 \geq$ Typicality(e) $\geq t_1$ then 1, else w,

w_D: if $t_1 \geq$ Typicality(e) then 1, else w,

where thresholds t1 and t2 satisfy the relation $0 \leq t1 \leq t2 \leq 1$, and $0 < w \leq 1$. The role of w is to weigh the examples that are covered in a manner (S, F or D) that is not compatible with their typicality.

The *accuracy* of a description is defined in terms of T_COMPLETENESS and T_CONSISTENCY:

ACCURACY = w_1*T_COMPLETENESS +w_2*T_CONSISTENCY

where $w_1 + w_2 = 1$. The weights w_1 and w_2 reflect the expert's judgement about the relative importance of completeness and consistency for the given problem. The default value of both is 0.5.

A measure of comprehensibility of a concept description is difficult to define. We will approximate it by measuring the representational complexity of description, defined as:

$$v_1 * \sum_{op \in BCR(dsp)} C(op) \quad + \quad v_2 * \sum_{op \in ICI(dsp)} C(op)$$

where BCR(dsp) is the set of all operator occurrences in the BCR, and ICI(dsp) is the set of all operator occurrences in the ICI. C(op), the complexity of an operator, is a real function that maps each operator symbol into a real number representing its complexity The valuescomplexities of the operators are ordered as follows:

C(interval) < C(internal disjunction) < C(=) < C(<>) < C(&) < C(v) < C(implication).

When the operator is a predicate, C increases with the number of the arguments of the predicate. Parameters v_1 and v_2 are weights such that $v_1+v_2 = 1$.

The BCR is supposed to describe the general and easy-to-define meaning of the concept, while the ICI is mainly used to handle rare or exceptional events. As a consequence, the BCR should be easier to comprehend than the ICI, and thus v_1 should be larger than v_2.

The cost of a description depends on two parts:

Measuring-Cost (MC) -- the cost of measuring variables used in the concept description.

Evaluation-Cost (EC) -- the computational cost of evaluating the concept description.

$$MC(description) = \sum_{e \in Pos \cup Neg} \sum_{v \in vars(e)} mc(v)/(|Pos| + |Neg|)$$

$$EC(description) = \sum_{e \in Pos \cup Neg} ec(e)/(|Pos| + |Neg|)$$

where *vars(e)* is the set of all occurrences of variables used to evaluate a concept description to classify the event *e*, *mc(v)* is the cost of measuring the values of the variable *v*, and *ec(e)* is the computational cost of evaluating the concept description to classify the event e. The latter depends on the computing time and/ or on the number of operators involved in the evaluation.

We now define the cost of a description:

$$Cost(description) = u_1 * MC(description) + u_2 * EC(description)$$

where u_1 and u_2 are weights defining the relative importance of the measuring-cost and the evaluation-cost.

The definitions of the above measures together with the specification of the way how they can be combined (sec. 4.2) completely define the general description quality (GDQ). Various weigths used in the measures are specified by the program's user to reflect the requirements of the problem, or determined experimentally. For more details about the description quality measure see (Bergadano et al., 1988).

5. LEARNING BY MAXIMIZING THE CONCEPT DESCRIPTION QUALITY

As mentioned before, learning a BCR of a concept description is performed in two phases. In the first phase, a complete and consistent (CC) concept description is generated by inductive learning from examples. In the second phase, the obtained CC description is optimized according to the general description quality (GDQ) criterion. In our approach, the first phase is done using the AQ15 learning program (Michalski et al., 86a). This section describes the second phase.

5.1 Search Heuristics

Applying the GDQ measure directly would be computationally expensive, because it would require the system to perform flexible matching for every newly generated description against the whole set of training examples. To improve the efficiency, we have introduced a *double level* search. The first level uses a simplified measure to determine which operator (SR, CR, RE or RC) is likely to improve the description, and the second level actually applies the operator and evaluates the description using GDQ.

The simplified measure used in the first level is *Potential Accuracy Improvement* heuristic (PAI). PAI is a function of the change in the coverage of positive and negative examples by the description due to an application of an operator.. Specifically:

$$PAI = \#P/\#TP - \#N/\#TN$$

where #P (#N) is the *change* in the number of positive (negative) examples covered by the newly generated description, and #TP (#TN) is the total number of positive (negative) examples. For generalizing operators, SR and RE, #P and #N are non-negative, and for specializing operators CR and RC, #P and #N are non-positive.

The advantage of PAI is that it can be computed much more efficiently than the GDQ. For every selector in the description, a list of examples covered by it is maintained using bit vectors. The sets of examples covered by complexes and covers can be obtained from this list by the intersection and union operations. Matching time can be improved further by maintaining also bit vectors for the examples covered by complexes (the time for the matching operation is traded off against the memory for storing the bit vectors). Note that computing the GDQ requires flexible matching, and thus cannot be done by an intersection and union of bit vectors.

The above formula does not take into consideration the reduction in the complexity of the description due to the application of operators. Specifically, removing a complex (a rule) reduces complexity more than removing a selector (a condition). To account for this, POSEIDON assigns a higher weight (preference) for applying the operator CR (complex removal) than for applying the operator SR (selector removal).

One may observe that removing a selector may enable the system to remove additional complexes afterwards. As such an operator generalizes a description, additional examples may be covered by the reduced complex, and some other complexes may become redundant. A special case of this is when the reduced complex becomes equal to some other complex in the description, and thus one of them can be removed. If the CR operation produces a complex similar to some other complex, the two may be merged into a single one by adding alternatives in the internal disjunctions of their selectors. For example, the complexes [shape = circle]&[size = 2] and [shape = square]&[size = 2] can be replaced by single complex [shape = circle v square]&[size = 2].

It is worth noting that in the case of operators CR (complex removal) and SR (selector removal), the PAI heuristic can be simplified by using an approximation:

$$PAI' = \#P/\#TP - \#N/\#TN$$

where #P (#N) is the number of positive (negative) examples covered by the *component* (complex or selector) to be removed. Such a heuristic is very efficient because it needs to be computed only once for every selector and every complex in the initial description. This computation can be done before the search starts, and does not need to be repeated for every node in the search.

The operator that produces the largest PAI is chosen, and applied to the description under consideration. The descriptions generated on the basis of PAI are then subjected to an evaluation by GDQ.

5.2 Search Algorithm

The search procedure is described by the following algorithm:

1. Identify in the search tree the best description D (one with the highest GDQ). Initially, D is the complete and consistent description obtained in stage 1 (by the AQ15 program)..
2. Apply to D the operator (from among CR_i , SR_{ij} , RE_{ij}, RC_{ij}) that potentially improves the GDQ of D the most, according to the Potential Accuracy Improvement (PAI) measure.
   CR_i. Remove the i-th complex from D.
   SR_{ij}: Remove the j-th selector from the i-th complex in D .
   RE_{ij}: Extend the referent of the j-th selector in the i-th complex in D.
   RC_{ij}: Contract the referent of the j-th selector in the i-th complex in D.
3. Compute the GDQ of the node obtained in step 2. If this GDQ is smaller than the GDQ of D, then proceed to step 1. As mentioned above, when computing the accuracy of a description, flexible matching is used.
4. Ask for an explanation of
   (a) the positive examples that cease to be covered
   (b) the negative examples that become covered
   If such an explanation is given, add the rules that make up the explanation to the ICI, otherwise add the exceptional example to the ICI.
5. Update the GDQ value of the new node, by taking into account the added ICI rules.
6. If the *stopping criterion* is satisfied, then STOP, otherwise proceed to step 1.

We shall now discuss the motivation behind the algorithm, some implementation details, and explain the search strategy. In step 1, the nodes are chosen on the best first basis, that is, the node with the highest GDQ is expanded first. This is not always an optimal choice, because "bad" nodes can sometimes lead to better descriptions after a number of removals. Whether the search will behave in this manner will depend on the adequacy of the GDQ as the measure of concept quality.

In step 2, a search operator is chosen heuristically using the PAI heuristic, and applied to the description. Only one operator is applied at any given time.

In step 3, the system computes the GDQ of the new node. It should be noted that, in the GDQ measure, the typical examples covered directly by the BCR can weigh more than those covered through flexible matching. The examples covered by ICI rules should weigh more than the ones

covered through flexible matching, but less than the ones covered by the BCR.

In step 4, the "explainer" module is used in order to improve the description even further. The BCR description is extended or contracted by adding ICI rules. First, complex removal might uncover some positive examples, that were previously covered. In this case, new rules could be added to the ICI, that would allow the system to reason about such "special" positive examples, and explain why they should still be classified as instances of the concept under consideration. On the other hand, selector removal might cause some negative examples to be covered. In this case, new ICI rules may have to be added to contract the BCR. Another issue concerning step 4 is whether an explanation should be required at all, since, in some cases, the chosen removal operator is not an appropriate one, and will lead to a very poor description. In this case it is not even worth to ask for an explanation, and search can continue in other directions.

The strategy is as follows. Suppose that *n* is the node (description) being expanded and *m* is the node obtained after applying an operator (e.g., the selector removal). The effort to obtain an explanation is made only if the GDQ of *m* is significantly better than that of *n*. In this case, the explainer is given the GDQ evaluations of both descriptions and asked for an explanation. The values of GDQ give the explainer the sense of importance of the request for the search procedure.

In step 5, the GDQ of the obtained TT description is updated after the new ICI rules have been added. Since ICI rules are taken into consideration in computing GDQ, new ICI rules will change the GDQ value for a concept. In step 6, the system decides whether to stop or continue the search. The *stopping criterion* is satisfied when the search space that has been explored is large (more than k1 nodes have been expanded), or when no qualitative improvement has been obtained for a long time (more than k2 nodes have been expanded since the last GDQ improvement). Values of k1 and k2 are parameters of the search. When the system stops, the best node in the search space is produced. The description corresponding to it becomes the chosen TT concept description.

One more characteristic of the search should be mentioned. At any one time only one operator is applied to the selected (best-GDQ) node. The new node can be selected only if its quality is better than the quality of the father node. This is different from standard best first search procedure, in which all the applicable operators are apllied to the node selected for expansion. This choice was introduced because the creation of a new node involves the computation of its quality, which, in some cases, can be time-consuming. To avoid generating low quality nodes, the best applicable operator is selected on a heuristic basis, and only this one is applied. Other operators are used only if the results obtained on this search branch turn out to be unsatisfactory.

5.3 An Abstract Example

An abstract example of the search process is given in Figure 2. The nodes contain BCR, ICI, and a

graphical representation of the covered examples. The tree is kept in the memory throughout the search. The rectangular areas denote complexes (or rules) in the BCR. The modification of the concept borders due to the ICI interpretation is marked by curved continuous lines. To simplify the terminology, a complex is called a rule, and a selector is called a condition (briefly, a cond).

In the example, the accuracy is computed according to the formula discussed in Section 4, assuming the same typicality for all the instances. The initial description is represented by node 1, and contains two rules (complexes). The rules cover two corresponding rectangular areas in the graphical representation, containing five positive examples out of eight, and one negative example out of five. The ICI extends this coverage by recognizing one more positive example. By eliminating condition (selector) s5 in the second complex node 3 is obtained. The accuracy of the description is now improved since all positive examples are covered.
Finally, by truncating the first complex node 5 is generated. It does not cover negative examples any more, and is simpler. This node is then accepted as the improved description resulting from the search. The other nodes lead to inferior concept representations with respect to GDQ, and are discarded. The quality has been computed with $w_1 = w_2 = 0.5$. For clarity, the cost is omitted, and the simplicity of the ICI is not taken into account. The simplicity of the BCR is evaluated by the number of rules and conditions.

6. EXPERIMENTS

POSEIDON was experimentally applied to two different problem domains: labor management contracts and congressional voting record. Specifically, it was used to acquire discriminant descriptions for acceptable/unacceptable labor management contracts, and voting behavior of republican/democrat congresspersons in the U.S. House of Representatives. Below is a brief description of the data used in the experiments.

6.1 Experimental data

Labor management contracts

The data about labor-management contracts was taken from *Collective Bargaining*, a review of current collective bargaining issues published by the Government of Canada through its Department of Labor. The data given in *Collective Bargaining* describes labor-management contracts which have been currently negotiated between organizations and those union locals that count at least 500 members.

The raw data is divided into economic sectors. Each contract is described by a number of attributes. Since the attributes vary between economic sectors, we decided to focus on one sector: personal and business services. This sector includes unions representing hospital staff, teachers, university professors, social workers and certain classes of administrative personnel of various organizations. The data use multivalued attributes, and therefore VL_1 is an obvious choice as a description language.

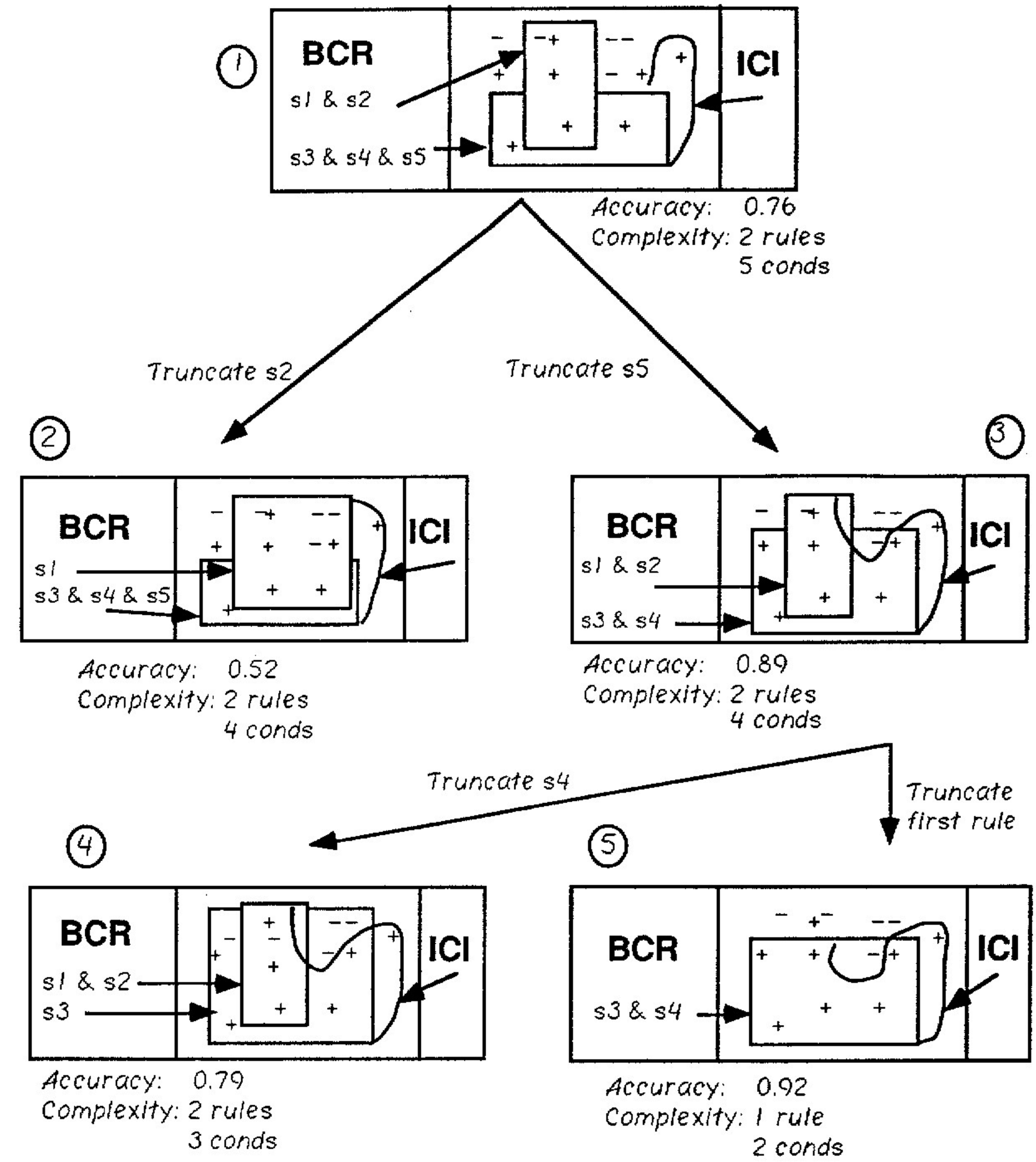

Figure 2. An illustration of the search process.

The data describe contracts finalized in the second half of 1987 and the first half of 1988. Each contract is described by sixteen attributes, belonging to two main categories. One category concerns issues related to salaries, e.g., pay increases in each year of contract, the cost of living allowance, a stand-by pay, etc., and the second category concerns issues related to fringe benefits,

e.g., different kinds of pension contributions, holidays, vacation, dental insurance, etc. Positive examples represent contracts that have been accepted by both parties. Negative examples represent contract proposals deemed unacceptable by one of the parties. The training set consisted of 18 positive and 9 negative examples of contracts; the testing set consisted of 19 positive and 11 negative examples. Here is a typical acceptable labor-management contract:

Duration of the contract = 2 years
Wage increase in the first year = 1.5%
Wage increase in the second year = 3.5%
Cost-of-living-allowance = unknown
Hours of work/per week = 38
Pension offer = none
Stand-by pay = $0.12/hr
Shift differential = second shift is paid 25% more than first shift
Educational allowance is offered
Holidays /per year = 11 days
Vacation offer is better than average in the industry
Long term disability insurance is offered by the employer
50% dental insurance cost is covered by the employer
Bereavement leave is available
Employer-sponsored health plan is not mentioned

which is represented in VL_1 as :

[Dur = 2] [wage1 = 1.5] [Wage2 = 3.5] [cola = unknown] [Work-hours = 38] [Pension = none] [Stby-pay = 12] [Shift-diff = 25] [Educ-allw = true] [Holidays = 11] [Vacation = better][lngtrm-disabil = true] [Dntl-ins = half] [Bereavement = true] [Empl-hplan = unknown]

U.S. Congress voting record

The second application was concerned with the U.S. Congress voting record. We have relied on the same data set as used by Lebowitz (1987) in the experiments on conceptual clustering. The data represents the 1981 voting record for 100 selected representatives. The data set was split randomly into a training and testing set, with voting records of democrats entered as positive examples, and voting records of republicans entered as the negative ones. The goal was to obtain discriminant descriptions characterizing democrat and republican congressmen. Here is an instance of U.S. Congress voting record of a democratic congressman:

Draft registration = no
Ban of aid to Nicaragua = no
Cut expenditure on MX missiles = yes

Federal subsidy to nuclear power stations = yes
Subsidy to national parks in Alaska = yes
Fair housing bill = yes
Limit on PAC contributions = yes
Limit on food stamp program = no
Federal help to education = no
State = north east
Population = large
Occupation = unknown
Cut in Social Security spending = no
Federal help to Chrysler Corp. = vote not registered

6.2 Description of Experiments and Illustration of Results

The above data were used in a series of experiments on testing the proposed method, the previous method implemented in AQ15, an exemplar-based method and the method based on pruned decision trees. The experiments included the following steps:

1. Learning a CC description from the training examples by applying the AQ15 program.

2. Determining the "top rule" description on the basis of the CC description, using the TRUNC method (Michalski et al., 1986). Such a description consists of a single rule selected from the CC description generated by the AQ15 program. It is the rule that covers the maximum number of positive examples among all other rules in the CC description. Top rule description is easy to determine, because the AQ15 generates rules together with measures indicating the number of examples covered in toto and uniquely by each rule in the description (the so called t-weight and u-weight; see below). In the method, one top rule description was generated for the positive examples of the concept, and one for the negative examples, i.e., from a CC description of the negative examples. An instance was classified to the concept if it best matched the top rule description of positive examples, and was rejected if it matched the top rule description of the negative examples. If both descriptions were matched with roughly the same degree of match, than the instance was classified as "no match." Using such a description with flexible matching represents a simple but important version of the TT method (Michalski, 1990).

3. Optimizing the CC description using the POSEIDON TRUNC-SG procedure.
4. Testing all the above descriptions (CC, Top rule and Optimized TT rules) on the testing examples.
5. For comparison, we also tested descriptions obtained by other methods, specifically, a simple

exemplar-based learning approach and the decision tree learning algorithm ASSISTANT.

To illustrate the difference between the CC descriptions generated by AQ15, the *top rule* and the optimized descriptions created by POSEIDON, below is a sample of these descriptions in the labor management domain. The input data consisted of 18 positive and 9 negative examples of acceptable labor contracts. Figure 3 shows the complete and consistent description produced by AQ15. In the figure, t represents the t-weight, which is the total number of examples covered by a rule, and u represents the u-weight, which is the number of examples uniquely covered by the rule.

```
[contract-duration >1]&[wage_incr_yr2 >3.0%]&[#holidays >10]:   (t = 11, u = 11)
 or
[wage_incr_yr1 > 4.5%]:                                        (t = 4, u = 4)
 or
[wage_incr_yr1 > 4.0%] & [wage_incr_yr2 > 4.0%]:               (t = 1, u = 1)
 or
[wage_incr_yr1 > 4.5%] & [#holidays > 9]:                      (t = 1, u = 1)
 or
[wage_incr_yr1 > 2.0%] & [vacation = above_average]:           (t = 1, u = 1)
    ::>  Acceptable  contract

[wage_incr_yr1 = 2.0% .. 4.0%] & [#holidays < 10] &
[vacation = below_average v average]:                          (t = 3, u = 3)
 or
[wage_incr_yr1 ≤ 4.5%] & [wage_incr_yr2 ≤ 4.0%] &
[holidays = 10] & [vacation = below_average v average]:        (t = 2, u = 2)
 or
[duration = 1] & [wage_incr_yr1 < 4.0%] & [#holidays < 10]&
[vacation = below_average v average]:                          (t = 1, u = 1)
[wage_incr_yr1 ≤ 4.0%] & [wage_incr_yr2 ≤ 3.0%] &
[vacation = below_average v average]:                          (t = 1, u = 1)
 or
[duration = 1] & [wage_incr_yr1 ≤ 4.0%]&
[vacation = below_average v average]:                          (t = 1, u = 1)
 or
[wage_incr_yr1 = 2.0%] & [wage_incr_yr2 ≤ 3.0%]:               (t = 1, u = 1)
    ::> Unacceptable  contract
```

Figure 3. The CC descriptions generated by AQ15

The top rule (Figure 4) is obtained by selecting the rule with the largest t-weight from the CC description of each concept.

BCR:

```
[contract-duration >1]&[wage_incr_yr2 >3.0%]&[#holidays >10]:   (t = 11,u = 11)
    ::>  Acceptable  contract

[wage_incr_yr1 = 2.0% .. 4.0%] & [#holidays < 10] &
[vacation = below_average v average]:                          (t = 3, u = 3)
    ::> Unacceptable  contract
```

ICI: Flexible matching

Figure 4 . The top rules generated by TRUNC method

By applying the TRUNC-SG method to the CC description, the following optimized TT description was obtained (Figure 5).

BCR:

```
[wage_incr_yr1 > 4.5%] v
[wage_incr_yr2 > 3.0%] v
[#holidays > 9] v
[vacation period = above_average] ::>
                                    Acceptable  contract
[wage_incr_yr1 ≤ 4.0%] & [#holidays < 10]v
[wage_incr_yr2 ≤ 4.0%] & [vacation = below_average v average] v
[#holidays < 10] v
[duration = 1] & [wage_incr_yr1 ≤ 4.0%]v
[wage_incr_yr2 ≤ 3.0%]  ::>
                                  Unacceptable  contract
```

ICI:

```
flexible matching and deductive matching using rules:

[wage_incr_yr1≥5.5%]&[vacation = below_average] ::>
                                          Acceptable  contract
[wage_incr_yr1≤3%]&[wage_incr_yr2<wage_incr_yr1] ::>
                                          Unacceptable  contract
[wage_incr_yr1≤3%]&[wage_incr_yr2≤3%]&
[hours_work ≥ 40]&[pension = empl_contr] ::>
                                          Unacceptable  contract
```

Figure 5. Optimized TT descriptions obtained by POSEIDON using the TRUNC-SG procedure

The BCR of the optimized descriptions are significantly simpler than the CC descriptions generated by AQ15. They seem to represent the most important characteristics of the labor management contracts: a contract is acceptable when it offers a significant wage increase (the first two complexes in Fig. 5), or it offers many holiday days, or the vacation is above average.

The training events that were not correctly classified by the BCR, as it was modified step by step during the search, were analyzed by a domain expert, who provided deductive rules allowing the system to classify almost all the training events (one of them could not be explained by the expert).

Let us consider the ICI rule:

```
[wage_incr_yr1<3.1%]&[wage_incr_yr2<wage_incr_yr1] ::>
                                        Unacceptable  contract
```

The rule addresses the case of a contract with a low wage increase in the first year, and an even lower increase in the second year. In those circumstances, the holiday and vacation offered do not matter and the contract is unacceptable by the union.

6.3 Results of Experiments

Four experiments have been performed. In each experiment, the number of events correctly and incorrectly recognized by both descriptions was determined, as well as the number of events that were not covered by either concept. This was done both on the training set and on a testing set of examples (not previously seen by the system). The results of the three experiments are summarized in Tables 3 to 6. Correct (incorrect) states the percentage of the number of testing events that were correctly (incorrectly) classified. The *No_Match* counts examples that matched no rules. In each experiment, the same training and testing sets were used. Moreover, the simplicity of the acquired descriptions is shown, since it strongly affects their comprehensibility and cost.

The first experiment (Table 3) used an exemplar-based concept description, i.e., a disjunction of the training events, also called a *factual* description. This description is obviously complete and consistent with regard to the training set. The first part of Experiment 1 applied this description to the testing examples using the *Strict Match* method (i.e., a new example must match exactly some training example to be classified). In this case, as expected, the description had almost no predictive power. It produced a No_Match answer for all testing examples of the labor contract data (100% No-Match), and for 96% testing examples of the congress voting data (2 examples were the same in training and testing sets).

Next parts of Experiment 1 tested simple forms of an exemplar-based learning method. The *1-Nearest Neighbor* row in the table lists results from applying the factual description with the matching method similar to the one described in (Kibler and Aha 1987). A given testing example is matched against all training examples and the example that gives the best fit is found. The class of this "best fit" training example is assigned to the testing example. Our measure of fit differed slightly from the one described in (Kibler and Aha, 1987), as the latter uses the *maximum* function for evaluating a disjunction, while our flexible matching uses the *probabilistic sum* (section 2.2). We also tested other nearest neighbor methods (3-N and 5-N), in which instead of one, several best fit training examples are used to determine the class of the testing example (applying the majority rule).

Simple Exemplar-based Description

Labor-mgmt problem (Labor): 27 rules and 432 conditions
Congress problem (Congress): 51 rules and 969 conditions

	Correct		Incorrect		No_Match	
	Labor	*Congress*	*Labor*	*Congress*	*Labor*	*Congress*
Strict Match						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	0%	4%	0%	0%	100%	96%
1-Nearest Neighbor						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	77%	**86%**	23%	14%	0%	0%
3-Nearest Neighbors						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	**83%**	84%	17%	16%	0%	0%
5-Nearest Neighbor						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	80%	84%	20%	16%	0%	0%

Table 3. Results of Experiment 1.

The second experiment (see *Table 4*) used concept descriptions generated by AQ15 without truncation. Such descriptions are consistent and complete (CC) with regard to the training examples, i.e., they classify all training examples 100% correct when using the strict matching method. The flexible matching method did not change this result. For the testing set, the number of correct classifications was relatively high (80-86%), the same for the strict and flexible matching methods. Flexible matching made no difference, probably due to two factors. Firstly, the CC descriptions include many specific rules, leaving little space for the "no match" cases (3%), in which flexible matching could help. Secondly, the descriptions consisted of only disjoint rules, as the program was run using the "disjoint cover" parameter. In such a situation, the "multiple match" cases do not occur, and flexible matching cannot help.

The above results are similar to those obtained in the previous experiment, which used an exemplar-based approach (*Table 3*). The main difference is that the AQ descriptions are much simpler in terms of the number of rules and the number of conditions involved (11 vs. 27 rules in the labor management problem, and 10 vs. 51 rules in the congress voting problem). The simpler descriptions allow the system to be more efficient in the recognition mode.

Complete and Consistent Description (No truncation)

Labor-mgmt problem (Labor): 11 rules and 28 conditions
Congress problem (Congress): 10 rules and 32 conditions

	Correct		Incorrect		No_Match	
	Labor	*Congress*	*Labor*	*Congress*	*Labor*	*Congress*
Strict Match						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	80%	86%	17%	14%	3%	0%
Flexible Match						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	80%	86%	17%	14%	3%	0%

Table 4. Results of Experiment 2.

The third experiment (Table 5) tested the *top rule* descriptions determined from the above CC descriptions, as described above. As shown on Table 5, the performance of this rule on testing examples using flexible matching was comparable to that of the CC and factual descriptions generated in the previous experiments. A significant difference, however, between the *top rule* description and the previous two was in the complexity of the description (2 vs.11 vs. 27 rules in the Labor Management problem, and 2 vs. 10 vs. 51 rules in the Congress Voting problem). Each concept is described only by one rule. Two concepts are involved in both domains, so each domain is described by two rules. It is quite interesting that such simple rules perform as well as much more complex descriptions generated in previous experiments.

The fourth experiment (Table 6) tested the optimized descriptions generated by the method implemented in POSEIDON, i.e., based the TRUNC-SG method. The descriptions were tested using flexible matching alone (*Flexible Match*) and in the combination with deductive matching (*Deductive Match*). For comparison, the performance of these descriptions using strict matching (*Strict Match*) was also evaluated (though this would be an impractical combination). As expected, these descriptions used with strict matching give relatively poor performance.

The optimized descriptions (BCR) combined with deductive matching (ICI) gave the best performance (90-92% correct). When used with only flexible matching, the performance is slightly lower. The descriptions are simpler than CC descriptions, although they include the ICI rules. !! They are more complex than the top rule descriptions, which are the simplest.

The Top Rule Description (the TRUNC method)

Labor-mgmt problem (Labor): 2 rules and 6 conditions
Congress problem (Congress): 2 rules and 6 conditions

	Correct		Incorrect		No_Match	
	Labor	*Congress*	*Labor*	*Congress*	*Labor*	*Congress*
Strict Match						
Training Set	52%	62%	0%	0%	48%	38%
Testing Set	63%	69%	7%	7%	30%	24%
Flexible Match						
Training Set	81%	75%	19%	25%	0%	0%
Testing Set	**83%**	**85%**	17%	15%	0%	0%

Table 5. Results of Experiment 3.

Optimized Description *(the TRUNC-SG method)*

Labor-mgmt problem (Labor): 9 rules and 12 conditions
Congress problem (Congress): 10 rules and 21 conditions

	Correct		Incorrect		No_Match	
	Labor	*Congress*	*Labor*	*Congress*	*Labor*	*Congress*
Strict Match						
Training Set	63%	84%	0%	0%	37%	16%
Testing Set	43%	73%	3%	4%	54%	23%
Flexible Match						
Training Set	85%	100%	0%	0%	15%	0%
Testing Set	83%	92%	13%	8%	4%	0%
Deductive Match						
Training Set	96%	96%	0%	4%	4%	0%
Testing Set	**90%**	**92%**	10%	8%	0%	0%

Table 6. Results of Experiment 4.

For the Labor data problem, descriptions applied with deductive matching produced higher performance than when used with flexible matching only (90 vs. 83%)6. For the Congress data problem, the performance was the same for the two matching methods. This is because deductive rules were acquired on the training set; in the specific testing set, the D-covered events were the same as F-covered ones.

Table 7 summarizes the results of an experiment comparing the performance of descriptions generated by simple exemplar-based methods, the TT descriptions generated by POSEIDON, and

pruned decision trees generated by ASSISTANT (a descendant of the Quinlan's ID3 program; Cestnik et al., 1987). ASSISTANT was applied with the same learning and training data, which were used in the experiments reported in Tables 3, 4, 5, and 6. The decision trees obtained by ASSISTANT were optimized using a tree-pruning mechanism (Cestnik et al., 1987). This mechanism is compared with the TRUNC-SG method in the next section.

The performance of concept descriptions is measured by the percentage of testing events recognized correctly by a description. The factual description (a disjunction of training examples) was applied with the flexible matching function. The complexity of a rule-based description was measured by stating the number of rules (#Rules) and the number of conditions (#Conds). The complexity of a decision tree was measured by the number of leaves (#Leaves) and the number of nodes (#Nodes). The number of rules in a rule-based description can be taken as comparable with the number of leaves in a decision tree, because for each leaf of the tree one can generate one rule (by tracing the nodes from the root to the leaf).

The above experiments, on both domains, show that the learning method implemented in POSEIDON produces descriptions that are much simpler and perform slightly better on the testing data. Being simpler, they will also be easier to understand and the cost of evaluating them will be lower. The complete meaning of the concept defined by such a description depends on the BCR, the optimized description generated through learning, and the ICI (which consists of the flexible matching procedure, defined a-priori, and a set of deductive rules, formulated by the expert).

The ICI also contributes to the comprehensibility of the descriptions. The expert defines the rules on the basis of misclassified examples, which are determined by the system during the learning process. The "top rule" description was much simpler than any other description, but performed worse than the optimized description and the decision tree. Depending on the desired trade-off, the "top rule" or the optimized description can be taken as the BCR of the concept being defined[5].

If the input data include the typicality of the examples, the method will generate the BCR which will tend to cover typical examples, and the ICI which will tend to cover less typical examples. When the typicality information is unavailable, as in our experiments, the system will propose a classification of examples to different classes of typicality. The examples covered by the BCR can be considered as typical, those covered by flexible matching as nearly-typical, and those covered by the deductive rules as non-typical.

---

6 The parameters of Poseidon, such as the simplicity/accuracy trade-off, have been set manually without much fine-tuning. Nevertheless, we have not tried to run the system with many parameter settings or with average values.

	Labor Contract	Congress Voting
Simple exemplar-based method		
Performance **(%Correct / % Incorrect)**		
1-nearest neighbor	77% / 23%	86% / 14%
3-nearest neighbor	83% / 17%	84% / 16%
5-nearest neighbor	80% / 20%	84% / 16%
Complexity **(#Rules / #Conds)**	27 / 432	51 / 969
Pruned decision tree (ASSISTANT + PRUNING)		
Performance **(%Correct / % Incorrect)**	86% / 14%	86% / 14%
Complexity **(#Leaves/ #Nodes)**	29 / 53	19 / 28
Complete and consistent description (AQ15 without rule truncation)		
Performance (%correct / % incorrect)	80% / 17%	86% / 14%
Complexity **(#Rules / #Conds)**	11 / 29	10 / 32
Top rule TT description (AQ15 with rule truncation)		
Performance **(%Correct / % Incorrect)**	83% / 17%	85% / 15%
Complexity **(#Rules / #Conds)**	2 / 6	2 / 6
Optimized TT description (POSEIDON)		
Performance **(%Correct / % Incorrect)**	90% / 10%	92% / 8%
Complexity **(#Rules / #Conds)**	9 / 12	10 / 21

Table 7. Summary of the results of testing descriptions generated by different methods

Another aspect of the proposed method is its potential ability to handle noise. Noisy examples are usually covered by the "light" rules (i.e., complexes that cover few examples). By removing these rules and unimportant conditions the effect of noise is minimized (Zhang and Michalski, 1989). Future research should investigate this aspect of the method to a greater detail. There is also need to determine the impact of the typicality of the examples on the quality of generated concept descriptions.

7. RELATED WORK

The research presented here relates to several other efforts on learning imprecise concepts, in particular, to methods learning pruned decision trees (e.g. Quinlan, 1987; Cestnik et al, 1987; Fisher and Schlimmer, 1988). In these methods, a concept description is a single tree structure ("one tier") that is supposed to account for all concept instances. An unknown instance is strictly matched against all candidate concept descriptions. Since pruned decision trees do not cover some of the training examples, they always produce some error on the training examples. In contrast, the TT method presented here can compensate for he lack of coverage by one tier (BCR) by an application of the second tier (ICI), either by using flexible matching or deductive inference rules. In addition, our approach is capable of taking into account the typicality of the examples, if it is provided in the training data. This feature gives the method an additional strength in handling noisy examples.

The method presented in (Quinlan, 1987) is based on a hill-climbing approach that first truncates conditions and then rules. No search is performed, only one alternative truncation is tried at every step and the final result may not be the best possible, although the procedure should be faster than the POSEIDON system. In the same paper (Quinlan, 1987) other methods for pruning decision trees are also described. Some of these methods require a separate testing set for the simplification phase, others use the same training set that was used for creating the tree. The simplification phase in the POSEIDON system can be done either with the original training set or with a separate set of examples.

The experiments by Fisher and Schlimmer (1988) on pruning decision trees are based on a statistical measure to determine the attributes to be pruned. Such measures require a sufficient data sample, and do not apply well to small training sets. In the TT approach, training events are analyzed logically rather than statistically, both in the phase of creating a complete and consistent description and when covered by the ICI rule base. Consequently, the TT method seems to be more adequate for learning with a relatively small number of examples. It could be interesting to study an integration of the two approaches.

The system developed by (Iba et al. 1988) uses a trade-off measure that is somewhat similar to the GDQ measure proposed in this paper. The GDQ measure, however, considers more factors.

Besides taking into account the typicality of the instances covered by the description, it considers the type of matching between an instance and a description. Moreover, the simplicity measured by the GDQ depends not only on the number of disjuncts in the description as in (Iba et al. 1988), but also on the different syntactic features of the terms in the description.

The CN2 inductive algorithm (Clark and Niblett, 1989) uses a heuristic function to terminate search during rule construction. The heuristic is based on an estimate of the noise present in the data. Such pruning of the search space of inductive hypotheses results in rules that may not classify all the training examples correctly, but that performs well on testing data. CN2 can be viewed as an induction algorithm that includes pre-truncation, while the algorithm reported here is based on post-truncation. CN2 applies truncation during rule generation and POSEIDON applies truncation after rule generation. The advantage of pre-truncation is efficiency of the learning process, but irrelevant conditions and redundant complexes generated during learning are not removed.

The exemplar-based learning system PROTOS (Bareiss, 1989) is similar to POSEIDON in the sense that both systems use a sophisticated matching procedure -- a knowledge based matching of an event with a concept description and acquiring the matching knowledge via explanations of training events provided by a teacher. There are, however, major differences: 1) PROTOS stores exemplars as base concept descriptions, whereas POSEIDON generates simple and easy-to-understand generalizations as base concept descriptions, 2) PROTOS uses domain knowledge in classifying all new cases, whereas POSEIDON uses ICI rules only for classifying exceptions, 3) During the learning process, PROTOS asks the teacher for explanations for all exemplars, whereas POSEIDON only asks for explanations of exceptions.

The problem of defining and using the typicality of examples has been considered in the past both in machine learning and cognitive science. Negative examples of low typicality are referred to as *near misses* in Winston's system (1975). Such examples are used in Winston's system to delineate the borders of a concept. Michalski and Larson (1978) introduced the idea of an outstanding representative of a concept. The concept of prototypical examples has been also studied by (Smith and Medin 1981) and by (Rosch and Mervis 1975). The TT representation gave us also a unique opportunity to define precisely the concept of exceptions and nearly-representative examples (see section 2.4) as those that are covered by the deductive rules and flexible matching, respectively.

To summarize, there are five major differences between the work presented here and related research described in the literature. First, the above method does not lose coverage, although it still yields a simpler description with improved predictive power. Second, it simplifies the description by performing independently both generalization and specialization. Third, any part of the description may be truncated in the simplification process. Fourth, the method takes into account

the typicality of the examples and a general description quality measure is used. Finally, it uses both flexible matching and deductive rules for events that are not covered explicitly.

It may seem interesting to situate the TT method introduced here in the spectrum of existing machine learning approaches, such as simple inductive techniques and case-based reasoning. Table 9 below describes these methods, as well as the two-tiered approach in terms of the type of concept representation, and the kind of matching applied for classification.

	Simple Induction	**Exemplar-based**	**Two-tiered**
Representation	General	Specific	General
Matching	Precise	Inferential	Inferential

Table 9. Comparison of the two-tiered approach with simple inductive and exemplar-based methods

8. SUMMARY AND OPEN PROBLEMS

In contrast to existing learning methods in which concepts are represented as single knowledge structures, POSEIDON assumes a two-tiered (TT) concept representation. In such a representation, the first tier, the base concept representation (BCR), captures the explicit and common concept meaning. The second tier, the inferential concept interpretation (ICI) defines allowable modifications of the base meaning and exceptions. This way, typical concept instances match BCR and thus can be recognized efficiently.

In POSEIDON, the BCR is obtained in a two-phase process. First, a complete and consistent description is learning from a conventional learning program (AQ15). Next, this description is optimized according to a general description quality measure (GDQ). This is done by a double-level search process that uses both generalization or specialization operators. The GDQ takes into account not only the properties of the BCR , but it also involves the ICI (e.g., complexity and accuracy of the total description). The ICI has two components, one, which specifies a flexible matching function, and the second, which specifies a rule base for handling exceptions and context-dependency. The rules can be of two types. The rules of the first type extend the meaning of the concept, and rules of the second type contract the meaning defined by BCR. The first type of rules are employed when an instance does not satisfy the BCR (is not S-covered), nor is covered by the flexible matching function (is not F-covered). The rules of the second type are used when an instance covers a BCR of more than one concept, or when there is a need for confirming the concept membership. In both cases the rules are used deductively. An advantage of the rules over other methods of matching is that they provide an explanation why a given instance is or is

not a member of the concept.

The experimental results have strongly supported the hypothesis that TT concept descriptions can be simpler and easier to understand than "single-tier" descriptions. TT descriptions may also perform better. For example, the obtained TT descriptions for the acceptable labor managements contracts gave the performance of over 90% correct and used only about 9 rules. In contrast, the best performance of a simple exemplar based method gave the 80% correct predictions on new examples and used 27 rules, and a pruned decision tree gave the performance of 86% and was equivalent to 29 rules. The system also performed better than the previous method based on the TRUNC procedure in terms of the performance (80%), but at the cost of a more complex concept description. Although these differences may not be considered very large, we prefer TT descriptions because they are simpler and often based on explicit domain knowledge.

There is a number of advances and differences of the method presented here, compared to previous work on learning TT concept representations (Michalski et al, 1986). That earlier approach produced some preliminary results in which flexible matching function was applied during the testing phase. The same research investigated the effect of truncating concept descriptions. In the system presented here, though, the flexible matching function is augmented by a set of rules defining how to extend or modify a concept description at the "knowledge level" (Dietterich 1986). In the TRUNC approach (Michalski et al., 1986) truncations of the description involved only rule removal operator. This is in contrast with POSEIDON, which is based on the search for two-tiered descriptions using three operators.

An important issue for future research and improvement of the implemented system is the integration of the search procedure with the inductive learning system used to generate the initial description (AQ). The first step in this direction is being experimented with: it allows the two systems to share the same heuristics and the same measure of quality. Further progress is related to the possibility of obtaining partially incomplete and inconsistent description also during the generation of the initial description. The problem of learning second tier rules has to be addressed in the future. In the method currently developed by (Plante and Matwin 1990), ICI rules are learned using chunking in an environment in which multiple explanations for both positive and negative training events are provided.

Another issue to cosider in the future is constructive induction (Michalski 1983), that the use of background knowledge to construct new attributes and higher level descriptions. Constructive induction generally produces descriptions that are easier to understand, and capture the salient features of the concept. The effects of this type of induction are even more relevant for our method. Constructive induction may fold several disjuncts into a single one, regardless of how many examples they cover. This feature of constructive induction may prevent removal of relevant complexes.

The system in its current form does not address the problems of dynamically emerging hierarchies of concepts. The system only learns one concept at a time, and concepts do not change or split as new examples become available. Another open issue is the ability of the system to self-reorganize. The distribution of knowledge between the BCR and the ICI will be determined by the performance of the system on large testing sets. If it turns out, e.g., that some ICI rules are used very often, then these rules could be compiled into explicit BCR assertions. It seems, therefore, that in concept representation one parameter can be traded against the other, within certain limits. This interesting research problem merits further investigation.

This paper has been focusing on attributional descriptions, and the experiments that were presented do not deal with relationships among objects in the examples. An important topic for future research is to develop methods for learning TT structural descriptions. A simple solution would be to replace the AQ15 program by a version of INDUCE (e.g., Michalski, 1983) for learning the initial CC description. The basic search procedure would essentially be the same, but would involve a more complex knowledge representation. Such a representation would allow additional description modification operators. Also, the computation of the GDQ of descriptions would need to be modified, and flexible matching would need to be extended to handle structural concept descriptions.

As practical problems often require only attributional descriptions and the method presented is domain independent, the POSEIDON program has a potential to become a useful new tool for knowledge acquisition and rule learning in a variety of applications.

ACKNOWLEDGEMENTS

The authors express their gratitude to Hugo de Garis, Attilio Giordana, Ken Kaufman, Franz Oppacher, Lorenza Saitta, Gail Thornburg and Gheorghe Tecuci for many useful comments and critisism. They also thank the reviewers of Machine Learning Journal for careful review and for pointing out aspects of the earlier version that needed improvement. Zbig Koperczak has provided help in acquiring the data used in the experiments.

This research was done in the Artificial Intelligence Center of George Mason University. Research activities of the Center are supported in part by the Defense Advanced Research Projects Agency under grant No. N00014-87-K-0874 administered by the Office of Naval Research, and in part by the Office of Naval Research under grants No. N00014-88-K-0226 and No. N00014-88-K-0397. The first author was supported in part by the Italian Ministry of Education (ASSI), and the second author was supported in part by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

Bareiss, R., "An Exemplar-based Knowledge Acquisition", *Academic Press*, 1989.

Bergadano, F., Giordana, A., "Pattern Classification: An Approximate Reasoning Framework", *International Journal of Intelligent Systems*, 1989.

Bergadano, F., Matwin, S., Michalski, R. S., Zhang, J., "Measuring Quality of Concept Descriptions", P*roc. Third European Working Sessions on Learning*, pp. 1-14, Glasgow, 1988.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., Freeman, D., "AutoClass: A Bayesian Classification System", *Procs. of the Fifth Int'l. Conf. On Machine Learning*, Ann Arbor, pp. 54-64 (1988).

Cestnik, B., Kononenko, I., Bratko, I., "ASSISTANT 86: A Knowledge-elicitation Tool for Sophisticated Users", *Procs. of the 2nd European Workshop on Learning*, pp. 31-45 (1987).

Clark, P. Niblett, T., "The CN2 Induction Algorithm", Machine Learning, vol. 3, No.4, , pp. 261-183, 1989.

Collins, A. M., Quillian, M. R., "Experiments on Semantic Memory and Language Comprehension" in *Cognition, Learning and Memory*, L. W. Gregg ed., John Wiley, 1972.

DeJong, G., Mooney, R., "Explanation-based Learning: An Alternative View", *Machine Learning*, vol. 1, No. 2, 1986.

Dietterich, T., "Learning at the Knowledge Level", *Machine Learning Journal*, vol. 1, No. 3, pp. 287-315, 1986.

Dietterich., T., Flann, N., "An Inductive Approach to Solving the Imperfect Theory Problem", *Procs. of the Explanation-based Learning Workshop*, Stanford University, pp. 42-46, 1988.

Fisher, D. H., Schlimmer, J. C., "Concept Simplification and Prediction Accuracy", *Procs. of the Fifth Int'l. Conf. On Machine Learning*, Ann Arbor, pp. 22-28 (1988).

Hammond, K., "Case-based Planning: Viewing Planning as a Memory Task", Academic Press, 1989.

Iba, W., Wogulis, J., Langley, P., "Trading off Simplicity and Coverage in Incremental Concept Learning", *Proceedings. of the Fifth Int'l. Conf. on Machine Learning*, Ann Arbor, pp. 73-79

(1988).

Kedar-Cabelli, S. T., McCarthy, L. T., "Explanation-based Generalization as Resolution Theorem Proving", *Procs. of the 4th Int. Workshop on Machine Learning*, Irvine, 1987.

Kibler, D., Aha, D., "Learning Representative Exemplars of Concepts", *Procs. of the 4th Int. Workshop on Machine Learning*, Irvine, 1987.

Kolodner, J., ed., *Proceedings of the Case-based Reasoning Workshop*, DARPA, Clearwater Beach, FL, 1988.

Lakoff, G., *Women, Fire, and Dangerous Things: What Categories Reveal about Mind*, University of Chicago Press, 1987.

Lebowitz, M., "Experiments with Incremental Concept Formation: UNIMEM", *Machine Learning Journal*, vol. 2, No. 2, 1987.

Michalski, R.S., Larson, J. B., "Selection of Most Representative Training Examples and Incremental Generation of VL_1 Hypotheses: the Underlying Methodology and the Description of Programs ESEL and AQ11," TR 867, Department of Computer Science., University of Illinois at Urbana-Champaign, 1978.

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

Michalski, R.S., Stepp, R.E., "Learning from Observation: Conceptual Clustering", in *Machine Learning: An Artificial Intelligence Approach,* Michalski, R. S., Carbonell, J. G., Mitchell, T. M. (Eds.), Tioga Pub. Co., Palo Alto, CA, 1983.

Michalski, R. S., Mozetic, I., Hong, J., Lavrac, N., "The Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains", Proc. of the 5th AAAI, pp. 1041-1045, 1986.

Michalski, R. S., "Two-Tiered Concept Meaning, Inferential Matching and Conceptual Cohesiveness," in *Similarity and Analogy*, S. Vosniadou and A. Ortony (Eds), Cambridge University Press, 1989.

Michalski, R. S., Ko, H., "On the Nature of Explanation, or Why Did the Wine Bottle Shatter", AAAI Symposium: Explanation-Based Learning, Stanford University, pp. 12-16, 1988.

Michalski, R. S., "How to Learn Imprecise concept: A Method Employing a Two-Tiered Representation for Learning", Procs. of the Fourth International Workshop on Machine Learning, Irvine, CA, pp. 50-58, 1987.

Michalski, R. S., Learning Flexible Concepts: Fundamental Ideas and a Methodology, in Machine Learning: An Artificial Intelligence Approach Volume III, Y. Kodratoff and R. S. Michalski (eds.), Morgan Kaufmann Publishers, 1990 (in press).

Mooney, R., Ourston, D., "Induction Over the Unexplained: Integrated Learning of Concepts with Both Explainable and Conventional Aspects", Procs. of 6th Int'l Workshop on Machine Learning, Ithaca, NY, pp. 5-7, 1989.

Minsky, M., "A Framework for Representing Knowledge", in The Psychology of Computer Vision, ed. P. Winston, (1975)

Mitchell, T. M., Keller, R., Kedar-Cabelli, S., "Explanation-based Generalization: a Unifying view," Machine Learning Journal, vol. 1, No. 1, pp. 11-46, 1986.

Mitchell, T. M., "Version Spaces: An Approach to Concept Learning", Ph. D. dissertation, Stanford University, December 1977.

Plante, B., Matwin, S., "Learning Second Tier Rules by Chunking of Multiple Explanations", Research Report, Department of Computer Science, University of Ottawa, 1990.

Prieditis, A. E. and Mostow, J., "PROLEARN: Towards A Prolog Interpreter that Learns," Proc. of IJCAI 87, Milan, pp. 494-498, 1987.

Quinlan, J. R., "Simplifying decision trees", Int. Journal of Man-Machine Studies, vol. 27, pp. 221-234, 1987.

Robinson J. A., Sibert E. E., "LOGLISP: An Alternative to Prolog", Machine Intelligence, vol. 10, Hayes, J. E. and Michie, D. (Eds.), pp. 399-419, 1982.

Rosch, E., and Mervis, C. B., "Family Resemblances: Studies in the Internal Structure of Categories", Cognitive Psychology, vol. 7, pp. 573-605, 1975.

Smith, E. E., Medin, D. L., "Categories and Concepts", Harvard University Press, 1981.

Sowa, J. F., "Conceptual Structures", Addison Wesley, 1984.

Sturt, E., "Computerized Construction in Fortran of a Discriminant Function for Categorical Data", Applied Statistics, vol. 30, pp. 213-222, 1981.

Watanabe, S., "Knowing and Guessing, a Formal and Quantitative Study",Wiley Pub. Co., 1969.

Weber, S., "A General Concept of Fuzzy Connectives, Negations and Implications based on t-norms and t-conorms", Fuzzy Sets and Systems, vol. 11, pp. 115-134, 1983.

Winston, P. H., "Learning Structural Descriptions from Examples", in "The Psychology of Computer Vision", P. Winston, ed., McGraw-Hill, 1975

Zadeh, L. A., "Fuzzy Logic and its Applications to Approximate Reasoning", Information Processing, North Holland, pp. 591-594, 1974.

Zhang, J. and Michalski, R. S., "Rule Optimization via SG-Trunc Method", Proc. Fourth European Working Sessions on Learning, 1989.