Machine Learning, 8, 5–43 (1992)

Learning Two-Tiered Descriptions of Flexible Concepts: The POSEIDON System

F. BERGADANO,[1]
S. MATWIN,[2] (STAN@CSI.UOTTAWA.CA)
R.S. MICHALSKI MICHALSKI@AIC.GMU.EDU
J. ZHANG
Artificial Intelligence Center, George Mason University, Fairfax, VA 22030

Editor: Jaime Carbonell

Abstract. This paper describes a method for learning *flexible* concepts, by which are meant concepts that lack precise definition and are context-dependent. To describe such concepts, the method employs a *two-tiered* representation, in which the first tier captures explicitly basic concept properties, and the second tier characterizes allowable concept's modifications and context dependency. In the proposed method, the first tier, called *Base Concept Representation* (BCR), is created in two phases. In phase 1, the AQ-15 rule learning program is applied to induce a complete and consistent concept description from supplied examples. In phase 2, this description is optimized according to a domain-dependent quality criterion. The second tier, called the *inferential concept interpretation* (ICI), consists of a procedure for *flexible matching*, and a set of inference rules. The proposed method has been implemented in the POSEIDON system, and experimentally tested on two real-world problems: learning the concept of an acceptable union contract, and learning voting patterns of 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 an ID-3-type decision tree learning, implemented in the ASSISTANT program. In the experiments, POSEIDON generated concept descriptions that were both, more accurate and also substantially simpler than those produced by the other methods.

Keywords. Concept learning, learning imprecise concepts, inductive learning, learning flexible concepts, two-tiered concept representation, flexible matching

1. Introduction

Most current methods of machine learning, both empirical and analytic, assume that concepts are precise and context-independent, and representable by a single symbolic description. An important consequence of this assumption is that the recognition of such concepts, which we call *crisp*, is 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 in typicality among instances.

In some methods, these assumptions are partially relaxed by assigning to a concept a fuzzy set membership function (e.g., Zadeh, 1974) or a probability distribution (e.g., Cheeseman, et al., 1988; Fisher, 1987). However, once such a measure is defined explicitly for a given concept, the concept again has a fixed, well-defined meaning. Moreover, these

[1]On leave from the University of Torino, Italy.
[2]On leave from the University of Ottawa, Canada.

methods remain unsatisfactory for coping with context-dependency, handling exceptional cases, or for capturing increases in knowledge about the concept properties.

When one looks at human concepts, one can see that most of them inherently lack precisely defined boundaries, and that their meaning is context-dependent. Although on the surface these properties can be viewed as undesirable, one can argue that they contribute to a cognitive economy of human knowledge representations (Michalski, 1987). In contrast to fuzzy set theory, our view is that this imprecision and context dependency can be more adequately captured by rules of inference and flexible concept matching than by a numerical set membership function. In other words, the imprecision and context-dependency has often a logical, rather than a probabilistic character. That is, to decide about the concept membership of an instance that is uncommon, borderline, or irregular in some sense, one usually reasons using background knowledge, draws an analogy or perform induction, rather than performs 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. The typicality can be characterized as the degree to which an instance shares the common concept properties. In addition, in different contexts, the same concept may have a vastly different meaning. For example, 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 their boundaries have certain degree of fluidity, and can change with the context in which the concepts are used. It is clear that in order to learn such concepts, machine learning systems need to employ richer concept representations than are currently used. In view of the ubiquity of flexible concepts, developing adequate methods for learning them is clearly one of the fundamental tasks for machine learning research.

Our approach to learning such concepts is based on the idea of *two-tiered 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 basic properties of the concept, while the ICI describes implicitly, through rules and matching procedures, the allowed modifications of the explicit meaning and its extensions in different contexts.

In the general formulation of the two-tiered representation, the "distribution" of the meaning between the two tiers is not fixed, but depends on the properties of the reasoning agent, and the criteria for evaluating the quality of concept descriptions. For example, in the method described here, the first step of a leaning process produces a concept description in which the BCR is a complete and consistent characterization of all training examples. Such a description, as shown experimentally, can be overly complex and perform poorly on new examples. Therefore, the second step optimizes this description according to a criterion measuring its "quality." Our experiments have shown that this optimized description, in addition to being substantially simpler, may also perform better in recognizing new concept examples, than the original complete and consistent description. In the application of the two-tiered method to modelling human concept representation, the Base Concept Representation would describe the most typical, common, and intentional meaning of a concept, while the Inferential Concept Interpretation would handle 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; Michalski, 1988; and 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 the description's complexity was substantially reduced without affecting its performance on new examples. The effect was obtained by removing those parts of the complete and consistent description that covered only a small fraction of examples (the so called *light disjuncts*, or *light rules*), and by applying a *flexible matching* procedure for concept recognition.

This paper extends and continues 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 to derive 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 considers the accuracy of the description, the computational cost of both tiers—Base Concept Representation and Inferential Concept Interpretation, and its cognitive comprehensibility (Bergadano, et al., 1988). By introducing such a general description quality measure one can view any form of concept learning as a process of modifying the input concept description in order to maximize a given description quality measure. In this characterization, the initial concept description may be in the form of positive examples, positive and negative examples, a complete and/or consistent concept description, a tentative description (e.g., supplied by a teacher), an abstract concept definition (as in the explanation-based learning), or a combination of these forms.

Another difference between the present approach and previous research is that flexible matching is used not only in the recognition process, as in (Michalski, et al., 1986), but also in the learning process, i.e., in searching for high "quality" concept descriptions. This feature also distinguishes the method from the related work described in (Bergadano & 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 that learning is more effective if one uses the same instruments for learning and for the performance phases was also present in some incremental learning systems (e.g., Fisher 1987). The work described here represents also an important advance over tree-pruning techniques (e.g., Quinlan, 1987) which apply much more restrictive description reduction operators and do not use any form of deductive matching or flexible interpretation of the learned descriptions. Other advances include the ability to take into consideration the typicality of training instances (when it is known), and the introduction of a rule base in the Inferential Concept Interpretation.

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 complete and consistent descriptions, and the earlier method using two-tiered representation based on the TRUNC procedure.

The method proposed has been implemented in the POSEIDON[1] 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 two-tiered representation can substantially reduce the concept representation, and at the same time improve its predictive power. They also show 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

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 boundary (e.g., Collins & Quillian, 1972; Minsky, 1975; Smith & Medin, 1981; Sowa, 1984). In domains with a significant amount of noise, it may be advantageous that the concept representation is partially inconsistent and/or incomplete with regard to the given instances of the concept. The latter has been demonstrated by the work on pruning decision trees (e.g., Quinlan, 1987), and the HILLARY system (Iba, et al., 1988), and the work on two-tiered representation (Michalski, 1987; and this paper).

In traditional approaches, the recognition of a concept instance is typically done by directly matching the instance description 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 a semantic network, 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 inference mechanisms to classify new instances. In these methods, however, the concept representation consists of stored individual examples (cases). Such a representation taxes memory, and makes matching concepts with instances more complex. The same objection applies to methods of exemplar-based learning that use slightly generalized cases, because the number of cases may still be large.

In contrast to the above, the two-tiered representation has the capability of employing a general concept description (BCR), and an inference mechanism (ICI) to match the description with instances. Thus, the concept representation can be simpler than the one that stores individual examples, or their independent generalizations. The BCR can be viewed as a characterization of the central tendency, the most relevant properties, and the basic intention behind the concept.

The Inferential Concept Interpretation handles special cases, exceptions[2] and context-dependency. 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 rules contained in the Inferential Concept Interpretation. Inference allows the recognition, extension or modification of the concept meaning according to its context.

When an unknown entity is to be recognized, it is first matched against the Base Concept Representation. Then, depending on the outcome, the entity may be related to the concept's inferential extensions or contractions. A simple inferential matching can be merely

a probabilistic inference based on some measure of similarity, e.g., the *flexible matching* method (Michalski et al., 1986). Advanced matching may involve any kind of inference—deductive, analogical or inductive.

Let us illustrate the idea of two-tiered representation using the concept of "chair." A simple two-tiered representation of that concept is given below (for an example of a two-tiered chair description actually learned, see Bergadano, et al., 1988a).

BCR: *Superclass*: A piece of furniture.

Function: To seat one person.

Structure: A seat supported by legs and a backrest attached from the side.

Physical properties: The number of legs is usually four. Often made of wood. The height of the seat is usually about 14–18 inches from the end of the legs, etc.

(BCR may also include a picture or a 3D model of typical chairs)

ICI: *Possible variations of the properties in BCR*: The number of legs can vary from one to four. The legs may be replaced by any support. The shape of the seat, the legs and the backrest, and the material of which they are made are irrelevant, as long as the function is preserved. The backrest may be very small or missing, etc.

Variations:

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 dependency:

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 stated in BCR. The chair does not serve for seating persons, but correspondingly small dolls.

This simple example illustrates several important features of two-tiered representation. Commonly occurring cases of chairs directly satisfy the BCR, and the ICI is not involved. The recognition time can thus be reduced for common cases. The BCR is not the same as a description of a prototype (e.g., Rosch & Mervis, 1975), as it can be a generalization characterizing different typical cases or be a set of different prototypes. The ICI does not represent only distortions or corruptions of the prototype, but it can describe some radically different cases. When an entity does not satisfy the base representation of any relevant

concept (which concepts are relevant is indicated by the context of discourse), or satisfies the base representation of more than one concept, the ICI is involved. The ICI can be changed, upgraded or extended, without any change to Base Concept Representation. While the BCR-based recognition involves just direct mathing, the ICI-based recognition can, in general, involve a variety of transformations and any type of inference.

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

As mentioned before, in the general two-tiered method, the distribution of the concept meaning between the Base Concept Representation and Inferential Concept Interpretation can vary, depending on the criterion of the concept description quality. For example, the Base Concept Representation can be just concept examples, and the Inferential Concept Interpretation can be a procedure for inferential matching as used in the cased-based reasoning approach. Consequently, the case-based reasoning approach can be viewed conceptually as a special case of the general two-tiered representation method.

2.1. Concept representation language

In the proposed method, the formalism used for concept representation is based on the *variable-valued logic system* VL_1 (Michalski, 1975). This formalism allows to express simply and compactly any function that maps a set of vectors into a discrete set. Its advantage is that it provides a formally well-defined, and at the same time, both comprehensible and powerful mechanism for expressing complex logical relationships. The basic component of the representation is a VL_1 *elementary condition* (formally called a *selector*), which is a relational expression:

[L # R]

where L is an attribute, R, called the *referent*, is a value or a disjunction of values from the domain of L, and # is one of the relational symbols $=, <, >, \leq, \geq, \neq$. Such a condition is satisfied by an example, if the value of attribute L in this example is in relation # to R. For example, [blood type = A v O] expresses the statement that blood type is A or O. A conjunction of such elementary conditions is a VL_1 conjunctive statement (formally called a *complex*). For example, the expression [shape = circle v square] & [length > 2] & [color $\neq$ red] is a VL_1 conjunctive statement or a complex. The last elementary condition in this statement is satisfied by an example, if the attribute "color" in this example takes the value that is not "red."

A disjunction of VL_1 conjunctive statements is a *VL_1 DNF expression*. An expression in which one VL_1 conjunctive statement implies another is called a *VL_1 rule*. If a VL_1 conjunctive statement, S, is a description of examples of class C, then this fact is equivalent to a rule S $\rightarrow$ [class = C]. For simplicity, from now on, a VL_1 elementary condition is called a *condition*, a VL_1 conjunctive statement with an implied class is called a *rule*, and a VL_1 DNF description of a class is called a *ruleset* for that class. Both, the base representation, as well as the rules in the inferential interpretation of a concept are represented as (VL_1) rulesets.

2.2. *Inferential Concept Interpretation: Flexible matching function*

A part of the Inferential Concept Interpretation is *flexible matching function*, F, which is assumed to be given as the background knowledge of the learner. The function measures the degree of match between an event (example) and a concept description. In the method implemented, F 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 for an event e, and a concept description D, is defined as the probabilistic sum of F for its rules. Thus, if D consists of two rules, r_1 and r_2, we have:

$$F(e,D) = F(e, r_1) + F(e, r_2) - F(e, r_1) \times F(e, r_2)$$

A weakness of the probabilistic sum is that it is biased toward descriptions with many rules. If a concept description D has a large number of rules, the value of $F(e, D)$ may be close to 1, even if $F(e, r)$ for each rule r, is relatively small (see Table 3 in Section 6). To avoid this effect, if the value of $F(e, r)$ is below a certain threshold, then it is assumed to be 0. In POSEIDON, this problem does not occur, because concept descriptions are typically reduced to only few rules (see the TRUNC-SG procedure in Section 3).

The degree of match, $F(e, r)$ between an event e, and a rule r, is defined as the average of the degrees of fit for its constituent conditions, weighted by the proportion of positive examples to all examples covered by the rule:

$$F(e, r) = \left(\sum_i F(e, c_i)/n \right) \times \#rpos/(\#rpos + \#rneg)$$

where $F(e, c_i)$ is a degree of match between the event e and the condition c_i in the rule r, n is the number of conditions in r, and *#rpos* and *#rneg* are the number of positive examples and the number of negative examples covered by r, respectively.

The degree of match between an event and a condition depends on the type of the attribute in the condition. An attribute can be one of four types: nominal, structured-nominal, linear and structured-linear (Michalski & Stepp, 1983). Values of a structured-nominal

(linear) attribute are nodes of an unordered (ordered) generalization hierarchy. In an ordered hierarchy, the children nodes stemming from any parent node constitute a totally ordered set.

In a nominal or structured-nominal condition, 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 condition is satisfied by an event, and 0 otherwise. In a linear or structured-linear condition, the referent is a range of values, or an internal disjunction of ranges, e.g., [weight = 1..3 v 6..9]. A satisfied condition returns the value of match 1. If the condition is not satisfied, the degree of match is a decreasing function of the distance between the value and the nearest end-point of the interval. If the maximum degree of match between an example and all the candidate concepts is smaller than a preset threshold, the result is "no match."

2.3. Inferential Concept Interpretation: Deductive rules

In addition to flexible matching, the Inferential Concept Interpretation includes a set of deductive rules that allow the system to recognize exceptions and context-dependent cases. For example, flexible matching allows an agent 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 take into consideration the context-dependency.

The deductive inference rules in the Inferential Concept Interpretation are expressed as Horn clauses. The inference process is implemented using the LOGLISP system (Robinson & Sibert, 1982). Numerical quantifiers and internal connectives are also allowed. They are represented in the annotated predicate calculus (Michalski, 1983).

2.4. Types of match

The method recognizes three types of match between an event and a two-tiered description:

1. *Strict match*: an event matches the Base Concept Representation exactly. In this case, the event is said to be S-covered.
2. *Flexible match*: an event is not S-covered, but matches the Base Concept Representation through a flexible matching function. In this case, the event is said to be F-covered.
3. *Deductive match*: the event is not F-covered, but it matches the concept by conducting a deductive inference using the Inferential Concept Interpretation rules. In this case, the event is said to be D-covered. (In general, this category could be extended to include also matching by analogical reasoning and induction; Michalski, 1989).

The above concepts provide a basis for proposing a precise definition of classes of concept examples that are usually characterized only informally. Specifically, examples that

are S-covered are called *representative* examples; examples that are F-covered are called *nearly-representative* examples; and examples that are D-covered are called *exceptions*. As mentioned earlier, one of the major advances of the presented method over previous methods using two-tiered representation (e.g., Michalski , et al., 1986) is that the Inferential Concept Interpretation includes not only a flexible matching procedure, but also inference rules. Thus, using our newly introduced terminology, we can say that the method can handle not only representative or nearly representative examples, but also exceptions.

3. An overview of POSEIDON

The ideas presented above have been implemented in a system for learning two-tiered descriptions, called POSEIDON. Table 1 summarizes the two basic phases in which the system learns the Base Concept Representation. The first phase generates a general consistent and complete concept description, and the second phase optimizes the description according to a given measure of description quality.

3.1. Basic algorithm

The complete and consistent description is determined by the AQ15 inductive learning program (Michalski, et al., 1986). The second phase improves this description by conducting a "double level" best-first search. This search is implemented by the TRUNC-SG procedure (SG symbolizes that the method uses both specialization and generalization operators). In this "double level" search, the first level is guided by the *description quality measure*, which ranks candidate descriptions (see Section 4). The second level search is guided by heuristics controlling the search operators to be applied to a given description. The search operators simplify the description by removing some of its components, or by modifying the arguments or referents of some of its predicates (see sec. 3.2). A general structure of the system is presented in Figure 1.

Table 1. Basic phases in generating BCR in POSEIDON.

Phase 1
Given:
Concept examples obtained from some source
Relevant background knowledge
Determine:
Complete and consistent description of the concept
Phase 2
Given:
Complete and consistent description of the concept
A general description quality (GDQ) measure
Typicality of examples (if available)
Determine:
The Base Concept Representation that maximizes GDQ.

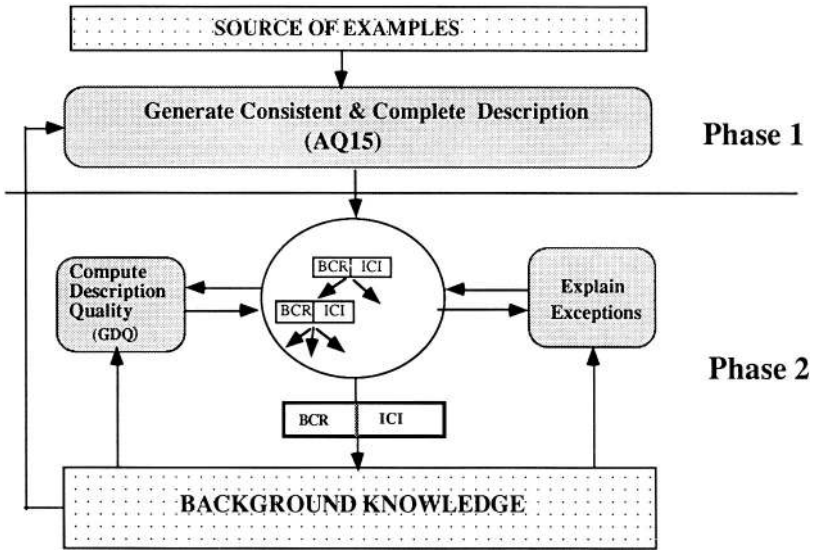

Figure 1. Learning in the POSEIDON system.

The search process is defined by:

Search space: A tree structure, in which nodes are two-tiered concept descriptions (BCR + ICI)
Operators: Condition removal, Rule removal, Referent modification (see Section 3.2).
Goal: To determine a description that maximizes the general description quality criterion.

The goal of the search 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. The nodes to be expanded are suggested by various heuristics discussed in Section 5.1.

The BCR is learned from examples, and represented as a ruleset (as described in Section 2.2). The Inferential Concept Interpretation consists of two parts: a flexible matching function and a rule base. The rule base consists of rules that explain exceptional examples, and is acquired through an interaction with an expert.

3.2. Operators for optimizing Base Concept Representation

A description can be modified using three general operators: rule removal, condition removal and referent modification (i.e., a modification of the subset of attribute values that represents a condition). The rule removal operator removes one or more rules from a ruleset. This is a specialization operator because it leads to "uncovering" some examples. It is the reverse

of the "adding an alternative" generalization rule (Michalski, 1983). Condition removal (from a rule) is a generalization operator, as it is an instance of the "dropping condition" generalization rule.

The referent modification operator changes the referent in a condition. Such changes can either generalize or specialize a description. Consequently, two more specific operators are defined: *condition extension*, which generalizes the description, and *condition contraction*, which specializes the description.

To illustrate these two types of referent modification, consider the condition

[size = 1..5 v 7]

A referent modification that produces a condition

[size = 1..7]

is a condition extension operator. If, however, the initial condition was

[size ≠ 1..5 v 7]

then the same referent modification, i.e., the change to [size ≠ 1..7], would represent a condition contraction operator. A referent modification (in the original condition) that produces a condition

[size = 1..5]

is a condition contraction operator. Similarly, if the initial condition was [size ≠ 1..5 v 7], then the modification to [size ≠ 1..5] would represent a condition extension operator. A summary of the effect of different operators on a description is given in Table 2.

Thus, applying the above search operators can either specialize or generalize the given description. A generalized (specialized) description covers potentially a larger (smaller) number of training examples, which can be positive or negative. At any given search step, the algorithm chooses an operator on the basis of an evaluation of the changes in the coverage caused by applying the operator (Section 5.2).

3.3. Learning the Inferential Concept Interpretation

As indicated above, by applying a search operator (RR, CR, CE or CC) to the current Base Concept Representation, one can make it either more general or more specific. If

Table 2. Search operators and their effect on the description.

Search Operator	Type of Knowledge Modification
Rule removal (RR)	Specialization
Condition removal (CR)	Generalization
Condition extension (CE)	Generalization
Condition contraction (CC)	Specialization

the modified representation is more specific, some positive examples previously covered may cease to be S-covered. These examples may, however, be still covered by the existing Inferential Concept Interpretation (and thus would become F-covered or D-covered). On the other hand, if the modified base representation is more general than the original one, some negative examples, previously uncovered, may now become S-covered. They may, however, remain to be excluded by the existing Inferential Concept Interpretation rules. Consequently, two types of rules in the Inferential Concept Interpretation can be distinguished: those that cover positive examples left uncovered by the base representation ("positive exceptions"), and rules that eliminate negative examples covered by the base representation ("negative exceptions"). A problem then is how to acquire these rules.

The rules can be supplied by an expert, inherited from higher level concepts, or learned from other knowledge. If the rules are supplied by an expert, they may not be operationally effective, but they can be made so through some form of analytic learning (e.g., Mitchell, et al., 1986; Prieditis & Mostow, 1987). If the rules supplied by an expert are too specific or not totally correct, they may be improved inductively (e.g., Michalski & Larson, 1978; Dietterich & Flann 1988; Mooney & Ourston, 1989). Thus, in general, learning the Inferential Concept Interpretation can be accomplished by different strategies, the same as learning the Base Concept Representation.

In the implemented method, the system identifies exceptions (i.e., examples not covered by the Base Concept Representation), and asks an expert for a justification (see sec. 5.2). The expert is required to express this justification in the form of rules. The search procedure, shown in Fig. 1, guides the process by determining examples that require justification. 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 justify why the examples identified by the learning system as exceptions are also members of the concept class.

4. Quality of concept descriptions

Concept descriptions are influenced by different factors and their combinations. Starting from that point, this section derives quality measures for concept descriptions.

4.1. Factors influencing the description quality

The learning method utilizes a general description quality measure that guides the search for an improved two-tiered description. The General Description Quality 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 strongly depend on the particular training set, and, consequently, may perform poorly in classifying future examples. For that reason, when learning from imperfect inputs, it is often better 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 intelligent system is supposed to give 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 too complex syntactically. This requirement is called the *comprehensibility principle* (Michalski, 1983). There is no well established measure of description's comprehensibility, therefore we approximate it by using a measure of a *representational simplicity* of a description. Such a simplicity measure is determined by counting the number of operators of different kinds involved: disjunctions, conjunctions, and the relations embedded in individual conditions. 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 (the *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 computation 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 such an approach is often appropriate, it also has significant disadvantages.

First, it combines 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, it usually forces 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 some important criterion, that it is not worth to even consider the alternatives. To overcome these problems, we use a combination of two measures, a lexicographic evaluation and a linear function-based evaluation, as described in the next section.

4.2. Combining individual factors into a single preference criterion

Given a set of candidate descriptions, we use the General Description Quality criterion to select the "best" description. In POSEIDON, the General Description Quality 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, 1983) is:

$$\text{LEF: } \langle(\text{Criterion1}, \tau_1), (\text{Criterion2}, \tau_2), \ldots (\text{Criterionk}, \tau_k)\rangle$$

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 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 choose the description 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, the representational simplicity and the computational complexity, specified in that order. The tolerances are parameters of the program, and are 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, the above approach uses a computationally efficient LEF to reduce the set of candidates to a small set, and then applies a more complex measure to the remaining candidates.

4.3. The role of typicality of examples

Accuracy is a major criterion to determine 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 & 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 measure of the degree of completeness and the degree of consistency of a description that takes into account the typicality of the examples. The typicality of examples can be approximated by the frequency of their occurrence. 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.[3]

In defining the completeness and consistency of a two-tiered description, other factors may be taken into account. 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 Base Concept Representation, and non-typical, or exceptional ones by the Inferential Concept Interpretation. In POSEIDON, the Base Concept Representation is inductively learned from examples provided by a teacher, and it is desirable that they are typical of the concept being learned. The Inferential Concept Interpretation rules are provided by a teacher. 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) examples. 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 defines the General Description Quality (GDQ) measure implemented in POSEIDON. To this end, we first define the *typicality-based completeness*, T__COMPLETENESS, and the *typicality-based consistency*, T__CONSISTENCY, of a two-tiered concept description:

$$\text{T\_\_COMPLETENESS} = \frac{\underset{e^+ \in \text{S-cov}}{\Sigma w_s * \text{Typicality}(e^+)} + \underset{3^+ \in \text{F-cov}}{\Sigma w_f * \text{Typicality}(e^+)} + \underset{e^+ \in \text{D-cov}}{\Sigma w_d * \text{Typicality}(e^+)}}{\underset{e \in \text{POS}}{\Sigma\ \text{Typicality}(e)}}$$

$$\text{T\_\_CONSISTENCY} = 1 - \frac{\underset{e^- \in \text{S-cov}}{\Sigma w_s * \text{Typicality}(e^-)} + \underset{3^- \in \text{F-cov}}{\Sigma w_f * \text{Typicality}(e^-)} + \underset{e^- \in \text{D-cov}}{\Sigma w_d * \text{Typicality}(e^-)}}{\underset{e \in \text{NEG}}{\Sigma\ \text{Typicality}(e)}}$$

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 representative 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.

Using the terms of T_COMPLETENESS and T_CONSISTENCY, the description *accuracy* is defined as:

$$\text{ACCURACY} = w_1 * \text{T\_COMPLETENESS} + w_2 * \text{T\_CONSISTENCY}$$

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. As mentioned earlier, we approximate it by a representational simplicity, which is a reversal function of representational complexity, defined as:

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

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 values of complexities 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 Base Concept Representation is supposed to describe the general and easy-to-define meaning of the concept, while the Inferential Concept Interpretation is mainly used to handle rare or exceptional events. As a consequence, the Base Concept Representation should be easier to comprehend than the Inferential Concept Interpretation, and thus v_1 should be larger than v_2. The computational complexity (or a 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 cost of evaluating the concept description.

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

$$\text{ED(description)} = \sum_{e \in Pos+Neg} ec(e)/(/Pos/ + /Neg/)$$

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:

$$\text{Cost(description)} = u_1 * \text{MC(description)} + u_2 * \text{EC(description)}$$

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

The definitions of the above measures together with the specification of the way they can be combined (sec. 4.2) define the general description quality. Various weights 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 base representation of a concept is performed in two phases. In the first phase, a complete and consistent concept description is generated by inductive learning from examples. In the second phase, the obtained complete and consistent description is optimized according to the general description quality criterion. In our approach, the first phase is done using the AQ15 learning program (Michalski, et al., 1986a). This section describes the second phase.

5.1. Search heuristics for optimizing Base Concept Representation

Optimizing BCR by a direct application of the General Description Quality measure is computationally expensive, because every newly generated description has to be matched flexibly against the complete set of training examples. To make the process more efficient, we have introduced a *double-level* search method. The first level uses a simple heuristic to determine which operator (RR, CR, CE or CC) is likely to improve the description, and the second level actually applies the operator, and evaluates the description using General Description Quality.

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

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

where ΔP (ΔN) is the *change* in the number of positive (negative) examples that would be covered by the description after applying the operator, and TP (TN) is the total number of positive (negative) examples. For generalizing operators, SR and CE, ΔP and ΔN are non-negative, and for specializing operators, CR and CC, ΔP and ΔN are non-positive.

The advantage of the Potential Accuracy Improvement measure is that it can be computed much more efficiently than the General Description Quality. For every condition in the current description, a list of examples covered by it is maintained using bit vectors. The sets of examples covered by a ruleset (representing a complete description) is then obtained by intersection and union operations. The matching time can be improved further by also maintaining bit vectors for the examples covered by rules (the matching time trades off with the memory for storing the bit vectors). Note that computing the General Description Quality requires flexible matching, and thus cannot be done by an intersection and union operations on bit vectors.

The above formula does not take into consideration the degree of reduction of the description complexity caused by applying an operator. For example, removing a rule reduces complexity more than removing a condition. To account for this, POSEIDON assigns a higher weight (preference) to applying the RR operator (rule removal) than for applying the CR operator (condition removal).

The condition removal operator generalizes the description, therefore, the description (ruleset) resulting from its application may cover some additional examples (positive or negative). Due to this, some rule(s) may become redundant. If the CR operation produces a rule that differs from another rule only in the value of attribute, the two rules may be merged into one, in which the attribute is related to the internal disjunction of values (this is a case of the so-called "refunion" operation; see Michalski & Stepp, 1983). For example, the rules [shape = circle]&[size = 2..6] and [shape = square]&[size = 2..6] can be replaced by single rule [shape = circle v square]&[size = 2..6].

It is worth noting that in the case of operators RR and CR, the Potential Accuracy Improvement 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* (rule or condition) to be removed. Such a heuristic is very efficient because it needs to be computed only once for every condition and every rule 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 Potential Accuracy Improvement is chosen, and applied to the description under consideration. The descriptions so generated are then subjected to an evaluation by the General Description Quality criterion.

5.2. Search algorithm

The search process is conducted according to the algorithm in Table 3:

Table 3. The algorithm for optimizing a concept description.

1. Identify in the search tree the best candidate description D

   (Initially, D is the complete and consistent description obtained by the AQ15 program in Phase 1, and then it is the highest rank description according to the General Description Quality criterion).

2. Apply to D the operator, from among the operators:

   RR_i: Remove the i-th rule from D.
   CR_{ij}: Remove the j-th condition from the i-th rule in D.
   CE_{ij}: Extend the referent of the j-th condition in the i-th rule in D.
   CC_{ij}: Contract the referent of the j-th condition in the i-th rule in D,

   that maximizes the Potential Accuracy Improvement measure.

3. Compute the General Description Quality (*GDQ*) of the description obtained in step 2. If the GDQ is smaller than the GDQ of original D, then proceed to step 1. (When computing the description accuracy for GDQ, flexible matching is used).

4. Ask an expert for an explanation of the exceptional examples, which are

   (a) the positive examples that cease to be covered, and
   (b) the negative examples that become covered.

   If an explanation is given, add the rules that make up the explanation to the Inferential Concept Interpretation, otherwise add to it the exceptional example(s).

5. Update the GDQ value of the new node by taking into account the added Inferential Concept Interpretation.

6. If the *stopping criterion* is satisfied, then STOP, otherwise proceed to step 1.

Let us explain the motivation and individual steps of the algorithm. Step 1 chooses the node (description) for expansion on the best-first basis, that is, chooses the node with the highest General Description Quality. (This is not always an optimal choice, because "worse" 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 General Description Quality as the measure of concept quality).

Step 2 chooses the "best" search operator according to the Potential Accuracy Improvement heuristic, and applies it to the current description. Step 3 computes the General Description Quality of the new node. It should be noted that, in the General Description Quality measure, the typical examples covered directly by the base representation can weigh more than those covered through flexible matching. The examples covered by Inferential Concept Interpretation rules weigh more than the ones covered through flexible matching, but less than the ones covered by the Base Concept Representation.

Step 4 determines exceptional examples, and asks an expert for an explanation of them. If the explanation is provided, appropriate rules are added to the Inferential Concept Interpretation. These rules may extend or contract the Base Concept Representation. For example, the rule removal operator might uncover some positive examples, that were previously covered. In this case, new rules added to the Inferential Concept Interpretation would allow

the system to reason about such "special" positive examples, and explain why they should be classified as instances of the concept being learned. On the other hand, the condition removal operator might cause some negative examples to be covered. In this case, new Inferential Concept Interpretation rules would have to be added to contract the Base Concept Representation.

An important issue concerning step 4 is when an explanation should be required from an expert ("explainer"). The problem is that in some cases the chosen operator may not be appropriate, because it leads to a very poor description. In such a case, it is not worthwhile to ask an expert for an explanation, and search should continue in other direction. The method follows the following strategy. Suppose that N is the node (description) being expanded, and M is the node obtained after applying an operator (e.g., the condition removal). The effort to obtain an explanation is made only if the General Description Quality of M is "significantly" better than that of N (above a certain threshold). In this case, the explainer is given the General Description Quality evaluations of both descriptions, N and M, and asked for an explanation. These evaluations give the explainer a sense of importance of the request. If the explainer cannot provide an explanation, the exceptional examples are directly added to the Inferential Concept Interpretation. Step 5 updates the General Description Quality of the obtained two-tiered description by taking into consideration the added Inferential Concept Interpretation rules. Step 6 decides whether to stop or continue the search. The *stopping criterion* is satisfied when the number of nodes explored exceeds value*k1*, or when the General Description Quality is not improved after the exploration of *k2* nodes since the last improvement. The search parameters *k1* and *k2* have a default value, which is modifiable by the user. When the search stops, the best node found until this point defines the chosen two-tiered concept description.

In conclusion, let's summarize the main difference between the above two-level search and the standard best-first search. The difference is that only one operator is applied to the (best-GDQ) node selected for expansion, rather than all available operators, as in the standard search. The operator applied is the "best" according to the PAI heuristic. Such a procedure helps to avoid generating low quality nodes, and thus makes unnecessary the computation of the General Description Quality for these nodes. Other operators are applied only if the results obtained along this branch of the search tree turn out to be unsatisfactory.

5.3. An abstract example

An abstract example of the search process is given in Figure 2. Individual nodes represent both components of a two-tiered description (Base Concept Representation and Inferential Concept Interpretation) generated at any given search step, and show the coverage of training examples by the description. The rectangular areas represent the coverage by the Base Concept Representation, and the curved lines denote the coverage by the Inferential Concept Interpretation.

In the example, the accuracy is computed according to the formula presented in Section 4, assuming that all examples have the same typicality. The initial description is represented by node 1. The BCR contains two rules represented by two rectangular areas, which cover five positive examples out of eight, and one negative example out of five. The Inferential

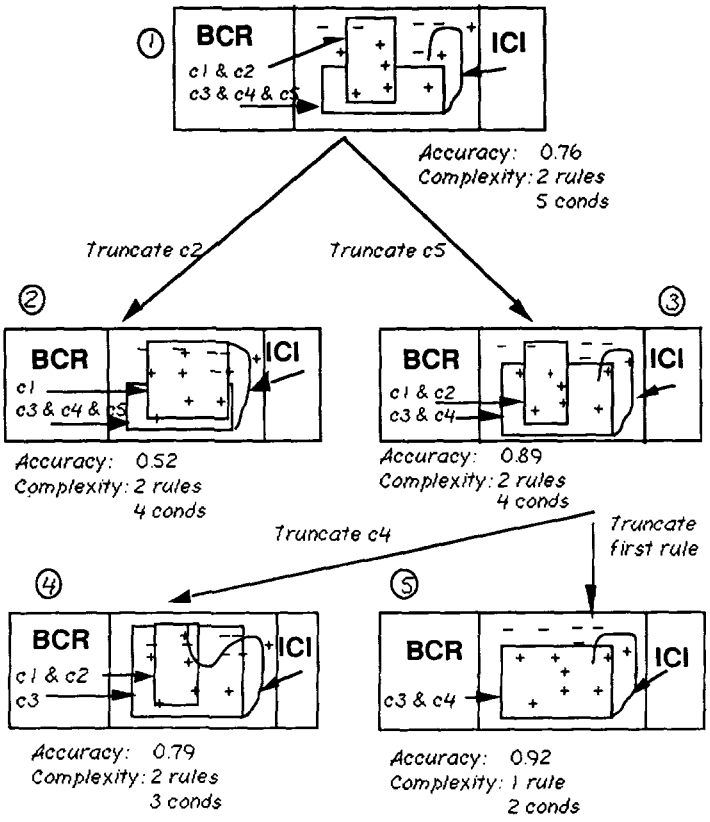

Figure 2. An illustration of the search process.

Concept Interpretation extends this coverage by recognizing one more positive example. Next nodes correspond to descriptions obtained by an application of operators marking the branches of the search tree. For example, node 3 is obtained by eliminating condition C5 in the second rule of the initial description. The new description is more accurate because all positive examples are now covered, without changing the coverage of the negative examples.

By truncating the first rule in node 3, node 5 is generated. The description no longer covers negative examples, and is simpler. This node is then accepted as the optimized description resulting from the search. The other nodes lead to inferior concept representations with respect to General Description Quality, and are discarded. The quality has been computed with $w_1 = w_2 = 0.5$. For simplicity, the cost is omitted, and the complexity of the Inferential Concept Interpretation is ignored. The complexity of the Base Concept Representation is indicated by the number of rules and the number of conditions.

6. Experiments

Experiments tested the POSEIDON program, and compared its performance with that of three other methods: variants of exemplar-based learning, the method for learning consistent

and complete descriptions (as implemented in AQ15), the method for generating *top rule* descriptions (as described by Michalski et al, 1986), and the method for generating pruned decision trees (as implemented in the ASSISTANT program; Cestnik, Kononenko, & Bratko, 1987). All these methods were applied to the same data from two problem domains. The learned descriptions were then tested on the same testing examples.

The first domain concerned labor-management contracts, and the problem was to learn a general description that discriminates between acceptable and unacceptable contracts. The second domain concerned congressional voting records, and the problem was to characterize the voting behavior of Republicans and Democrats in the U.S. House of Representatives. To describe the experiments, we start with a brief characterization of the data used.

6.1. Experimental data

Labor-management contracts

The data regarding labor-management contracts were obtained from *Collective Bargaining*, a review of current collective bargaining issues published by the Department of Labor of the Government of Canada. The data describe labor-management contracts that were negotiated between various organizations and labor unions with at least 500 members.

The raw data covered several economic sectors. Because the attributes describing individual contracts varied among different economic sectors, the experiments focused on only one sector: personal and business services. This sector includes unions representing hospital staff, teachers, university professors, social workers and certain classes of administrative personnel. The data involved multivalued attributes, and therefore the VL_1 language was very suitable and directly applicable to these data.

The data used in the experiments describe contracts concluded in the second half of 1987 or the first half of 1988. Each contract is described by sixteen attributes, belonging to two main categories. One category concerns issues related to the salaries, e.g., pay increases in each year of the 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.

Below is a typical example of an 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 = better than average in the industry
Long term disability insurance = offered by the employer
50% dental insurance cost = covered by the employer
Bereavement leave = available
Employer-sponsored health plan = not mentioned

The above description was represented as the following VL_1 rule:

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

::> [contract = acceptable]

(In the above rule, for simplicity, the conjunction is represented by concatenation).

U.S. Congress voting record

The data regarding the U.S. Congress voting record were the same as the ones used by Lebowitz (1987) in his experiments on conceptual clustering. The data represent the 1981 voting records of 100 selected representatives (50 in the training set and 50 in the testing set). The problem was to learn descriptions discriminating between the voting record of Democrats and Republicans. Below is an example of the voting record of a Democrat in the U.S. Congress:

Draft registration = no
Ban 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 A description of experiments

For each problem domain, the experiments involved the following steps:

1. Learning a complete and consistent description from the training examples (by the AQ15 program).
2. Determining the *top rule* description from the above description using the TRUNC method (Michalski, et al., 1986).
   Such a description consists of a single rule that covers the maximum number of positive examples among all other rules in the complete and consistent description. Such a description is easy to determine, because the AQ15 generates rules together with measures indicating the number of examples covered *totally* and *uniquely* by each rule (i.e., the t-weight and u-weight, respectively; see below). In the experiments, one top rule description was generated for positive concept examples, and one for the negative examples (i.e., from a complete and consistent description of the negative examples). An instance was classified as belonging to a 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, then the instance was classified as "no match." Learning the *top rule* description, and using it with flexible matching, represents a simple, but important version of the two-tiered concept learning approach (Michalski, 1990).
3. Determining an optimized two-tiered description from the complete and consistent description using the TRUNC-SG procedure.
4. Determining descriptions of the given concepts using other methods, specifically, variants of the exemplar-based learning approach, and the decision tree learning algorithm ASSISTANT.
5. Testing the performance of all generated descriptions on the testing examples.

To illustrate the difference between the complete and consistent descriptions, the *top rule*, and the optimized descriptions created by POSEIDON, figures below show a sample of these descriptions in the labor management domain. 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.

By selecting from each of the above descriptions the rule with the largest t-weight, *top rule descriptions* were obtained (Figure 4).

By applying the method implemented in POSEIDON (the TRUNC-SG optimization of the complete and consistent description, and the ICI rule acquisition), the following optimized two-tiered description was obtained (Figure 5).

During the BCR description optimization process, the system determined the training events that were incorrectly classified by the base representation. An expert was asked to formulate rules explaining these examples (the ICI rules in Figure 5). For example, the first ICI rule for an unacceptable contract (Figure 5) describes contracts with the wage increase in the first year lower than 3%, and an even lower increase in the second year. In such circumstances, the holiday and vacation time do not matter, and the contract is classified as unacceptable (by the union).

```
[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%] & [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%] & [vacation > average]:                       (t = 1, u = 1)

    ::>  [Contract =  acceptable]

[wage_incr_yr1 = 2..4%]&[#holidays < 10]&[vacation ≤ 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)

    ::>  [Contract =  unacceptable]
```

Figure 3. The complete and consistent descriptions generated by AQ15.

BCR:

```
[contract-duration >1]&[wage_incr_yr2 >3%]&[#holidays >10]:        (t = 11, u = 11)
    ::>  [Contract =  acceptable]

[wage_incr_yr1 = 2..4%]&[#holidays < 10]&[vacation ≤ average]:     (t = 3, u = 3)
    ::>  [Contract =  unacceptable]
```

ICI: Flexible matching

Figure 4. The *Top rule descriptions* generated by the TRUNC method.

As one can see, the optimized BCR descriptions are significantly simpler than the complete and consistent descriptions generated by AQ15. They also seem to represent the most important characteristics of the labor management contracts. Specifically, a contract is unacceptable when it offers a significant wage increase (the first two rules in Fig. 5), or it offers many holiday days, or the vacation time is above average.

6.3. Results from testing POSEIDON and other methods

As mentioned earlier, experiments tested POSEIDON and three other methods, specifically, variants of exemplar-based learning, the method for learning consistent and complete descriptions, a method for generating *top rule* descriptions, and a method for generating pruned decision trees. All of these methods were employed to learn a concept description from

```
BCR:
  [wage_incr_yr1 > 4.5%] v
  [wage_incr_yr2 > 3.0%] v
  [#holidays > 9] v
  [vacation period = above_average]
       ::>  [Contract =  acceptable]

  [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%]
       ::>  [Contract =  unacceptable]
ICI:
  Flexible matching
  Deductive matching using rules:
  [wage_incr_yr1≥5.5%]&[vacation < average]
       ::>  [Contract =  acceptable]
  [wage_incr_yr1≤3%]&[wage_incr_yr2 < wage_incr_yr1]
       ::>  [Contract =  unacceptable]
   [wage_incr_yr1≤3%]&[wage_incr_yr2≤3%]&[hours_work≥40]&[pension= empl_contr]
       ::>  [Contract =  unacceptable]
```

Figure 5. Optimized two-tiered descriptions obtained by POSEIDON.

the same set of training examples. All the learned descriptions were then applied to the same testing examples. The performance was evaluated by counting the number of examples that were classified correctly, incorrectly, or unclassified.

Tables 4 to 7 present the results of different experiments. Table 8 provides a summary of all results. In the tables, columns "Correct" and "Incorrect" specify the percentage of the testing events that were correctly and incorrectly classified, respectively. The column *No_Match* specifies the number unclassified examples (i.e., the examples that did not match any description to a sufficient degree). To provide an estimate of the complexity of descriptions learned, the tables also list the number of conditions and rules in each description. In the case of pruned decision trees, the table lists the number of nodes and leaves (the number of leaves corresponds to the number of rules that can be directly determined from the decision tree).

Experiment 1 (Table 4) tested a *factual* description, and variants of the examplar-based approach (1-, 3- and 5-nearest neighbor match). A factual description is a disjunction of all the training events, and, as such, is obviously complete and consistent with regard to the training set. The first part of Experiment 1 tested the factual description on the testing examples using the *strict match* method. In such a method, a testing example must match exactly one of the training examples to be classified. In this case, obviously, the description had no predictive power. It produced No_Match answers for all testing examples of the labor contract data, and for 96 testing examples of the congressional voting data (two examples were the same in the training and testing sets).

Table 4. Results of experiment 1.

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 Neighbors						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	80%	84%	20%	16%	0%	0%

Subsequent parts of Experiment 1 tested the factual description using the *k-nearest neighbor* method with different k. The method involved determining k closest (best "fitting") learning examples to the one being classified, and assigning to the testing example the class of the majority of the closest examples. Such a method is equivalent to simple forms of exemplar-based learning. The *1-Nearest Neighbor* row in the table lists results from applying the factual description with a matching method somewhat similar to the one described in (Kibler & Aha, 1987). The only difference was that Kibler and Aha's method uses the *maximum* function for evaluating a ruleset (disjunction), while our flexible matching uses the *probabilistic sum* (Section 2.2). We also tested the method with k = 3 and 5.

The second experiment (*Table 5*) used concept descriptions generated by AQ15 without truncation. Such descriptions are consistent and complete 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 complete and consistent descriptions include many specific rules, leaving little space for the "no match" cases (3%), in which flexible matching could help. Secondly, the descriptions consisted only of 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 4*). 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.

Table 5. Results of experiment 2.

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 6. Results of experiment 3.

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%

The third experiment (Table 6) tested the *top rule* descriptions determined from the above complete and consistent descriptions. As shown in Table 6, the performance of these rules using flexible matching was comparable to that of the complete and consistent descriptions, as well as factual descriptions (compare with Tables 4 and 5).

It may be surprising that the top rule descriptions performed better on the testing set than on the training set. This is due to the fact that the training set contained more exceptions than the testing set. The system used the TRUNC method, in which the truncation process removes rules that cover all except the most typical training examples.

The *top rule* descriptions consist of only one rule per concept, and therefore they are significantly simpler than the factual, and consistent and complete descriptions (they use only 2 vs. 11 vs. 27 rules in the Labor Management problem, and 2 vs. 10 vs. 51 rules in the Congress Voting problem). It is quite revealing that such simple rules performed as well as much more complex descriptions generated in previous methods.

The fourth experiment (Table 7) tested optimized descriptions generated by POSEIDON, i.e., derived by the TRUNC-SG method. The descriptions were tested using flexible matching alone (*Flexible Match*), and in the combination with deductive matching (*Deductive Match*).

Table 7. Results of experiment 4.

Optimized (POSEIDON) 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%

For comparison, the performance of these descriptions was also tested using strict match. The latter is rather an impractical combination. As expected, these descriptions used with strict matching gave 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 was slightly lower. The descriptions are simpler than complete and consistent descriptions, although they include the Inferential Concept Interpretation rules. They are, of course, more complex than the top rule descriptions, which do not use any interpretation rules.

For the Labor data, descriptions applied with deductive matching produced higher performance than when used with flexible matching only (90 vs. 83%).[4] 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 8 summarizes the results of experiments, specifically, it compares the performance and complexity of descriptions generated by simple exemplar-based methods, the two-tiered 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 to the same learning and training data, which were used in the previous experiments (whose results were presented in Tables 4–7). 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 factual description 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.

Table 8. Summary of the results of testing descriptions generated by different methods.

	Labor	Congress
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%	87%/14%
Complexity (# Leaves/# Nodes)	29/53	19/28
Complete and consistent description (AQ15 without rule truncation)		
Performance (% Correct/% Incorrect)	80%/17%	87%/14%
Complexity (# Rules/# Conds)	11/29	10/32
Top rule two-tiered description (AQ15 with rule truncation)		
Performance (% Correct/% Incorrect)	83%/17%	85%/15%
Complexity (# Rules/# Conds)	2/6	2/6
Optimized two-tiered description (POSEIDON)		
Performance (% Correct/% Incorrect)	90%/10%	92%/8%
Complexity (# Rules/# Conds)	9/12	10/21

[a]The total performance does not sum up to 100% because of "no match" cases.

In the above experiments, for both domain problems, the learning method implemented in POSEIDON produced descriptions that are simpler (except for the *top rule* descriptions), and also perform better on the testing data than other tested methods. Being simpler, these descriptions are also easier to understand, and have a lower evaluation cost. The meaning of the concept defined by such descriptions depends on the base representation (i.e., a TRUNC-SG optimized description learned from examples), and the inferential concept interpretation (consisting of an apriori defined flexible matching procedure and a set of deductive rules, formulated by the expert).

Using rules in the inferential concept interpretation has an advantage that exceptional cases are easy to explain. In the current method, the system determines which examples are exceptional (those that are misclassified by the base representation). The expert analyzes them, and determines the rules for ICI. The top rule descriptions were significantly simpler than any other descriptions, but performed somewhat worse than the optimized description and the decision tree. Depending on the desired trade-off between the accuracy and simplicity, the *top rule* or the optimized description can be taken as the base representation of the concept being defined.

6.4. The role of parameters and related issues

POSEIDON has many parameters which can be controlled by a user. On the surface, this might be considered as a disadvantage. In our view, a learning system that allows the user to explicitly modify parameters that affect learning processes (but which are not just method-dependent), is to be preferred over a system that does not explicitly define such parameters. The point is that in the latter systems these parameters are defined only *implicitly*, by the assumptions and the structure of the method. For example, many systems do not take into consideration the typicality of examples. In POSEIDON, this is equivalent to an assumption that the typicality of all examples is equal to the default value 1. As another example, consider the cost of measuring the value of attributes. If a learning program does not have parameters representing such costs, then this is equivalent to an assumption that all costs are the same (which in reality is often not true). By being able to control such learning parameters, the user can produce results that better fit the task at hand. For example, for some tasks, the accuracy of descriptions may be a decisive criterion, while for others the description simplicity may be of equal concern.

An important problem to be investigated is the sensitivity of POSEIDON to its various parameters. While a comprehensive answer to this problem goes beyond the scope of this paper, we report below a preliminary sensitivity analysis regarding the parameters controlling the trade-off between the description accuracy and simplicity. Such parameters are considered to have the most important effect on the performance of learned descriptions. Specifically, they are the *tolerances* in the lexicographic evaluation functional measuring the description quality (Sec. 4.2). To explain their role, let us briefly review the description quality measure. This measure combines several criteria, such as the accuracy, the simplicity, and the cost. Each criterion is associated with a tolerance interval such that differences within this interval are not considered unimportant. Thus, if the tolerance interval of accuracy is very narrow, then the accuracy becomes the prevailing criterion in quality evaluation. On the other hand, if this tolerance interval is wide, the remaining criteria become more significant.

An experiment was performed using the same Congress voting data, as used in experiments reported in Tables 4–7. The training set had 51 examples, while the testing set had 49 examples. The concept to be learned was the voting record of Republicans in the U.S. Congress. The description tested in Table 7, had 10 rules and 21 conditions, and yielded the accuracy of 100% on the training set, and 92% on the testing set. The description was obtained using the accuracy tolerance (τ_1) value equal .05. To determine the method's sensitivity to this parameter, the accuracy tolerance τ_1 was set to values .55, .35, .02, .005, and for each value the description accuracy was measured. For the above accuracy tolerances, the system's performance on the testing set was 88%, 88%, 90%, and 92%, respectively. Thus, this experiment seems to indicate that the accuracy of the descriptions slowly grows with the narrowing of the tolerance interval on the accuracy in the description quality measure, which completely confirms an intuitive expectation.

In general, when the accuracy tolerance interval is wide, the simplicity of the description assumes an important role, yielding performances close to the performance of the *top rule* in the two-tiered description. Intermediate values, such as the one used in the experiments presented in Table 8 ($\tau_1 = 0.05$) produced the best results, e.g., the performance of 92%

on the testing set from the Congress data. In the case of the narrow tolerance interval for accuracy, the simplicity has a lower impact on the quality of the description. An interesting topic for future research is to systematically investigate the influence of such parameter changes on the performance of the description.[5]

Another issue that should be explored more in the future is the role of example typicality of learning examples. In the presented method, if the input examples are assigned typicality values, the generated base concept representation will tend to cover the most typical examples, while the inferential concept interpretation will tend to cover less typical examples. A problem for future investigation is to determine the effect of the typicality on the overall quality of generated concept descriptions. When the typicality information is unavailable, the system itself will assign examples to different classes of typicality. The examples covered by the base representation are classified as typical, those covered by flexible matching as nearly-typical, and those covered by the deductive rules as non-typical.[6] An interesting experiment would be to compare such classifications with human classifications. Another interesting issue relates to the noise in the data. The preliminary analysis indicates that the proposed method has a significant ability for handling noisy data. Experiments show that noisy examples are usually covered by the "light" rules, i.e., rules that cover few examples. By removing such rules from the description, the effect of noise can be significantly minimized (Zhang & Michalski, 1989). Future research should investigate these aspects of the method in greater detail.

7. Related work

The research presented here relates to various efforts on learning imprecise concepts, in particular, to learning methods generating pruned decision trees (e.g., Quinlan, 1987; Cestnik, et al., 1987; Fisher & 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 classified by following the decision tree from the root to the leaf indicating the class. Because pruned decision trees do not cover some of the training examples, and the recognition process does not use flexible matching, such trees must always produce some error on the training examples. This may not be significantly detrimental to the overall quality of the decision tree, however, as it avoids overfitting.

The two-tiered method avoids overfitting by simplifying original descriptions, yielding base concept representations that, in the formal logical sense, are usually also incomplete and inconsistent. The two-tiered method, however, can compensate for the lack of coverage or for an excessive coverage of the first tier (BCR), by the application of the second tier (ICI). This can be done by flexible matching and/or deductive inference rules. The latter ones are normally unaffected by noise, because they depend on a deeper understanding of the domain. In addition, the presented method takes into consideration the typicality of the examples (if it is available). This feature gives the method an additional help for 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. The final result might possibly be far from optimal. By avoiding

the search, such a procedure should, however, be significantly faster than the one implemented in POSEIDON. If the speed of learning and the simplicity of descriptions are of central importance, then the TRUNC method (that determines the top rule descriptions without search) should be applied rather than TRUNC-SG. 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, and others use the same training set that was used in creating the tree. The simplification phase in POSEIDON can also be done either using the original training set, or using a separate set of examples.

The experiments by Fisher and Schlimmer (1988) on pruning decision trees use a statistical measure to determine the attributes to be pruned. Such measures require a rather large data sample, and thus do not apply well to small training sets. In the two-tiered approach, training events are analyzed logically, rather than statistically, both in the phase creating a complete and consistent description, and in the optimization phase. Consequently, the two-tiered approach seems to be more suited for learning from a relatively small number of examples. An interesting possibility for future research is to integrate a statistical measure, such as used by Fisher and Schlimmer, or other, in the process of rule learning and truncating with large data sets.

The system developed by (Iba, et al., 1988) uses a trade-off measure that is somewhat similar to the general description quality (GDQ) measure proposed in this paper. Our GDQ measure considers more factors. Besides taking into account the typicality of the instances covered by the description, it considers different types of matching between an instance and a description. Moreover, the simplicity measured by GDQ depends not only on the number of rules in the description as in (Iba, et al., 1988), but also on the different syntactic features in the description.

The CN2 inductive algorithm (Clark & 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 perform 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. On the other hand, such an approach has difficulty with identifying irrelevant conditions and redundant rules.

The two-tiered method described here can also be viewed as a kind of constructive induction in the sense of (Michalski, 1983). In fact, the whole learned description may include new terms, absent from the examples used for learning. This behavior is also encountered in several other systems (e.g. Sammut & Banerji, 1986; Drastal, Czako, & Raatz, 1989). However, constructive learning in POSEIDON is due to the second tier based on domain knowledge characterizing non-typical examples. This is different from using domain knowledge to rewrite or augment the whole training set (e.g., Rouveirol, 1991), or to generate new attributes by a data-driven approach (Bloedorn & Michalski, 1991), or a hypothesis-driven approach (Wnek & Michalski, 1991).

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 Inferential Concept Interpretation 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 using some typicality measure of examples has not so far been given much attention in machine learning, although there were attempts in this direction. For example, Michalski and Larson (1978) introduced the idea of "outstanding representatives" of a concept to focus the learning process on the most significant examples. In cognitive science, the concept of typicality of examples has been studied extensively (e.g., Rosch & Mervis, 1975; Smith & Medin, 1981). The concept of two-tiered representation has naturally led us to the proposition of a precise definition of representative, nearly-representative and exceptional examples, namely, as those that are covered by the first tier, the second tier's procedure for flexible matching, and the second tier's inference rules, respectively (see Section 2.4).

To summarize, there are several major differences between the method presented and related research described in the literature. First, the method has the ability to recover from the loss of coverage due to the description truncation by using the second tier. Specifically, the procedures of flexible matching or deductive rules are used to cover examples not covered explicitly. As has been demonstrated experimentally, this ability often leads to a significant reduction of concept descriptions, and at the same time, to an improvement of their predictive power. Second, the description reduction is done by independently performing both generalization and specialization operators. Third, any part of the description may be truncated in the simplification process, not just only specific parts (as, e.g., in decision tree truncation). Fourth, the method is able to take into account the typicality of the examples. Finally, the method uses a general description quality measure, which takes into consideration a number of different aspects of a description.

It may be informative to relate the presented two-tiered approach to some other machine learning approaches in terms of the type of concept representation and the kind of matching applied for classification. Table 9 below makes such a comparison.

8. Summary and open problems

The most significant aspect of the presented method is that it represents concepts in a two-tiered fashion, in contrast to traditional learning methods that represent concepts by a

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

	Simple Induction	Exemplar-Based	Two-Tiered
Representation	General	Specific	General
Matching	Precise	Inferential	Inferential

monolithical structure. In this representation, the first tier, the base concept representation (BCR), captures the explicit and common concept meaning, and the second tier, the inferential concept interpretation (ICI) defines allowable modifications of the base meaning and exceptions. Thus, typical concept instances match the BCR, and thus can be recognized efficiently. Such a two-tiered representation is particularly suitable for learning flexible concepts, i.e., concepts that lack precise definition and are context-dependent.

In the POSEIDON system that implements the method, the BCR is learned in two steps. First, a complete and consistent description is learned by a conventional learning program (AQ15). Next, the description is optimized according to a general description quality measure. This is done by a double-level search process that uses both generalization and specialization operators. The General Description Quality takes into account not only properties of BCR, but also of ICI (by measuring the complexity and accuracy of the total description). The ICI has two components: one specifies a flexible matching function, and the second specifies inference rules for handling exceptions and context-dependency. The ICI rules can be of two types. The rules of the first type extend the meaning of the concept, while the rules of the second type contract this meaning. The first type rules are employed when an instance is neither covered by the BCR (not S-covered), nor by the flexible matching function (not F-covered). The second type of rules are used when an unknown instance covers a base representation of more than one concept, or when concept membership has to be confirmed. In both cases, the rules are used deductively. An advantage of using rules for matching over other matching methods is that they can serve as an explanation why a given instance does or does not belong to the concept.

The experimental results have strongly supported the hypothesis that two-tiered concept descriptions can be simpler and easier to understand than "single-tier" descriptions. Two-tiered descriptions also perform better. For example, the two-tiered descriptions obtained for the acceptable labor managements contracts gave a performance of over 90% correct using 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 the corresponding pruned decision tree performed at 86%, and had 29 leaves (each of which may be viewed as corresponding to one rule). 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. In addition, two-tiered descriptions are relatively easy to understand, and can easily represent an explicit domain knowledge.

The presented method is different in several significant ways from the earlier method of learning two-tiered representations (Michalski, et al., 1986). The flexible matching procedure is used not only in the testing phase, but also in the learning phase. In addition to a flexible matching function, the method employs rules for extending or contracting the concept meaning. The earlier TRUNC method used only one specialization operator (rule removal), while TRUNC-SG employed in POSEIDON uses two generalization and two specialization operators. The price for that is that the new method is significantly more complex.

There are many interesting problems for future research. Some of them were indicated previously, in particular, in section 6.4. Among especially interesting and important problems is how to integrate the description optimization phase with the initial description generation phase (done by AQ). The first step in this direction is currently being investigated.

In it, the two processes share the same heuristics and the same measure of description quality. The next step is to directly generate the target descriptions. Another problem for future research is how to learn second tier rules from examples. In the initial method developed by (Plante & Matwin, 1990), the inferential concept interpretation rules are learned by a chunking process in the situations when multiple explanations of (positive or negative) training events are provided.

Future research should also address the application of constructive induction (Michalski, 1983) in the process of learning flexible concepts. In constructive induction, background knowledge is used to construct new attributes and higher level descriptions. As a result, produced descriptions can capture the salient features of the concept, and can be simpler and more comprehensible. The ideas of constructive induction seem to be very relevant to the method proposed. For example, through constructive induction the system may be able to fold several rules into a single one, or prevent the removal of relevant rules.

The current system does not address the problem 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 reorganize itself. The distribution of knowledge between the Base Concept Representation and the Inferential Concept Interpretation should be determined by the performance of the system on large testing sets. If it turns out, for instance, that some inferential concept interpretation rules are used very often, then they could be compiled into the base representation. Further research is needed on the role and importance of different parameters used in the method, and on the trade-offs that they can control.

This paper has focused on learning attributional descriptions, that is, descriptions that characterize entities by attributes, and thus do not represent their structural properties. An important topic for future research is to develop methods for learning two-tiered 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 complete and consistent description. The basic search procedure would essentially be the same, but would deal with a more complex knowledge representation. Such a representation would allow additional description modification operators. Also, the computation of the general quality of descriptions would require modification, and flexible matching would need to be extended to handle structural concept descriptions.

As practical problems frequently require only attributional descriptions, and the method presented is domain-independent, POSEIDON has the potential to be useful for concept learning and knowledge acquisition in a wide range of applications.

Acknowledgments

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 criticisms. Thanks go also to members of Machine Learning Doctoral Seminar at George Mason University for helpful discussions and suggestions. The authors also thank the reviewers of Machine Learning Journal for very careful and detailed reviews that helped to improve the earlier versions of the paper. Special thanks go to Zbig Koperczak for his 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 and N00014-91-J-1854 administered by the Office of Naval Research, and in part by the Office of Naval Research under grants No. N00014-88-K-0226, No. N00014-88-K-0397 and N00014-91-J-1351. 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.

Notes

1. The system is named after POSEIDON, the Greek god of the sea, which represent fluidity and changing aspects of nature.
2. The term "exceptions" is used here in its colloquial meaning. In section 3.4, the term is given a precise meaning.
3. 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.
4. This difference, for the labor data, is not χ_2 significant. Nevertheless, we think that there are other reasons to prefer deductive matching over flexible matching. Deductive classification is based on rules and knowledge-based inference, and is therefore easier to understand by humans. The rules may be modified locally, while changing the flexible matching function is difficult and produces uncontrolled, global consequences. In other words, examples that are correctly recognized through ICI deductive rules are also explained *ipso facto* in terms of domain knowledge. The same cannot be said of examples correctly recognized by flexible matching, which is a knowledge-independent distance measure. To reflect this, the GDQ measure assigns a higher score to a description with deductive matching than with flexible matching.
5. In our experiment, for small values of $\tau_1 = 0.02$ and .005, which emphasize the role of accuracy in the measured quality of a description, the performance on the testing set was close or equal to the performance obtained for $\tau_1 = 0.05$, and higher than performance of 86% for AQ15 in Table 8. The reason is that in the last two experiments, as well as in the original experiment in Table 8, it was always possible to find a description that was simpler than the one produced by AQ15, but still 100% correct on the training data. Therefore, by giving more importance to accuracy, the simpler description was preferred, and better performance on the test set was obtained.
6. This three-way classification of the examples is, in fact, a simple method of learning typicality. A similar feature is available in Cobweb (Fisher, 1987). On the other hand, if the typicality information is available, it is used by POSEIDON to improve the quality of the learned description.

References

Bareiss, R. (1989). *An exemplar-based knowledge acquisition*. Academic Press.

Bergadano, F., & Giordana, A. (1989). Pattern classification: An approximate reasoning framework. *International Journal of Intelligent Systems*.

Bergadano, F., Matwin, S., Michalski, R.S., & Zhang, J. (1988a). *Learning flexible concept descriptions using a two-tiered knowledge representation: Part 1—ideas and a method* Reports of Machine Learning and Inference Laboratory. MLI-88-4. Center for Artificial Intelligence, George Mason University.

Bergadano, F., Matwin, S., Michalski, R.S., & Zhang, J. (1988b). Measuring quality of concept descriptions. *Proceedings of the Third European Working Sessions on Learning* (pp. 1-14). Glasgow.

Bloedorn, E. & Michalski, R.S. (1991). Data-driven constructive induction in AQ17: A method and experiments (*Reports of Machine Learning and Inference Laboratory*). Center for Artificial Intelligence, George Mason University.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AutoClass: A Bayesian classification system. *Proceedings of the Fifth International Conference On Machine Learning* (pp. 54-64). Ann Arbor, MI.

Cestnik, B., Kononenko, I., & Bratko, I. (1987). ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. *Proceedings of the Second European Workshop on Learning*. (pp. 31-45).

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3, 261-283.

Collins, A.M., & Quillian, M.R. (1972). Experiments on semantic memory and language comprehension. In L.W. Gregg (Ed), *Cognition, learning and memory*. John Wiley.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1.

Dietterich, T. (1986). Learning at the knowledge level. *Machine Learning*, *1*, 287-315.

Dietterich., T., & Flann, N. (1988). An inductive approach to solving the imperfect theory problem. *Proceedings of the Explanation-based Learning Workshop* (pp. 42-46). Stanford University.

Drastal, G., Czako, G., & Raatz, S. (1989). Induction in an abstraction space: A form of constructive induction. *Proceedings of IJCAI 89* (pp. 708-712). Detroit.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139-172.

Fisher, D.H., & Schlimmer, J.C. (1988). Concept simplification and prediction accuracy. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 22-28). Ann Arbor.

Hammond, K. (1989). *Case-based planning: Viewing planning as a memory task*. Academic Press.

Iba, W., Wogulis, J., & Langley, P. (1988). Trading off simplicity and coverage in incremental concept learning. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 73-79). Ann Arbor.

Kedar-Cabelli, S.T., & McCarthy, L.T. (1987). Explanation-based generalization as resolution theorem proving. *Proceedings of the Fourth International Workshop on Machine Learning*. Irvine.

Kibler, D., & Aha, D. (1987). Learning representative exemplars of concepts. *Proceedings of the Fourth International Workshop on Machine Learning*. Irvine.

Kolodner, J., (Ed.) (1988). *Proceedings of the Case-Based Reasoning Workshop*. DARPA, Clearwater Beach, Fl.

Lakoff, G. (1987). *Women, fire, and dangerous things: What categories reveal about mind*. University of Chicago Press.

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, *2*.

Michalski, R.S. (1975). Variable-valued logic and its applications to pattern recognition and machine learning. In D.C. Rine (Ed.), *Computer science and multiple-valued logic theory and applications*. North-Holland Publishing Co.

Michalski, R.S., & Larson, J.B. (1978). *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, Reports of the Department of Computer Science). University of Illinois at Urbana-Champaign.

Michalski, R.S. ((1983). A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. Palo Alto, CA: Tioga Pub. Co.

Michalski, R.S., & Stepp, R.E. (1983). Learning from observation: Conceptual clustering. In R.S., Michalski, J.G., Carbonell, T.M. Mitchell, (Eds.), *Machine learning: An artificial intelligence approach*. Palo Alto, CA: Tioga Pub. Co.

Michalski, R.S., Mozetic, I., Hong, J., & Lavrac, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proceedings of the Fifth AAAI* (pp. 1041-1045).

Michalski, R.S. (1989). Two-tiered concept meaning, inferential matching and conceptual cohesiveness. In S. Vosniadou and A. Ortony (Eds.), *Similarity and analogy*. Cambridge University Press.

Michalski, R.S., & Ko, H. (1988). On the nature of explanation, or why did the wine bottle shatter. *AAAI Symposium:: Explanation-Based Learning*, (pp. 12-16). Stanford University.

Michalski, R.S. (1987). How to learn imprecise concepts: A method employing a two-tiered knowledge representation for learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 50-58). Irvine, CA.

Michalski, R.S. (1990). Learning flexible concepts: Fundamental ideas and a methodology. In Y. Kodratoff and R.S. Michalski (Eds.), *Machine learning: An artificial intelligence approach, volume III*. Morgan Kaufmann Publishers.

Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.

Mitchell, T.M., Keller, R. & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, *1*, 11-46.

Mitchell, T.M. (1977) *Version spaces: An approach to concept learning*. Ph.D. dissertation, Stanford University.

Mooney, R. & Ourston, D. (1989). Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. *Proceedings of Sixth International Workshop on Machine Learning*. (pp. 5-7). Ithaca, NY.

Plante, B., & Matwin, S. (1990). *Learning second tier rules by chunking of multiple explanations* Research Report. Department of Computer Science, University of Ottawa.

Prieditis, A.E. & Mostow, J. (1987). PROLEARN: Towards a prolog interpreter that learns. *Proceedings of IJCAI 87* (pp. 494–498). Milan.

Quinlan, J.R. (1987). Simplifying decision trees. *Int. Journal of Man-Machine Studies, 27*, 221–234.

Robinson J.A. & Sibert E.E. (1982). LOGLISP: An alternative to prolog. In J.E. Hayes & D. Michie, (Eds.), *Machine intelligence, vol. 10.*

Rosch, E. & Mervis, C.B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology, 7*, 573–605.

Rouveirol, C. (1991). Deduction and semantic bias for inverse resolution. *Proceedings of IJCAI 91*. Sydney. (to appear).

Sammut, C. & Banerji, R.B. (1986). Learning concepts by asking questions. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. Palo Alto, CA: Tioga Pub. Co.

Smith, E.E. & Medin, D.L. (1981). *Categories and concepts*. Harvard University Press.

Sowa, J.F. (1984). *Conceptual structures*. Addison Wesley.

Sturt, E. (1981). Computerized construction in Fortran of a discriminant function for categorical data. *Applied Statistics, 30*, 213–222.

Watanabe, S. (1969). *Knowing and guessing, a formal and quantitative study.* Wiley Pub. Co.

Weber, S. (1983). A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy Sets and Systems, 11*, 115–134.

Winston, P.H. (1975). Learning structural descriptions from examples. In P. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.

Wnek, J. & Michalski, R.S. (1991). *Hypothesis-driven constructive induction in AQ17: A method and experiments* (Reports of Machine Learning and Inference Laboratory). Center for Artificial Intelligence, George Mason University.

Zadeh, L.A. (1974). Fuzzy logic and its applications to approximate reasoning. *Information processing. North Holland.*

Zhang, J. & Michalski, R.S. (1989). Rule optimization via SG-Trunc method. *Proceedings of the Fourth European Working Sessions on Learning*. Glasgow, December.