Learning Flexible Concepts from Examples: Employing the Ideas of Two-Tiered Concept Representation

Jianping Zhang

MLI 90-15 28

A publication of the Machine Learning and Inference Laboratory
Artificial Intelligence Center
George Mason University
Fairfax, Virginia 22030-4444 U.S.A.
(703) 764-6259

Editor: R. S. Michalski
Assistant Editors: J. Holmes and Michael Hieb

The MLI Reports replace ISG Reports
published until December 1987
by the Artificial Intelligence Laboratory
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

LEARNING FLEXIBLE CONCEPTS FROM EXAMPLES: EMPLOYING THE IDEAS OF TWO-TIERED CONCEPT REPRESENTATION

BY

JIANPING ZHANG

B.S., Wuhan University, 1982

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1990

Urbana, Illinois

LEARNING FLEXIBLE CONCEPTS FROM EXAMPLES: EMPLOYING THE IDEAS OF TWO-TIERED CONCEPT REPRESENTATION

Jianping Zhang, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1990

This thesis describes an exploration of methods involved in learning flexible concepts that is an important and less explored area in machine learning. The two approaches described in this thesis are based on the idea of *two-tiered* (*TT*) concept representations. In the TT representation, the first tier, called the *Base Concept Representation (BCR)*, contains an explicit description of core concept properties, and the second tier, called the *Inferential Concept Interpretation* (ICI), defines allowable modifications of the explicit meaning and its dependence on the context of discourse.

The first approach generates a BCR of a concept by first creating a complete and consistent (CC) description from supplied training examples, and then optimizing the description according to a general description quality (GDQ) criterion. The CC description is obtained by applying the AQ inductive learning methodology. The optimization process is done by a double-level best-first search, using the SG-TRUNC description reduction process. The ICI consists of a user-specified procedure of *flexible matching* and a set of inference rules. This approach has been implemented in the POSEIDON learning system, and experimentally tested on two different real-world problems.

In the second approach, a concept is represented as a set of extended complexes and a set of exemplars. Each extended complex is a generalized description and consists of a base complex (equivalent to a disjunct), a similarity measure, a threshold, and a set of weights. Accordingly, the task of learning a concept is to create a set of extended complexes and a set of exemplars. An extended complex is generated in two phases: base complex generation and extended complex optimization. When generating an extended complex, two evaluation functions, quality evaluation function and potential quality evaluation function are applied. This approach has been implemented in FCLS that is tested in various domains including artificial and real world domains.

DEDICATION

To My Wife, Yilin Weng and
My Parents, Sih-chang Chang and Yaoquan Pan

ACKNOWLEDGMENTS

I am grateful to my thesis advisor, Ryszard S. Michalski, for his advice, supports and encouragement during the period of my dissertation research. The approaches developed in this thesis are based the idea of two-tiered concept representations proposed by him.

I would also like to thank my committee chairman David Wilkins for his valuable helps and suggestions to this research. Thanks are due also to other committee members Larry Rendell, Sylvian Ray and Authur Baskin for their insightful comments and recommendations.

Francesco Bergadano and Stan Matwin deserve special thanks for their joint effort on the work reported in Chapter 3 and 4 of this thesis. Thanks go to Eric Bloedorn for his patient proofreading of my manuscript, to Gheorghe Tecuci for his helpful comments on this research. I appreciate the friendship and the working environment provided by the members of the Machine Learning and Inference Laboratory in George Mason University: Jerzy Bala, Kejitan Dontas, Ken Kauffman, Pawel Stefanski and Janusz Wnek.

I am indebted to Bill Gear for his friendship and helps through the years. He is the person to introduce me to the University and help me to come.

Finally, I would like to express my gratitude to my wife, Yilin Weng, for her love and patience, especially during the years when we were separated. I am grateful to my parents. The selfless supports and encouragements provided by them are unforgettable.

This research was done, while the author was with the Artificial Intelligence Center of George Mason University. The activities of the Center are supported in part by the Defence Advanced Research Projects Agency under grant No. N00014-87-K-0874, administered by the Office of Naval Research, and in part by the Office of Naval Research under grant No. N00014-88-K-0226 and N00014-88-K-0397.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Learning is one of the most important characteristics of human intelligence. Without learning, an agent will repeat an error endlessly, and cannot improve its performance gradually through experience. Such an agent cannot be deemed intelligent. As a result of its important role in human life and intelligence, learning has been an important and active research area for the cognitive sciences, psychology and artificial intelligence.

Artificial intelligence is now experiencing extraordinary growth, and applications of its ideas and methods are appearing in many fields. As more and more complex tasks are set for AI systems, more and more knowledge must be represented in them. The scope of knowledge in any system must be widened to avoid a common problem with current systems, so called *brittleness* (Holland, 1986). Introducing all the required knowledge into any new system is a very complex, tedious and error-prone process, requiring special expertise. The way to simplify the process of introducing knowledge into a system is to build a learning system to acquire the knowledge automatically.

The goal of machine learning is to develop computational models of learning and construct learning machines. The research in machine learning goes back to the 1950's. During the 30 years, numerous learning approaches have been proposed, explored and implemented (Michalski, Carbonell and Mitchell, 1983, 1986; Kodratoff and Michalski, 1990). According to (Carbonell, Mitchell and Michalski, 1983; Michalski, 1986), these learning approaches are classified into five different learning strategies:

- rote learning
- Learning from instruction
- learning by deduction
- learning by analogy
- learning by induction
  - learning from examples
  - learning from observations or discovery

Among these learning strategies, learning from examples is most widely studied. This thesis addresses the problems of learning flexible concepts from examples, and describes methods and implemented learning systems for automatic acquisition of descriptions of flexible concepts from examples.

1.1 Background

The problem investigated in this thesis falls into the category of the learning concepts from examples paradigm. The task of learning concepts from examples has broad applicability, because in many domains, experts can easily come up with a set of examples, while they are unable to articulate the rules they use in making decisions, and in some areas there are no experts.

In the learning from examples paradigm, a set of training examples annotated with concept membership information is used as the basis for automatically inducing a general description for each concept. The concept description learned is both correct for the given examples and a good predictor for the classification of unobserved examples of the concept. The object to be learned in this paradigm is an intentional description of a concept. A concept is a set of objects (called events). The objects to learn from are a set of examples and counter examples of a concept. An example (positive example) of a concept is an object known to belong to the concept, and a counter example (negative example) of a concept is an object that does not belong to the concept.

An example may be a physical object, a situation, a cause, a concept, or nearly anything else that can be described in terms of a given language. Some of the learning tasks which fit this paradigm are:

1 Learning rules for assigning physical objects to classes given examples of physical objects from each of a finite number of classes.
2 Learning condition-action rules. Given examples of when an action should and should not

be applied, a generalized expression for the condition may be learned

3 Learning rules for fault or disease diagnosis from examples of the faults or diseases.

For example, given some poisonous mushrooms (positive examples) and some non-poisonous mushrooms (negative examples), the goal is to construct a description of the "poisonousness" of mushrooms that will allow the correct determination of the edibility of other mushrooms.

The training examples are usually described in one of the two types of representation languages: attribute-based or predicate-based. In an attribute-based representation, an example is represented as an n-tuple of attribute values, where n is the number of attributes. All n attributes define the event space, and each example is a point in the space. Each concept is a subspace of the event space. A domain is associated with each attribute used to describe examples. The domain indicates the values the attribute may assume. The values in a domain may be unordered (or nominal), ordered (or linear), or hierarchically structured (Michalski, 1983). A predicate-based representation allows examples represented as structural descriptions. In structural descriptions, each example may consists of several objects, and a set of relationships among these objects. A predicate-based representation is more powerful than an attribute-based representation, but the limited expressiveness of an attribute-based representation allows relatively efficient learning algorithms to be designed. The attribute-based representation can be used in many real world applications.

Accordingly, a concept is also described in terms of either of the two representation languages. The attributes or predicates used to describe a concept may be different from those used to describe examples. These new attributes (or predicates) are created by a method called constructive induction (Michalski, 1983; Rendell, 1988). In this thesis, an attribute-based representation is selected to describe both examples and concepts.

Concept descriptions produced in early developed inductive learnings, such as AQ (Michalski and Larson, 1978) and ID3 (Quinlan, 1983), are complete and consistent with respect to the examples. A description is complete if it covers all examples of the concept (positive examples), while a description is consistent if it does not cover any counter examples of the concept (negative examples). In order to achieve completeness and consistency in the presence of noise or when concepts are imprecise, one may generate overly complex and detailed descriptions. Such descriptions, however, may not perform well in future cases and suffer the disadvantage of excessive complexity. This is the well known phenomenon of overfitting. Many newly developed approaches, such as PLS1 (Rendell, 1983), C4 (Quinlan, 1987), CN2 (Clark and Niblett, 1989) and AQ16 (Zhang and Michalski, 1989), allow the descriptions produced to be incomplete and/or

inconsistent.

Most inductive learning systems generate concept descriptions by detecting and describing similarities among positive examples and dissimilarities between positive examples and negative examples. Inductively constructing concept descriptions from training examples involves the transformation of training examples using a set of refinement operators (Michalski, 1983). A refinement operator is either a specialization or a generalization operator. When applied to an hypothesis or a training example, a generalization/specialization operator transforms it into a more general/special hypothesis.

Each hypothesis describes a subset of all events, while all hypotheses representable in a given representation language form a hypothesis space. Learning can be viewed as a search through the hypothesis space to find the description of the target concept (Mitchell, 1978). Generalization/specialization operators are search operators. Search heuristics are some preference criteria (also called biases). One of the most important criterion is the accuracy of hypotheses. Hypothesis accuracy depends on the completeness and consistency of these hypothesis with regard to learning examples. Simplicity and comprehensibility are two other preference criteria.

An inductively generated description should not only classify training examples, but also unseen events, so it should be more general than training examples. Unfortunately induction is a necessarily error prone process. It is falsity preserving rather than truth preserving. That is, a concept description inductively generated from examples cannot be guaranteed to be correct; it may be only an approximation of the concept.

Two types of concept descriptions may be generated in the paradigm of learning from examples:

Characteristic description: specifies all common properties of a concept,
Discriminant description: only specifies those properties of a concept that distinguish the concept from the other concepts.

A characteristic description is the most specific and most certain description possible, but it has low predictive power. A discriminant description is the most general and is less certain, but has more predictive power.

What was described above is the inductive learning approach, Another learning approach in the learning from examples paradigm is exemplar-based learning. This approach is also related to the research reported in this thesis. The task of this approach is the same as those of inductive

learning: given a set of training examples labelled with concept membership information, to yield a description for each concept. The major difference between these two approaches is that descriptions produced by the two approaches are represented in different ways. The descriptions generated by an inductive approach are general and abstract, while the descriptions in exemplar-based approach are represented as a set of exemplars that are either given as training examples or constructed by an exemplar-based learning system. Exemplar-based learning involves remembering and indexing specific training events. Classification involves interpreting a new event by recalling a similar exemplar for guidance. In contrast, inductive learning generalizes specific training events to form an general description. Classification involves deducing that a new event is subsumed by the general description. In inductive learning, generalization is performed during learning, while generalization in exemplar-based learning is deferred to classification.

The simplest exemplar-based learning algorithm is called the proximity model (Smith and Medin, 1981), which stores all training examples as exemplars. The best examples model (Smith and Medin, 1981) does not store all given training examples. Instead, only selected, typical examples are stored. The classification of unseen events is determined by computing the similarity of an unseen event with all stored exemplars. An unseen event is classified as being a member of the concept which contains its most similar stored exemplar. Several exemplar-based learning systems, e.g, PROTOS (Bareiss, et al, 1990) and IBL (Aha and Kibler, 1989), have been implemented.

1.2 Motivations

Many current methods of learning concepts from examples assume that concepts are precise and context-independent entities, representable by a single symbolic description. A recognition of such concepts is therefore very simple: if an event satisfies a concept description, then it belongs to the concept, otherwise it does not. In this way, events that are similar to the concept description, but which do not satisfy all conditions of the concept description, will not be recognized as being a member of the concept. Another common assumption is that concept examples are equally representative, that is, there is no distinction between typical and less typical examples. All events of a concept would satisfy the concept description equally well.

When one looks at human concepts, however, one can see that most of them inherently lack precisely defined boundaries and have a central tendency, and their meaning is often context-dependent. Examples of human concepts are usually not all equivalent. They may be characterized

by a *degree of typicality* in representing a concept. The examples represented the central tendency of the concept are more typical than the others. For example, a robin is usually viewed as a more typical bird than a penguin or an ostrich. Moreover, in different contexts, the concept "bird" may apply to a live, flying bird, a sculpture, a chick hatching out of the egg, or even an airplane. Thus, human concepts are *flexible*, as they can modify or change their meaning in different situations or contexts.

The imprecision of flexible concepts often has logical, rather than only probabilistic character. This means that the classification of a non-typical example of a flexible concept normally involves reasoning, rather than the calculation of probabilities. In other words, to decide the concept membership of an example that is non-common, borderline, or irregular in any sense, one may reason deductively from one's background knowledge, draw an analogy or perform induction, instead of relying on a statistical analysis.

Finally, the complete concept meaning perceived by a person depends on the amount of knowledge this person possesses about it. The difference lies in the number of facts they know about, and in the depth of their understanding of the concept and its properties. Such background knowledge-dependency in understanding a concept indicates that human concepts are personalized, living and growing constructs, rather than fixed and stable entities that mean exactly the same thing to all those using them. As the meaning of concepts may change from individual to individual and evolve in time, such concepts cannot be defined precisely as objective impersonal entities with a context-independent meaning. Thus, in general, human concepts are very different entities than well-defined and context-independent structures that we use to represent concepts in today's computer systems.

In the real world, very few measurements and decisions are known with 100% confidence. Often, it is not possible to collect all relevant information before a decision must be made. The information gathered is likely to contain errors and inconsistencies. The resources available are often limited, forcing one to stop when a "good enough" solution is found even it is not optimal. The rules used to make a decision may only work well for the most frequently encountered problems, but exceptional circumstances may require special consideration. For example, in disease diagnosis it is usual to check for a common disease associated with certain symptoms before doing expensive tests for rare diseases with similar symptoms.

It is clear that in order to handle such flexibility of concepts, machine learning systems need to employ richer concept representations than are currently used. Developing methods for acquiring flexible concepts and reasoning with them emerges as an important goal for machine

learning research. Specifically, to represent and learn flexible concepts, the following three problems must be solved:

- **Imprecise and irregular boundaries.** Crisp concepts are defined by a set of necessary and sufficient conditions, or a disjunction of such conditions. The boundaries of such concepts are precise and rectangular. In contrast to crisp concepts, no such necessary and sufficient conditions exist for flexible concepts and the shape of flexible concepts is not rectangular Furthermore, the boundaries of flexible concepts is imprecise. That is, the classification of the events that are around boundaries are not clear.

- **Varying degree of typicality.** Examples of flexible concepts have varying degree of typicality. Typical examples share more common properties than do less typical examples. Typical examples represent central tendencies of the concepts. Psychological results (Smith and Medin, 1981) showed that typical examples can be easily and quickly recognized, but less typical examples cannot.

- **Context-dependency.** As we described above, the meaning of a concept depends on the context in which it is used, also depends on the amount of knowledge a person possesses about it. The central tendency or the basic idea of a concept does not change with the context. Exceptional cases often occur in some special contexts.

Traditional logic-based representations fail to handle any one of the problems described above. However, there have been many attempts at solving it. One of the most prominent is the work on fuzzy sets by Zadeh and his collaborators and many followers (e.g., Zadeh, 1976; Mumdani and Gaines, 1981). In cognitive science literature, the inadequacy of representing human concepts by context-independent, logic-style definitions (the classical view), has been widely recognized (e.g., McCloskey and Glucksberg, 1978; Barsalou and Medin, 1986; and Lakoff, 1987). The probabilistic view and the exemplar view have been proposed to solve such an inadequacy (e.g., Smith and Medin, 1981; Medin and Smith, 1984). In machine learning, many methods have been proposed and implemented. These methods and systems include tree pruning algorithms (e.g., Quinlan, 1987), rule truncation algorithms (e.g., Michalski, et al., 1986), probabilistic learning algorithm PLS1 (Rendell, 1983),the two exemplar-based learning methods Protos (Bareiss, et al. 1987), and IBL (Aha and Kibler, 1989), and STAGGE (Schlimmer, 1987).

1.3 Thesis Overview

The general goal of the research reported in this thesis is to investigate approaches to represent and learn flexible concepts. The approaches investigated are based on the idea of two-tiered concept representations proposed by Michalski (1987). In a two-tiered concept representation, the meaning of a concept is defined by two components, the *base concept representation* (BCR) and the *inferential concept interpretation* (ICI). The BCR defines explicitly the properties of the concept, while the ICI describes allowed modifications and transformations of the concept's properties in different situations and/or contexts. Chapter 2 introduces the basic idea of a two-tiered concept representation and gives a few illustrative examples. Many of the material in Chapter 2 is excerpted from (Michalski, 1990).

In Chapter 3, a method that learns two-tiered descriptions of flexible concepts is described. The method generates a BCR of a concept by first creating a complete and consistent (CC) concept description from supplied training examples, and then optimizing the description according to a general description quality (GDQ) criterion. The CC description is obtained by applying the AQ inductive learning methodology. The optimization process is done by a double-level best-first search, using the SG-TRUNC description reduction process. The ICI consists of a user-specified procedure of *flexible matching* and a set of inference rules.

The method has been implemented in the POSEIDON learning system, and experimentally tested on two different real-world problems: learning the concept of an acceptable union contract and learning the distinction between republicans and democrats in the U.S. Congress. For comparison, a few other learning methods were also applied to the same problems. These methods included simple variants of exemplar-based learning and the decision tree learning, based on the ASSISTANT program. The results have shown that the POSEIDON generated concept descriptions outperformed those produced by the other methods both in terms of the performance on new examples and in terms of simplicity. All experimental results are reported in Chapter 4.

Chapter 5 describes an alternative method that learns flexible concepts, and its implementation: Flexible Concept Learning System (FCLS). The representation used in the method can be viewed as a simple form of the two-tiered concept representation. This method combines inductive learning method with exemplar-based learning method. In the method, a concept is represented as a set of extended complexes and a set of exemplars. Each extended complex is a generalized description and consists of a base complex, a similarity measure, a threshold and a set of weights. Accordingly, the task of learning a concept is to create a set of extended complexes and a set of exemplars. An extended complex is generated in two phases: base complex generation and extended complex optimization. When generating an extended complex, two evaluation functions,

quality evaluation function and potential quality evaluation function are applied.

In Chapter 6, FCLS is tested in various domains including the domains favorable for FCLS, the domains unfavorable for FCLS, and real world domains. The results are compared with the results from the decision tree learning system C4.5. In Chapter 7, some related work is discussed. Limitations and possible future work are summarized in Chapter 8.

CHAPTER 2

TWO-TIERED CONCEPT REPRESENTATION

The idea of the two-tiered (TT) representation was originally proposed by Michalski (1987), and was motivated by an observation that although individual human concepts may lack precise definition when used along in a context-independent sense, they acquire a precise meaning when used in a combination with other concepts and in a specific context. Consider, for example, the statement: "This tall man in the group in the corner of the room." If there is only one man visible taller than other people in the indicated group, the statement above precisely specifies the man of interest. The concept "tall", although by itself and/or without context is imprecise (as well as concepts "group", "corner", or "room"), in the context conveys a precise meaning, that of the height of the man pointed out in the group.

Thus, the TT approach views flexible concepts as inherently and intentionally imprecise, when they are considered alone and outside of a specific context. Consequently, it does not try to give them a complete and precise meaning in an explicit and context-independent sense. Instead, it attempts to describe precisely only the central tendency, and use inference rules and matching procedures to implicitly characterize the complete concept meaning and context-dependency.

The underlying supposition of the TT approach is that the imprecision of human concepts stems not from an undesirable vagueness of our concept definitions, but rather from the universal need for cognitive economy. By allowing concepts to have a context-modifiable meaning, and making them precise only to the extent to which a given situation and/or context requires them to

be precise, the expressive power of concepts is greatly enhanced. The latter means that one can employ fewer concepts for expressing more meanings, and this helps us to simplify our descriptions of our immensely complex universe. The experiments reported in this thesis seem to confirm this idea in a microworld to which it was applied.

2.1. The Basic Ideas

In order to develop a computational method for learning concepts, one needs to make assumptions about the meaning of "concepts" in the method. In the research reported in this thesis, it is assumed that concepts are named representations of classes of entities, whose borderlines are variable and context-dependent. The entities are assumed to have central tendencies within the concept classes, and therefore different concept instances may be characterized by different typicality. A concept representation can take a wide range of forms: an explicit description of observable properties of the entities in the class, an abstract description of the function of the entities and their relation to other concepts, a complete or partial listing of the entities, an implied concept characterization by the concept usage, or as a combination of the above. There can be an enormous variation in the specific instantiation of some concepts. Consider, for example, the concepts such as "object" or "set". Because of the assumed central tendencies, context-dependence and other previousely mentioned properties, a concept representation should allow a varying degree match with concept instances, and use of context-dependent inference rules in performing such matches (examples below illustrate this point in more detail). The problem then is how a concept with such central tendencies and context-dependent meaning can be efficiently represented and learned?

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

In traditional approaches, recognizing a concept instance typically involves a direct matching

of the properties of the instance with the stored concept representation. Such a matching may include comparing feature values in an instance with those in the concept description, or tracing links in networks of concepts, but has not been assumed to involve any complex inferential processes. More recently, researchers working on exemplar-based reasoning (e.g., Bareiss, et al., 1989; Kolodner, 1988 and Hammond, 1989) have recognized the need for using advanced inference mechanisms for classifying new instances. In these methods, however, the concept description consists of stored individual examples. Such a representation is memory-taxing, makes matching concepts with instances more complex, and does not allow one to easily determine the similarities or differences between different concepts.

In contrast to the above, the two-tiered (TT) representation involves a combination of an explicit general concept description and an inference mechanism for matching the description with individual instances. This way, the concept representation can be simpler than one based on storing individual examples, and can be easily compared with descriptions of other concepts. At the same time, it allows one to perform a sophisticated reasoning when classifying new instances.

Research on the TT approach has been motivated by the observation that human knowledge can be viewed as a combination of two components, recorded knowledge and inferential extension, i.e., knowledge that can be easily inferred from recorded knowledge. Extending this property to individual concepts, Michalski (1987) has suggested that the meaning assigned to concepts also combines such two components.

Specifically, the two-tiered concept representation consists of two parts: a *Base Concept Representation* (BCR) and an *Inferential Concept Interpretation* (ICI). The BCR defines the concept explicitly, by giving a description of the concept in terms of attributes observed in examples and/or higher level concepts employed during concept formation. The BCR can be viewed as a characterization of typical concept instances. It captures the principle, the most relevant properties and/or the intention behind the concept.

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

In the general approach, the "distribution" of the meaning between the BCR and the ICI is not assumed to be fixed, but is modifiable based on a criterion of description quality. One extreme

of such a distribution is when the BCR represents explicitly all possible concept variations in different contexts, and the ICI is just a direct match. Another extreme is when the BCR is empty, and all concept meaning resides in context-dependent inference rules. The description quality criterion reflects computational properties of the learner and requirements of the problem domain. The former ones include the relative costs of remembering concept properties vs. deriving them through inference.

In an important, cognitively-oriented special case of the TT representation, the BCR is assumed to express the general unifying idea, the typical function of the concept, and/or common measurable properties implied by or correlated with this idea or function. Such BCR can be viewed as representing the "first approximation of the concept." The ICI, in this case, defines the matching procedures and inference rules for handling less typical instances and context-dependency. This type of distribution of the concept meaning between the two tiers facilitates an efficient concept recognition, and is related to the idea of censored production rules (Michalski and Winston, 1986).

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

An advantage of distributing the concept meaning between the BCR and the ICI is that it permits a learner to flexibly modify or extend the concept meaning by varying matching procedures and inference rules and/or by changing the context of discourse. The concept meaning can thus be changed without having to alter the base concept representation. By evaluating the type and the amount of inference involved in matching a concept with an instance, one may produce a qualitative or quantitative estimation of the strength of such a match. The ability to produce a measure of the strength of match indicates one principle difference between this approach and the approach associated with a membership function, which needs to be defined to the learner by a person. The influence of the context is hidden in the definition of this membership function. In the proposed TT approach, a concept is associated with interpretation procedures and context-dependent inference rules, which implicitly define the membership of an instance in a concept. These rules and procedures can be used to compute the membership function in different contexts.

Another advantage of representing a concept as two parts is its flexibility of storing and using a concept description. The BCR represents the central tendency of a concept, and tends to be satisfied much more frequently than applying the ICI. This makes it possible to allow a trade off

among speed, precision, and memory in concept recognition. The BCR can be used alone for rapid, low cost classification, but with somewhat less precision than if the ICI is applied. The ICI does not have to stay in memory until it is applied, therefore, the two-tiered representation approach can be applied to realize the idea of variable precision logic (Michalski and Winston, 1986).

Figure 2.1 illustrates the relationship between the BCR and the ICI in a TT concept representation. It shows that the ICI can, in general, extend the concept meaning beyond the BCR in one area of the description space, and reduce the meaning in another area.

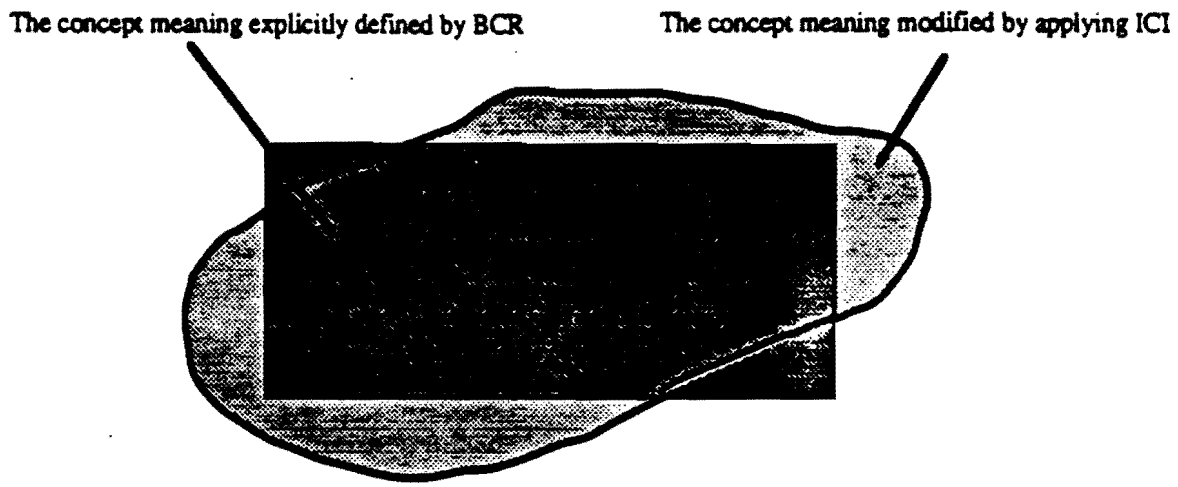


Figure 2.1: An illustration of the relationship between
the Base concept Representation (BCR) and the Inferential Concept Interpretation (ICI)

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

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

used materials, etc.).

ICI: Number of legs can vary from one to four. The material of which the chair is made, the shape of the seat, the backrest and the legs are irrelevant as long as the function is preserved. Backrest may be missing. The legs may be replaced by any support.
If legs are replaced by wheels --> type(chair) is wheelchair
Chair without the backrest --> type(chair) = stool
Chair with the armrests --> type(chair) = armchair
Context = museum exhibit --> chair is not used for seating persons any more, but has all the physical characteristics of a chair.
Context = toys --> the size can be much smaller than typical. The chair does not serve for seating persons, but correspondingly small dolls.

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

The TT concept representation approach appears to have a strong cognitive justification. A simple introspection tells us that people regularly make generalizations of specific facts, in addition to memorising concept examples (usually only the most "significant" ones). They use these generalizations for distinguishing between different concepts and for classifying new examples. When a generalization is insufficient for classifying some examples, they conduct reasoning, which may involve various kinds of relevant knowledge and different types of inference.

The idea of the TT representation is also supported by research on the so-called transformational model (Smith and Medin, 1981). In this model, matching object features with concept descriptions may involve transformations of object features into those specified in the concept description. Such a matching has an inferential character. Some recent work in cognitive

linguistics also seems to support the ideas of the TT representation approach. For example, Lakoff (1987), in his idealized cognitive models (ICM) approach, stipulates that humans represent concepts as a structure, which includes a fixed part and mappings that modify the fixed part. The fixed part is a propositional structure, defined relative to some idealized model. The mappings are metaphoric transformations of concepts meaning.

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

2.2 Examples Illustrating The Ideas

In this section, a few examples are given to illustrating the idea of the two-tiered representation. All examples are excerpted from (Michalski, 1990).

Example 1: Concept of sugar maple

Our prototypical image of a sugar maple is that it is a tree with three to five-lobbed leaves that have V-shaped clefts. Some of us may also remember that the teeth on the leaves are coarser than those of red maple, that slender twigs turn brown, and the buds are brown and sharp-pointed. As a tree, of course, a maple has a root, trunk and branches.

Suppose that while strolling on a nice winter day someone tells us that a particular tree is a sugar maple. A simple introspection tells us that the fact that the tree does not have leaves would not strike us as a contradiction of what we know about sugar maples. Yet, clearly, the presence of leaves of a particular type is deeply embedded in our typical image of a maple tree. The two-tiered approach explains this phenomenon simply: the inferential concept interpretation associated with the general concept of deciduous trees evokes a rule "in winter deciduous trees lose leaves." Since a maple is deciduous tree, the rule would apply to the maple tree. The result of this inference would override the stored standard information about maple trees, and the inconsistency would be resolved. Matching an instance with the concept requires in this case deductive reasoning from the knowledge associated with a more general concept.

Example 2: The concept of an abstract tree structure

Suppose that a student is reading his first book on computer data structures and encounters a drawing of a graph structure, which the author calls a tree. Calling such a structure a tree will likely not evoke any objection in the student, because one can see in this structure some abstracted and modified features (e.g., upside down direction) of a biological tree. In this case, matching the graph structure with the concept of a tree involves a generalization operation on the base representation.

Examples 3: Concept of a triangle

Let us consider the concept of a triangle. Formally, a triangle can be described as a geometrical figure consisting of three non-colinear points connected by straight lines. Using the notation of annotated predicate calculus (APC), which is equivalent to predicate calculus, but permits one to write logical expressions in a more compact form (Michalski, 1983), one can write:

Triangle(T, P1, P2, P3) <= Consists(T, P1 & p2 & p3) & Type(P1 & P2 & P3) = point &
Connected_by(P1, P2 & P1, P3 & p2, p3) = straight_line &
RelationAmong(P1, P2, P3) = non-colinear

In the above formula, the symbol "&" is used in two related meanings; one - to denote an ordinary (external) conjunction connecting predicates, and second - to denote an internal conjunction, i.e., conjunction of terms, treated as a compound argument of a predicate. For example, the predicate "Consists(T, P1 & P2 &p3)" states that the triangle T consists of points P1 and P2 and P3.

Suppose that someone tells us that the tall towers in his hometown form a big triangle. Obviously, the meaning of the triangle in this statement differs from that in the formal geometrical description. To match the two, one needs to make the following assumptions and transformations:

1. In the context of describing a configuration of physical objects such as towers, the individual objects play the role of nodes. Thus, the statement implies that there are three towers in the town. The matching operation involves drawing an analogy between the abstract nodes and the towers, which can be characterized as consisting of one step of generalization (GEN):

   Point ——— GEN ——> Object

   and one step of specialization (SPEC):

Object——SPEC ----> Tower

2. In the context of towers, the presence of a "straight line" is imaginary, i.e., there is no physical connection, but one could imagine a straight line between the objects (towers). The condition "Connected_BY" is satisfied in such an abstract sense. This is an operation of generalization. Thus, matching the statement about a triangular arrangement of towers with the formal definition of a triangle involves here both, a generalization and specialization.

The examples above show that relating a concept instance to a concept description is not just a straightforward comparison of attribute values in an instance with those in the concept description, as done in various mechanized decision processes. They show that such a process may involve different forms of inference.

2.3. Learning Two-tiered Concept Descriptions

Accordingly, learning a two-tiered concept description consists two parts:

1. acquiring the base concept representation, and
2. acquiring the inferential concept interpretation

The base concept representation includes a set of assertions that describes the principles of the concept. The inferential concept interpretation may include a set of inference rules, matching procedures, the importance of properties and a set of exceptions. The BCR is usually learned from typical examples, while the ICI is learned from the less typical examples and exceptions. Furthermore, the matching procedures are responsible for the less typical examples whereas the inference rules are responsible for the exceptions. The BCR and the matching procedure can be inductively learned from typical and less typical examples. But because the number of exceptions is limited, it is difficult to learn using an inductive learning method. Following a few possible methods that can be applied to learn ICI rules are simply descussed.

First, the ICI rules can be inherited from some higher level concepts. Suppose that the system is given an instance of a broken chair. More precisely, the description of the instance matches neither the BCR nor the ICI of the chair, because one of the four legs is missing. Suppose that the ICI of the leg concept will deduce from this description that the shorter leg is broken. The chair concept is a specialization of the higher level concept furniture. Suppose that the ICI for furniture has the following rules: if an object misses one leg then it is broken and if some part of an

object that otherwise is an instance of furniture is broken, then the object still is an instance of furniture. Through these two rules, one can infer that a chair which misses one leg is a broken chair. These two rules may now be inherited by the ICI of the chair concept.

Second, an analytic learning system, such as described in (Mitchell et al., 1986; DeJong and Mooney, 1986), can be applied to obtain an explanation. In these systems, a typical event is explained by deductively inferring it from the underlying background theory. The result is an operational rule for the concept. This rule is then generalized, e.g. using techniques described in (Mitchell et al., 1986). In our approach, only nontypical events will be subject to explanation. The purpose of the explanation here is to justify the special character of the event explained, rather that to operationalize the proof of its membership in the concept.

Finally, an explanation of an event may be obtained from an expert, as is the case in knowledge acquisition for expert systems. The POSEIDON system described in next chapter uses this method in its initial version.

CHAPTER 3

THE *POSEIDON* LEARNING SYSTEM

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

This chapter describes the learning system POSEIDON[1] (Bergadano et. al, to appear) that learns two-tiered descriptions of flexible concepts. POSEIDON is an extension and continuation of the early ideas in (Michalski et al., 1986). One important advance is the development of a heuristic double-level search procedure, called SG-TRUNC, which explores the space of two-tiered descriptions in order to determine a globally optimized description. The search employs both generalization and specialization operators, and is guided by a new criterion, the *general description quality* measure *(GDQ)*. This measure takes into consideration the description

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

accuracy, its cognitive comprehensibility and the computational cost of both tiers of a description (BCR and ICI). By introducing such a general description evaluation measure (Bergadano et al., 1988), the meaning of concept learning can be defined in a new way. Namely, concept learning can be viewed as a process of improving initial descriptions according to a given description quality criterion. The initial description can be in the form of positive and negative examples, a Complete and Consistent (CC) concept description, a teacher-supplied tentative description, or other.

Other advances include the use of a more powerful accuracy measurement in GDQ, which is able to take into consideration the typicality of training instances (when it is known), and the introduction of a rule base for implementing the ICI. the following sections describe the overview of the POSEIDON system, next chapter shows the experimental results of POSEIDON on two real world domains.

3.1. Concept Representation

Concepts in POSEIDON are represented in a two-tiered concept representation. In this two-tiered concept representation, a concept is represented by two parts, the BCR which is equivalent to a disjunctive normal form and the ICI which consists of a predefined flexible matching function and a set of deductive rules. This section describes this representation.

3.1.1 Base Concept Representation

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

$$[L \# R]$$

where the attribute L is called the *referee*. R, called the *referent*, is a set of values from the domain of L. Symbol # denotes one of the relational symbols $=$, $<$, $>$, $\leq$, $\geq$, $\neq$. A selector is satisfied by an event if the value of attribute L in this event is in relation # to R.

For example, the expression [shape = circle v square] & [length > 2] is a complex containing two selectors. The first selector is satisfied by an event, if the attribute "shape" in this event takes value "circle" or "square".

3.1.2 Inferential Concept Interpretation: Flexible Matching Function

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

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

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

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

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

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

$$F(e,cpx) = \sum_{i=1}^{n} \frac{F(e,sel_i)}{n} * \frac{\#cpxpos}{(\#cpxpos + \#cpxneg)}$$

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

The degree of match between an event *e* and a selector *sel* is weighted by the coverage of positive and negative examples by the selector. Such a degree of match depends on the type of the

attribute in the selector. An attribute can be one of two types: nominal and linear (Michalski,1983). In a nominal selector, the referent is a single value or an *internal disjunction* of values, e.g., [color = red v blue v green]. The degree of match is 1 if such a selector is satisfied by an event, and 0 otherwise. In a linear or structured linear selector, the referent is a range of values, or an internal disjunction of ranges, e.g., [weight = 1..3 v 6..9]. A satisfied selector returns the value of match 1. However, if the selector is not satisfied, the degree of match is the ratio of the distance between the value of the attribute in the event and the nearest value in the referent, and the size of the domain of the attribute. For example, suppose the domain of the attribute *height* has 10 values: 1 to 10, and the event is (weight = 2, height = 4, ...), and the selector is [height = 7..9]. The degree of match is $\frac{7-4}{10} = 0.3$.

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

3.1.3. Inferential Concept Interpretation: Deductive Rules

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

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

3.1.4. Types of Matching

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

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

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

Now a precise meaning to the term *exception* used throughout this chapter can be given. Events are called *exceptions*, if they are D-covered. Events that are S-covered are said to be *representative* examples, and events that are F-covered are said to be *nearly-representative* examples.

Two-tiered concept descriptions are usually simpler, easier to understand and more efficient to use than concept descriptions represented in a conventional logic type representation. In the experiments described in next chapter, they also performed better on testing sets. One major advance of the presented method over the previous method using TT representation (e.g., Michalski et al, 1986) is that the ICI includes not only a flexible matching procedure, but also inference rules. Thus, the method is oriented toward handling not only representative or nearly representative examples, but also exceptions.

3.2. An Overview of POSEIDON

In POSEIDON, a two-tiered concept description is produced in two phases. First, a complete and consistent (CC) description is created from supplied training examples by applying the AQ inductive learning methodology, and then the CC description is optimized according to a general description quality (GDQ) criterion. The optimization process is done by a double-level best-first search, using the SG-TRUNC description reduction process. ICI rules are provided by human experts. This section describes the overview of the algorithm used in POSEIDON and discusses the operators used in the SG-TRUNC description reduction process, the detailed description of the algorithm is deferred to section 3.4.

3.2.1. Basic Algorithm

Based on the ideas presented above, we have implemented a system for learning TT descriptions, called POSEIDON. Figure 3.1 specifies the input, output, and the function of the system. The BCR is determined in two phases. The first phase generates a general consistent and complete (CC) concept description, and the second phase optimizes the description according to a criterion of concept quality.

The CC description is determined using the AQ15 inductive learning program (Michalski et al. 86). The initial BCR consists of the CC description together with the flexible matching procedure described in section 3.1.2.

Phase 1

Given:

Concept examples obtained from a source

Relevant background knowledge

Determine:

Complete and consistent (CC) description of the concept

Phase 2

Given:

CC description of the concept

A general description quality (GDQ) measure

Typicality of examples (if available)

Determine:

A two-tiered description that maximizes the GDQ

Figure 3.1 . Specification of the POSEIDON system

The second phase improves the initial description by conducting a "double level" best first search. This search is implemented in the SG-TRUNC procedure (SG symbolizes that the method uses both specialization and generalization operators). The first level search is guided by the

General Description Quality measure (GDQ), which ranks descriptions for consideration (see sec. 3.4). The second level search is guided by heuristics that specify the search operators to be applied to a given description. The search operators simplify the BCR of a description by removing some of its components, or by modifying the arguments or referents of some of the selectors. A general structure of the system is presented in Figure 3.2.

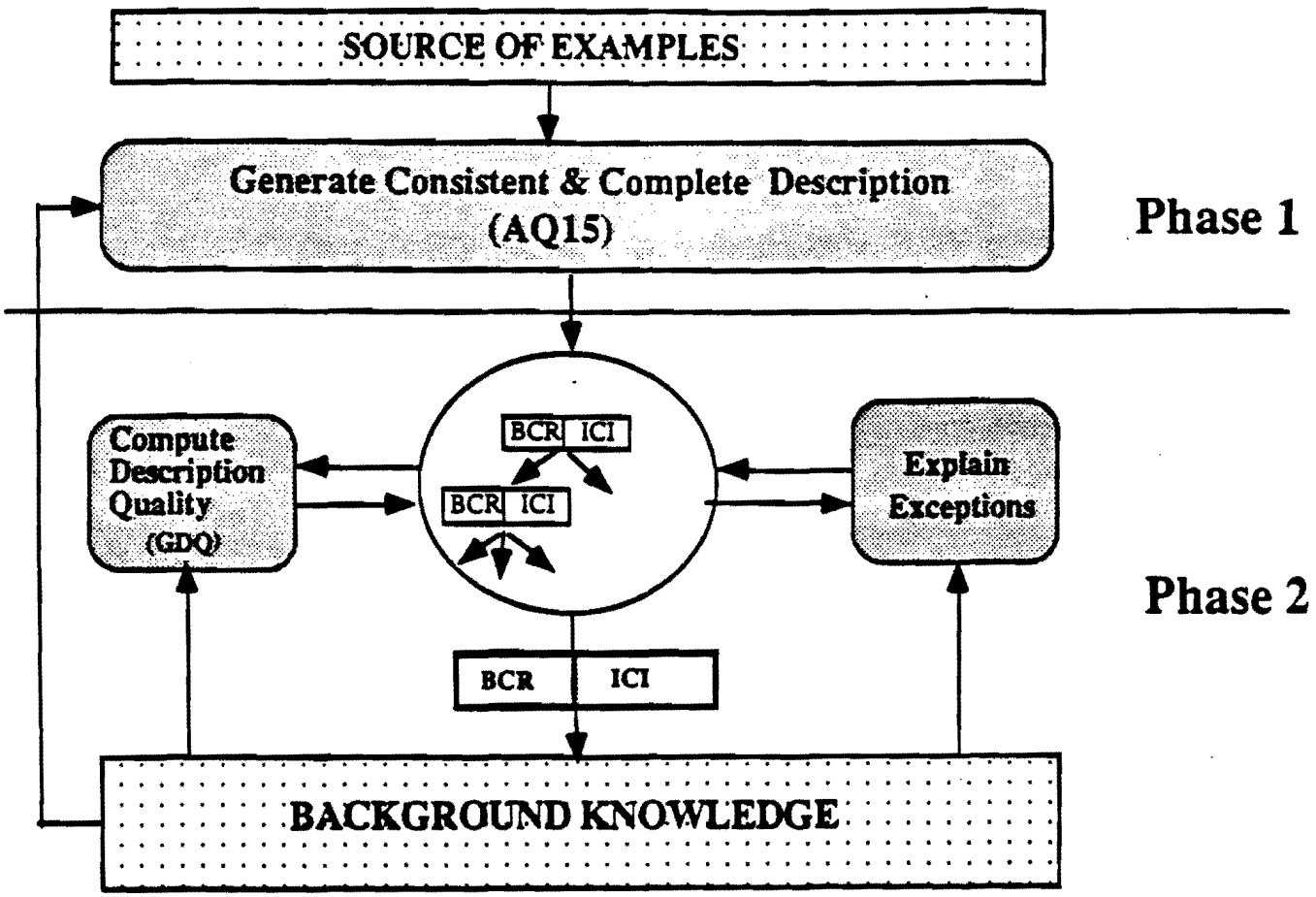


Figure 3.2. Learning in the POSEIDON system

The search process is defined by:

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

criterion (GDQ).

The goal of the search procedure is not necessarily to find an optimal solution, as this would require a combinatorial search. Rather, the system tries to maximally improve the given concept description by expanding only a limited number of nodes in the search tree, which are suggested by certain heuristics to be discussed in section 3.4.1.

The BCR is learned from examples and represented as a set of rules (a VL_1 cover, as described in section 3.1.1). The ICI consists of two parts: a flexible matching function and a rule base. The rule base is used for explaining exceptional examples, and is acquired through an interaction with an expert.

3.2.2. Operators Used in Optimizing Base Concept Representation

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

[size = 1..5 v 7]

A referent modification that produces a generalization

[size = 1..7]

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

[size <> 1..5 v 7]

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

[size = 1..5]

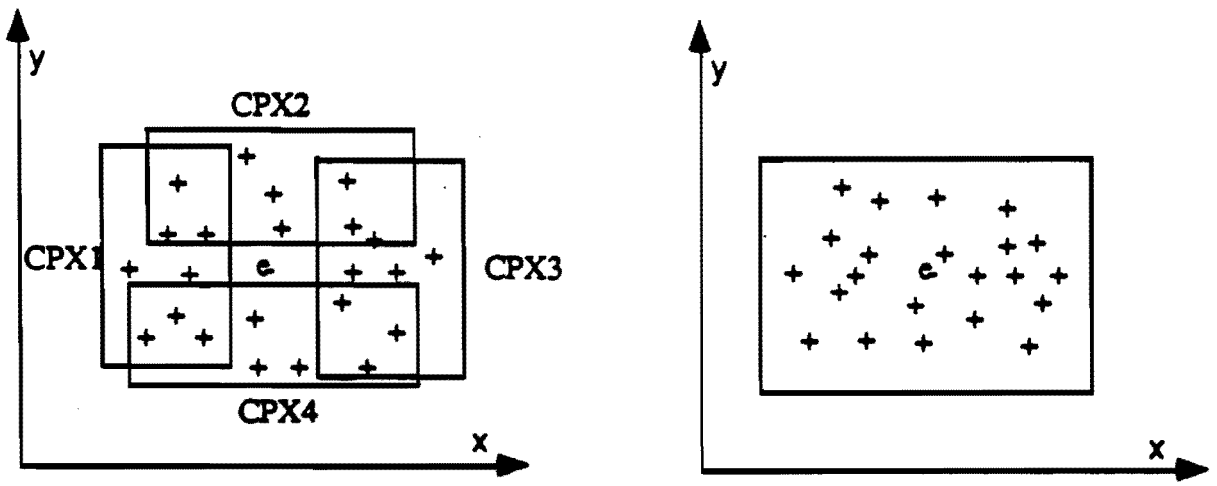
represents also a referent contraction operator. Figure 3.3 lists search operators and specifies their effect on the description.

The removal of conditions may merge complexes. The negative exceptions (the exceptions from negative examples) break one complex into several. In Figure 3.4(a), the exception e breaks

one complex into four. If the exception is allowed to be covered by some of the four complexes, some conditions can be removed from one of the four complexes so that the four complexes merge into one larger complex (see Figure 3.4(b)).

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

Figure 3.3: Search Operators

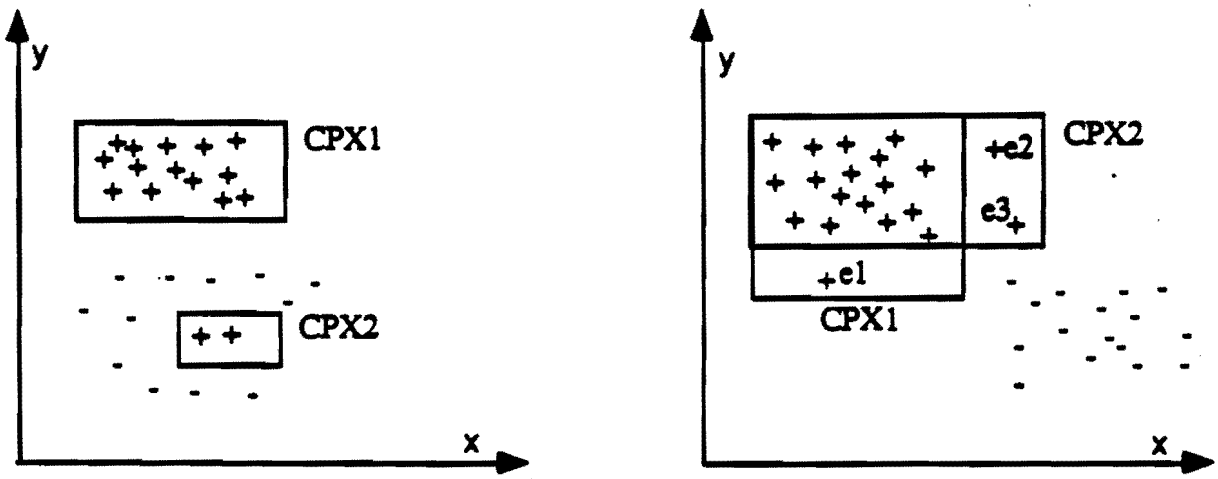


(a): Before selector removal (b) After selector removal

Figure 3.4: How the selector removal merges complexes

Two types of complexes can be removed. The first type of complexes is the complexes that cover a small number of positive examples and is called light complexes (small disjuncts). The examples covered by light complexes are often far from the other positive examples, thus these complexes often describe rare, exceptional cases. If the learning examples from which complexes are derived are noisy, a light complex may also be an indicative of errors in the data. Figure 3.5(a)

illustrates this situation. CPX1 and CPX2 are two complexes and cover 14 and 2 examples, respectively. CPX1 is a representative complex of the concept. The two examples covered by CPX2 are considered as exceptions or noise. The second type of complexes that can be removed is the complexes that cover a large number of examples, but uniquely cover a small number of examples. This type of complexes carries a lot of redundant information and is called as redundant complexes. The examples uniquely covered by a redundant complex can be viewed as less typical examples. They are not exceptions, because they are still close to other positive examples. In Figure 3.5(b), both CPX1 and CPX2 could be redundant complexes. e1, e2 and e3 are less typical examples.



(a) Complex with low t-weight (b) Complexes with low u-weights

Figure 3.5: types of complexes that can be removed

The loss of coverage resulting from removing redundant complexes may be restored by flexible matching. After a redundant complex is removed, only those examples uniquely covered by it are no longer covered. Since these uncovered examples are close to many typical examples which are still covered by some other complexes. That is, these uncovered examples are close to some remaining complexes, so they can be easily matched with these complexes by flexible matching. Suppose CPX1 in Figure 3.5(b) is removed, e1 is no longer covered strictly, but it can be matched with CPX2 by flexible matching. The loss of coverage that results from removing a light complex cannot be easily restored by flexible matching. In this case, the examples that are no longer covered are not close to any remaining complexes. In Figure 3.5(a), after CPX2 is removed, the two examples covered by it are not close to CPX1, so they cannot be matched with

CPX1 by flexible matching. In order to restore the loss of coverage resulting from removing such complexes, some ICI rules are needed.

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

3.2.3 Learning The Inferential Concept Interpretation.

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

The rules can be supplied by an expert, inherited from higher level concepts, or learned from other knowledge. If the rules are supplied by an expert, their form may not be operationally effective. In such a case, they can be made effective through some form of analytic learning (e.g., Mitchell et al., 1986; Prieditis and Mostow, 1987). If the rules supplied by an expert are too specific or not totally correct, they may be improved by induction (e.g., Dietterich and Flann 1988, Mooney and Ourston, 1989). Thus, learning the ICI can be accomplished by different strategies, like learning the BCR.

In the implemented method, the system identifies exceptions (i.e., examples not covered by the current BCR and ICI), and asks an expert for an explanation (see sec. 3.4.2). The expert is supposed to justify the special character of such events, and express this justification in the form of rules. The search procedure shown in Figure 3.2 guides the process by determining the examples that need explanation. It is a hard task to distinguish exceptions from noise. In our method, any

exceptions that cannot be explained by an expert are treated as noise and are ignored.

3.3. Quality of Concept Descriptions

The learning methodology utilizes a general description quality measure (GDQ) that guides the search for an improved two-tiered description. GDQ plays an important role in the proposed two-teired concept description learning approach. This section first discusses some general issues about the GDQ measure, then define the GDQ measure implemented in POSEIDON.

3.3.1 Factors Influencing The Description Quality

The learning methodology utilizes a general description quality measure (GDQ) that guides the search for an improved two-tiered description. The GDQ measure is influenced by three basic characteristics: the *accuracy*, the *comprehensibility*, and the *cost*. This subsection 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. A common method for improving accuracy is to require that the description be complete and consistent with respect to the training events (Michalski, 1983). When a description is incomplete and/or inconsistent, the number of positive examples it does not cover and the number of negative examples it covers provides important information for evaluating its quality. Those numbers can be used to measure the degree of completeness and consistency of a given description. To achieve completeness and consistency when input examples are noisy, overly complex and detailed descriptions may be generated. Such descriptions tend to be too dependent on the training set, and, consequently, may perform poorly in classifying future examples. This phenomenon is well known in statistics as overfitting (Watanabe, 1969; Sturt, 1981).

The comprehensibility of the acquired knowledge depends on subjective and domain-dependent criteria. Because an AI system is often supposed to supply advice to humans, knowledge used by such a system should be understandable by human experts. A "black box" classifier will not be helpful in this respect. Therefore, knowledge acquired by a learning system should be related to terms, relations and concepts used by experts, and should not be syntactically too complex. This is called the *comprehensibility* principle (Michalski, 1983). There is no well established measure of description comprehensibility. Therefore we approximate it by a

representational simplicity of the description. The simplicity (an inverse of complexity) is evaluated by counting the number of operators involved, and taking into consideration an estimated cognitive complexity of the operators. Such an approximation of the comprehensibility of a two-tiered representation 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 a description related to its storage and use (computational complexity). Other things being equal, descriptions which are easier to store and/or easier to use for recognizing new examples are preferred. When evaluating the cost of a description, two characteristics are of primary importance. The first is the cost of measuring the values of variables occurring in the description. In some application domains, e.g., in medicine, this may be a very important consideration. The second characteristic is the computational cost of evaluating the description. Again, some real-time applications, e.g. speech or image recognition, impose constraints on the evaluation time of a description. The cost (approximated by computational simplicity) and the comprehensibility (approximated by representational simplicity) are usually related to each other, but, in general, these are two different criteria.

The criteria discussed above are applied to two-tiered descriptions. The accuracy of the acquired knowledge does not only depend on the explicit information, but also on the implicit reasoning abilities. The ICI also affects cost, because it allows the performance system to use a simpler BCR, and reason about special details only in exceptional cases. Finally, the comprehensibility of a two-tiered representation must be carefully evaluated, since one of its implied goals is to state a clear and simple concept description in the BCR, and to account for special cases through a reasoning process. In fact, both the BCR and the ICI are parts of the concept description and they are used together in concept recognition. They influence each other. It is not necessarily true that if a BCR performs well with one ICI, then it also performs well with a different ICI.

For example, the experiments described in chapter 4 show that a BCR performed poorly with an empty ICI (i.e., when matching is strict), but it performed well with a flexible matching function. Therefore, learning a two-tiered concept description must take into consideration both the BCR and the ICI. In our method, this is done by evaluating a TT description (BCR+ICI) using the GDQ. Specifically, candidate concept descriptions are compared on the basis of how well they perform with flexible matching and with deductive ICI rules.

These criteria need to be combined into a single evaluation procedure that can be used to

compare different concept descriptions. A possible solution is to have an algebraic formula that, given the numeric evaluations of individual criteria, produces a number that represents their combined value. Examples of such approaches are multiplication, weighted sum, maximum/minimum, t-norm/t-conorm (e.g., Weber, 1983). Although these approaches are often appropriate, they have certain disadvantages. First, they usually combine a set of heterogeneous evaluations into a single number, and the meaning of this final number is hard to understand for a human expert. Second, they may force the system to evaluate all the criteria, even if it is sufficient to compare descriptions on the basis of one or two most important ones. The latter situation occurs when one description is much better than the other according to the most important criteria.

To overcome these problems, we use a combination of the *lexicographic evaluation functional* (LEF) and a *weighted evaluation functional* (WEF) for defining a measure involving all three above mentioned criteria.

3.3.2 Combining Individual Factors into a Single Preference Criterion

This subsection describes a LEF/WEF method for combining the simple description evaluation criteria into one general measure. We describe the *lexicographic evaluation functional* (LEF) first (Michalski, 83). The general description quality (GDQ) measure under LEF is defined as:

$$GDQ(description) = \langle(Accuracy,\tau_1), (Comprehensibility,\tau_2), (Cost,\tau_3)\rangle$$

where τ_1, τ_2, and τ_3 are tolerances.

In this evaluation scheme, the criteria are ordered according to their importance, and a tolerance threshold is associated with each criterion. If the difference of the evaluation of two expressions under a given criterion is less than the corresponding tolerance, the two descriptions are considered equivalent with respect to that criterion. The most important measure in the LEF is evaluated first, and the subsequent measure is evaluated only if the application of the previous one results in more than one candidate description.

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 the tolerance. It may be somewhat difficult to determine this threshold. If the tolerance is too small, chances of using the other criteria decrease. If the tolerance

is too large, the importance of the criterion might be underestimated. Furthermore, in the case of large tolerance, many descriptions might be equivalent under the LEF. To minimize this problem, the LEF measure is extended in the following way: LEF is first applied with relatively large tolerances, so that all the relevant criteria are taken into account. If the comparison results in a tie (i.e., there is more than one candidate description), a weighed evaluation functional (WEF) is used to combine the measures. The description with the highest WEF is selected. The weights for WEF are determined by the user.

3.3.3 The Role of The Typicality of The Examples

The accuracy is the major criterion for determining a concept description quality. Accuracy depends on the completeness and consistency of the description, as well as on the typicality of the events covered by the two parts of the description. In evaluating the accuracy of a two-tiered concept description, one has to take into account the fact that degree of confidence in the results of inference decreases from deduction to induction (Michalski, 1987). These requirements are met by introducing the notion of typicality (Rosch and Mervis, 1975). Completeness and consistency are made dependent on the typicality of the covered examples and on the way these examples are covered. We assume that an expert can provide typicality of examples at the time they are presented to the system responsible for building the initial description. The experts are usually quite good at determining the typicality of events in their area of expertise. In the case when the typicality is not provided, the system makes the standard assumption that it is equal for all examples.

Descriptions that cover many typical positive events are most preferred. Completeness is therefore proportional to the typicality of the events covered. Moreover, if negative events are covered, the consistency of the description is smaller if the typicality of the negative events covered is high.

Completeness and consistency of a TT description brings up additional requirements: a good description should cover the typical examples explicitly, and the non-typical ones implicitly. It is also preferred that the typical events are covered by the BCR, and non-typical, or exceptional events are covered by the ICI. In fact, the BCR is inductively learned from the events provided by user, and it is more reliable when the training events are typical. The ICI, on the contrary, is deductively obtained from the background knowledge, or from a human expert, and relies more on general and domain knowledge. Generally, the ICI is more reliable when dealing with the special or rare cases, since experts normally avoid explaining a large number of typical events. For these

reasons, a typical positive explicitly-covered event should contribute to completeness more than if it was implicitly-covered. And vice-versa, non-typical positive implicitly-covered events contribute to completeness less than if they were explicitly-covered. These assumptions are reflected by weights w_s, w_f, w_d, used in the definitions of completeness and consistency (section 3.3.4).

Furthermore, since ICI rules are obtained from background knowledge or from a humar expert, they are more reliable than the flexible matching function. Consequently, a positive D-covered event should contribute to completeness more than if it was F-covered. We may also observe that flexible matching is not very useful for low typicality exceptions. A similar argument holds for consistency.

3.3.4 General Description Quality Measure

This section defines the GDQ measure implemented in POSEIDON. First, we define the concepts of the description completeness, COMPT, and consistency, CONST, which take into consideration the typicality of examples. Such typicality-dependent completeness is defined as:

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

The typicality-dependent consistency is defined as :

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

where POS is the set of positive events covered by a two-tiered concept description and NEG is the set of negative events covered by a two-tiered concept description. Typicality(e) is a numeric degree of typicality of the event e specified by an expert. Furthermore, weights assigned to different types of coverage depend on certain thresholds to reflect the appropriateness of types of coverage for different kinds 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 t_1 and t_2 are thresholds, and $1 \geq t2 \geq t1 \geq 0$, $1 \geq w > 0$.

The *accuracy* of a description is defined in terms of COMPT and CONST:

Accuracy(description) = w_1*COMPT(description) +w_2*CONST(description)

where $w_1 + w_2 = 1$. The weights w_1 and w_2 reflect the expert's judgement about the relative importance of completeness and consistency for the given problem. The default value of both is 0.5. A measure of comprehensibility of a concept description is difficult to define. We will approximate it by measuring the representational complexity, 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. The 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 BCR is supposed to describe the general and easy-to-define meaning of the concept, while the ICI is mainly used to handle rare or exceptional events. As a consequence, the BCR should be easier to comprehend than the ICI, and thus v_1 should therefore be larger than v_2.

The cost of a description depends on two parts:

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

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

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

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 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 measure cost and evaluation cost.

With the exception of the weights, which can be determined experimentally, we have defined all three components of the GDQ. More details about the quality measure can be found in (Bergadano et al., 1988).

3.4. Learning by Maximizing The Concept Description Quality

As mentioned before, learning two-tiered concept descriptions is performed in two phases. In the first stage, a complete and consistent concept description is obtained from an inductive learning system. In our approach, we have relied on the AQ15 learning program (Michalski et al., 1986). This paper concentrates therefore on the second phase, i.e., on the improvement of the description obtained in the first phase with respect to its GDQ.

3.4.1 Search Heuristics

Applying the GDQ measure directly would be computationally expensive, because it would require the system to perform flexible matching for every newly generated description against the

whole set of training examples. To improve the efficiency, we have introduced a *double level* search. In the first level, a description to be modified in the second level (a node to be expanded) is determined using the GDQ. In the second level, an operator (SR, CR, RE or RC) to be applied to the description is selected using the *Potential Accuracy Improvement* heuristic (PAI). The value of PAI is defined as a function of the change in the coverage of positive and negative examples in the modified description. Specifically, we have:

$$PAI = \frac{\#P}{\#TP} \cdot \frac{\#N}{\#TN}$$

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

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

The above formula does not take into consideration the reduction in the complexity of the description due to the application of different operators. Specifically, removing a complex (a rule) reduces complexity more than removing a selector (a condition). To account for this, POSEIDON associates a higher weight with the operator CR (complex removal) than with the operator SR (selector removal), i.e., it prefers to apply CR than SR accordingly to some predefined weights.

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

If such an explanation is given, add the rules that make up the explanation to the ICI.

5. Update the GDQ value of the new node, by taking into account the added ICI rules.

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

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

In step 2, a search operator is chosen heuristically and applied to the description. The heuristics used are discussed in the previous section. Only one operator is applied at any given time.

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

In step 4, the "explainer" module is used in order to improve the description even further. The BCR description is extended or contracted by adding ICI rules. First, complex removal might uncover some positive examples, that were previously covered. In this case, new rules could be added to the ICI, that would allow the system to reason about such "special" positive examples, and explain why they should still be classified as instances of the concept under consideration.

On the other hand, selector removal might cause some negative examples to be covered. In this case, new ICI rules may have to be added to contract the BCR. Another issue concerning step 4 is whether an explanation should be required at all, since, in some cases, the chosen removal operator is not an appropriate one, and will lead to a very poor description. In this case it is not even worth to ask for an explanation, and search can continue in other directions.

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

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

$$PAI' = \frac{\#P}{\#TP} \cdot \frac{\#N}{\#TN}$$

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

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

3.4.2 Search Algorithm

The search procedure is described by the following algorithm:

1. Identify in the search tree the best description D (one with the highest GDQ). Initially, D is the complete and consistent description obtained in stage 1.
2. Apply to D the operator (from among CR_i , SR_{ij} , RE_{ij}, RC_{ij}) that potentially improves the GDQ of D the most, according to the Potential Accuracy Improvement (PAI) measure.

   CR_i. Remove the i-th complex from D.
   SR_{ij}: Remove the j-th selector from the i-th complex in D .
   RE_{ij}: Extend the referent of the j-th selector in the i-th complex in D.
   RC_{ij}: Contract the referent of the j-th selector in the i-th complex in D.
3. Compute the GDQ of the node obtained in step 2. If this GDQ is smaller than the GDQ of D, then proceed to step 1. As mentioned above, when computing the

   accuracy of a description, flexible matching is always used.
4. Ask for an explanation of

   (a) the positive examples that cease to be covered (positive exceptions)

   (b) the negative examples that become covered (negative exceptions)

values of GDQ give the explainer the sense of importance of the request for the search procedure.

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

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

3.4.3 An Abstract Example

An abstract example of the search process is given in Figure 3.6. The nodes contain BCR, ICI, and a graphical representation of the covered examples.

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

In the example, the accuracy is computed according to the formula discussed in Section 3.3.4, assuming the same typicality for all instances. The initial description is represented by node 1, and contains two rules (complexes). The rules cover two corresponding rectangular areas in the graphical representation, containing five positive examples out of eight, and one negative example out of five. The ICI extends this coverage by recognizing one more positive example. By eliminating condition (selector) s5 in the second complex node 3 is obtained. The accuracy of the

description is now improved since all positive examples are covered.

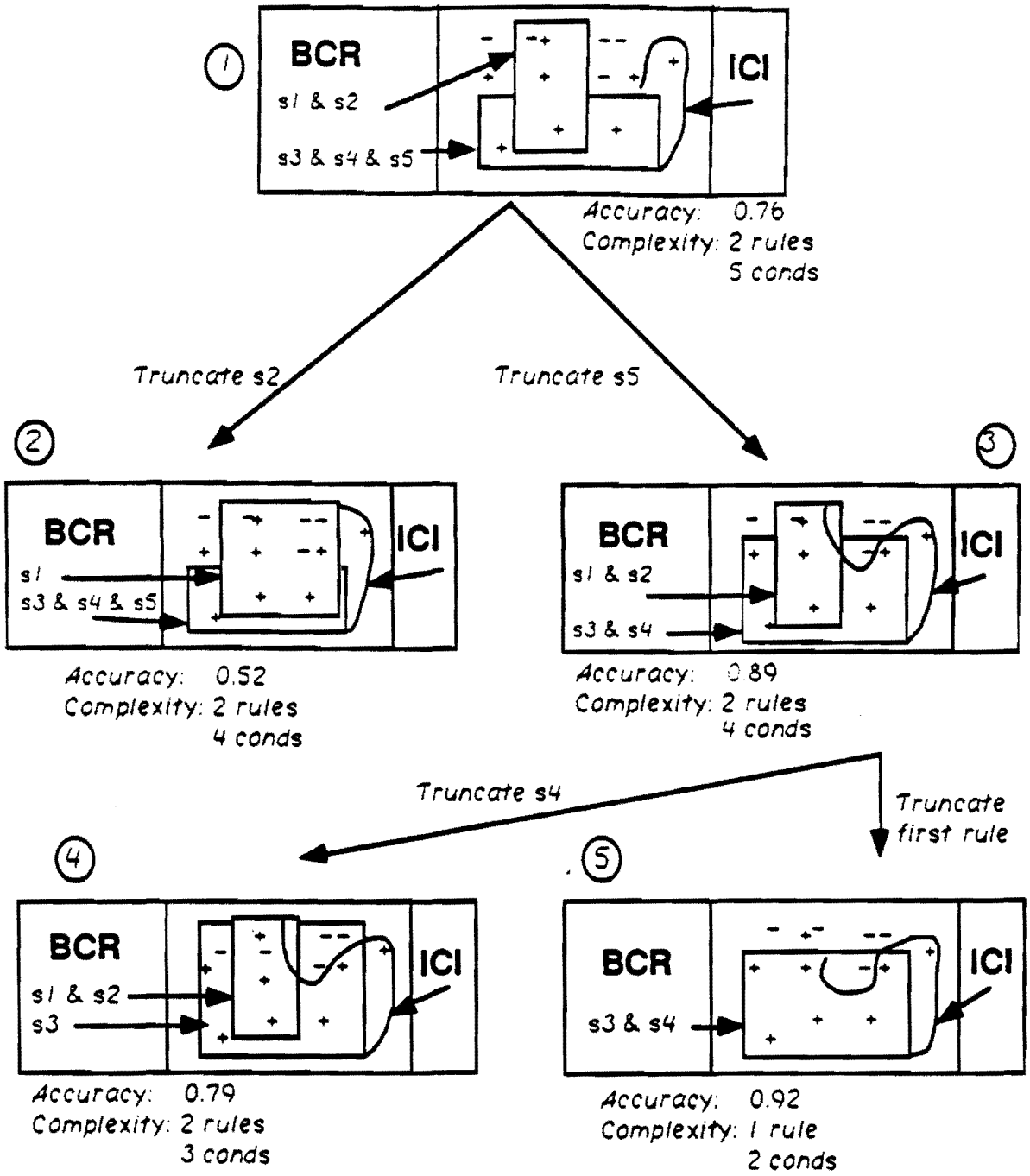


Figure 3.6. An illustration of the search process.

Finally, by removing the first complex node 5 is generated. It does not cover negative

examples any more, and is simpler. This node is then accepted as the improved description resulting from the search. The other nodes lead to inferior concept descriptions with respect to GDQ, and are discarded. The quality has been computed with $w_1 = w_2 = 0.5$. For clarity, the cost is omitted, and the simplicity of the ICI is not taken into account. The simplicity of the BCR is evaluated by the number of rules and conditions.

CHAPTER 4

EXPERIMENTAL RESULTS OF POSEIDON

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

4.1 Experimental Data

Labor management contracts

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

The raw data is divided into economic sectors. Each contract is described by a number of attributes. Since the attributes vary between economic sectors, all data used in the experiments were selected from one sector: personal and business services. This sector includes unions

representing hospital staff, teachers, university professors, social workers and certain classes of administrative personnel of various organizations. The data use multivalued attributes, and therefore VL_1 is an obvious choice as a description language.

The data describe contracts finalized in the second half of 1987 and the first half of 1988. Each contract is described by sixteen attributes, belonging to two main categories. One category concerns issues related to salaries, e.g., pay increases in each year of contract, the cost of living allowance, a stand-by pay, etc., and the second category concerns issues related to fringe benefits, e.g., different kinds of pension contributions, holidays, vacation, dental insurance, etc. Positive examples represent contracts that have been accepted by both parties. Negative examples represent contract proposals deemed unacceptable by one of the parties. The training set consisted of 18 positive and 9 negative examples of contracts; the testing set consisted of 19 positive and 11 negative examples. Here is a typical acceptable labor-management contract:

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

which is represented in VL_1 as :

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

U.S. Congress voting record

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

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

4.2 Description of Experiments and Illustration of Results

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

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

2. Determining the "top rule" description on the basis of the CC description, using the TRUNC method (Michalski et al., 1986). Such a description consists of a single rule selected from the CC description generated by the AQ15 program. It is the rule that covers the maximum number

of positive examples among all other rules in the CC description. The top rule 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 in the description (the so called t-weight and u-weight; see below). In the method, one top rule description was generated for the positive examples of the concept, and one for the negative examples, i.e., from a CC description of the negative examples. An instance was classified to the concept if it best matched the top rule description of positive examples, and was rejected if it matched the top rule description of the negative examples. If both descriptions were matched with roughly the same degree of match, than the instance was classified as "no match." Using such a description with flexible matching represents a simple but important version of the TT method (Michalski, 1990).

3. Optimizing the CC description using the POSEIDON SG-TRUNC procedure.

4. Testing all the above descriptions (CC, Top rule and Optimized TT rules) on the testing examples.

5. For comparison, the descriptions obtained by other methods, specifically, a simple exemplar-based learning approach and the decision tree learning algorithm ASSISTANT, were tested.

To illustrate the difference between the CC descriptions generated by AQ15, the *top rule* and the optimized descriptions created by POSEIDON, below is a sample of these descriptions in the labor management domain. The input data consisted of 18 positive and 9 negative examples of acceptable labor contracts. Figure 4.1 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.

The top rule (Figure 4.2) is obtained by selecting the rule with the largest t-weight from the CC description of each concept.

By applying the SG-TRUNC method to the CC description, the following optimized TT description was obtained (Figure 4.3).

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


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

Figure 4.1: The CC descriptions generated by AQ15

The BCR of the optimized descriptions are significantly simpler than the CC descriptions generated by AQ15. They seem to represent the most important characteristics of the labor

management contracts: a contract is acceptable when it offers a significant wage increase (the first two complexes in Figure 4.3), or it offers many holiday days, or the vacation is above average.

BCR:

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

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

ICI: Flexible matching

Figure 4.2: The top rules generated by TRUNC method

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

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

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

---

descriptions include many specific rules, leaving little space for the "no match" cases (3%), in which flexible matching could help. Secondly, the descriptions consisted of only disjoint rules, as the program was run using the "disjoint cover" parameter. In such a situation, the "multiple match" cases do not occur, and flexible matching cannot help.

Simple Exemplar-based Description

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

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

Table 4.1: Results of Experiment 1.

The above results are similar to those obtained in the previous experiment, which used an exemplar-based approach (Table 4.1). 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.

The third experiment (Table 4.3) tested the *top rule* descriptions determined from the above CC descriptions, as described above. As shown on Table 4.3, the performance of this rule on testing examples using flexible matching was comparable to that of the CC and factual descriptions generated in the previous experiments. A significant difference between this description and the previous two was in the complexity of the description (2 vs.11 vs. 27 rules in the labor management problem, and 2 vs. 10 vs. 51 rules in the congress voting problem). Each concept is described by one rule. Two concepts are involved in both domains, so each domain is described by two rules. It is quite interesting that so simple rules perform at the same level, or even better in some cases, than much more complex descriptions generated in previous experiments.

The fourth experiment (Table 4.4) tested the optimized descriptions generated by the method described in chapter 3 and implemented in POSEIDON (i.e., based the SG-TRUNC method). The descriptions were tested using flexible matching alone (*Flexible Match*) and in the combination with deductive matching (*Deductive Match*). For comparison, the performance of these descriptions using strict matching (*Strict Match*) was also evaluated (though this would be an impractical combination). As expected, these descriptions used with strict matching give relatively poor performance.

The optimized descriptions (BCR) combined with deductive matching (ICI) gave the best performance (90-92% correct). When used with only flexible matching, the performance is slightly lower, but still better than descriptions used in the previous experiments. This high performance seems to be primarily an effect of applying the description quality measure that uses all three types of matching (strict, flexible and deductive) during the learning process. This is one of new features of the system, not present in the earlier work (e.g., Bergadano and Giordana, 1989), where deductive matching is introduced only after the learning phase is completed.

The optimized descriptions are simpler than CC descriptions, although they include the ICI rules. They are more complex than the top rule descriptions, which are the simplest. For the Labor data problem, descriptions applied with deductive matching produced significantly higher performance than when used with only flexible matching (90 vs. 83%). For the Congress data problem, the performance was the same for the two matching methods. This is because no deductive rules were acquired in the problem.

BCR:

```
[wage_incr_yr1 > 4.5%] v
[wage_incr_yr2 > 3.0%] v
[#holidays > 9] v
[vacation period = above_average] ::>
                                    Acceptable  contract

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

ICI:

```
flexible matching and deductive matching using rules:

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

Figure 4.3: Optimized TT descriptions obtained by POSEIDON using the SG-TRUNC procedure

4.3 Results of Experiments

Four experiments have been performed. In each experiment, the number of events correctly and incorrectly covered by both descriptions was determined, as well as the number of events that were not covered by either concept. This was done both on the training set and on a testing set of examples (not previously seen by the system). The results of the three experiments are summarized in Tables 4.1 to 4.4. Correct (incorrect) states the percentage of the number of testing events that were correctly (incorrectly) classified. The *No_Match* counts examples that matched no rules. In each experiment, the same training and testing sets were used.

The first experiment (Table 4.1) used an exemplar-based concept description, i.e., a disjunction of the training events, also called a *factual* description.

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

Next parts of Experiment 1 tested simple forms of an exemplar-based learning method. The *1-Nearest Neighbor* row in the table lists results from applying the factual description with the matching method similar to the one described in (Kibler and Aha, 1987). A given testing example is matched against all training examples and the example that gives the best fit is found. The class of this "best fit" training example is assigned to the testing example. Our measure of fit differed slightly from the one described in (Kibler and Aha, 1987), as the latter uses the *maximum* function for evaluating a disjunction, while our flexible matching uses the *probabilistic sum* (section 3.2). We also tested other nearest neighbor methods (3-N and 5-N), in which instead of one, several best fit training examples are used to determine the class of the testing example (applying the majority rule).

The second experiment (Table 4.2) used concept descriptions generated by AQ15 without truncation. Such descriptions are consistent and complete (CC) with regard to the training examples, i.e., they classify all training examples 100% correct when using the strict matching method. The flexible matching method did not change this result. For the testing set, the number of correct classifications was relatively high (80-86%), the same for the strict and flexible matching methods. Flexible matching made no difference, probably due to two factors. Firstly, the CC

Complete and Consistent Description *(No truncation)*

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

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

Table 4.2: Results of Experiment 2.

The Top Rule Description *(the TRUNC method)*

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

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

Table 4.3: Results of Experiment 3.

Optimized Description (the SG-TRUNC method)

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

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

Table 4.4: Results of Experiment 4.

One may argue that deductive matching may be preferred over flexible matching. Examples that are correctly recognized through ICI deductive rules are *ipso facto* also explained in terms of domain knowledge. This cannot be said about examples correctly recognized on the basis of flexible matching, as this is due to a knowledge-independent distance measure. This feature may be particularly important from the viewpoint of the description comprehensibility. To reflect this, the quality measure assigns a higher score to a description with deductive matching than with flexible matching.

Table 4.5 summarizes the results of an experiment comparing the performance of descriptions generated by simple exemplar-based methods, the TT descriptions generated by POSEIDON, and pruned decision trees generated by ASSISTANT (a descendant of the Quinlan's ID3 program; Cestnik et al., 1987). ASSISTANT was applied with the same learning and training data, which were used in the experiments reported in Tables 4.1, 4.2, 4.3, and 4.4. The decision trees obtained by ASSISTANT were optimized using a tree-pruning mechanism (Cestnik et al., 1987).

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

As shown in Table 4.5, in this experiment, the optimized TT description both performed better and was simpler than the best exemplar-based description and the pruned decision tree. The "top rule" description was much simpler than any other description, but performed worse than the optimized description and the decision tree. Depending on the desired trade-off, the "top rule" or the optimized description can be taken as the BCR of the concept being defined.

The above experiments show that the learning method implemented in POSEIDON produces descriptions that are both simple and well performing on the testing data. The complete meaning of the concept defined by such a description depends on the BCR, the optimized description generated through learning, and the ICI (which consists of the flexible matching procedure, defined a-priori, and a set of deductive rules, formulated by the expert). The expert defines the rules on the basis of misclassified examples, which are determined by the system during the learning process.

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

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

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

Table 4.5. Summary of the results of testing descriptions generated by different methods

Chapter 5

THE FLEXIBLE CONCEPT LEARNING SYSTEM: FCLS

This chapter describes an alternative concept representation formalism which is based on the idea of the two-tiered concept representation, and its associated concept learning algorithm. The learning approach reported in this chapter has been implemented in the Flexible Concept Learning System: (FCLS) in Common Lisp on a Sun 3 workstation. FCLS has been run on various domains from artificial to natural, and the description of the experimental results is deferred to the next chapter.

5.1 Problems

In chapter 1, it was stated that the main goal of the research reported in this thesis is to develop approaches to represent and learn flexible concepts. The motivations of the research were also discussed in chapter 1. This section discusses some specific problems in representing and learning flexible concepts. The rest of the chapter describes an approach that is proposed to solve these problems.

As mentioned in chapter 1, flexible concepts usually have imprecise and irregular boundaries, their examples have differing degrees of typicality, and they often include some exceptional cases. In traditional logic type representations, each disjunct can only describe a concept with a rectangular boundary. Therefore, a concept with irregular boundaries has to be represented by a set of disjuncts, among which many are small (Holte et. al, 1989). A small

disjunct covers only few examples. Figure 5.1(a) is a concept with an irregular boundary. Figure 5.1(b) shows how the concept shown in Figure 5.1(a) is approximately represented by 4 disjuncts. Three of the four disjuncts in figure 5.1(b) are small.

(a) (b)

Figure 5.1: the methods for representing a concept with irregular boundary

One of the difficulties with an inductive learning system is to learn highly disjunctive concepts (Rendell, 1988). What makes a concept highly disjunctive is an inappropriate representation formalism or an insufficient representation vocabulary. By representation formalisms, I refer to the formalisms such as disjunctive normal forms, decision trees, exemplar models, and perceptrons. By representation vocabulary, I refer to the terms or concepts used in a specific representation formalism. For example, a large number of disjuncts are needed to approximate a concept that is linearly separable. Ideally, every concept could be represented by one disjunct if the representation (both formalism and vocabulary) is adequate enough, because there must be a unifying principle behind every concept. The term "disjunct" is used here in a broader sense than in a DNF. For example, a linear threshold unit could be viewed as a disjunct, and a disjunct of a DNF with a similarity measure is called a disjunct. Accordingly, two approaches: enriching the representation vocabulary (Constructive induction) and selecting an appropriate representation formalism (using hybrid representation formalism), were proposed to overcome the difficulty of learning highly disjunctive concepts (Rendell, 1988; Utgoff, 1988).

Generally speaking, constructive induction is applied to merge distant disjuncts, while hybrid representation formalism is applied to merge proximate disjuncts. Distant disjuncts are not syntactically similar to each other. Therefore they cannot merge unless some new terms which can reflect the semantic similarity of the distant disjuncts are constructed. All proximate disjuncts are

syntactically similar, and form a cluster (peak) which can be packed into one disjunct by some hybrid representations. Different clusters may merge through constructive induction. The main concern of this chapter is the hybrid representation that merges proximate disjuncts.

The problem of small disjuncts was first recognized by Michalski et al. (1986). According to the experimental results reported in (Michalski et al., 1986), when some small disjuncts were removed from a complete and consistent concept description, the performance of the description was slightly improved. Later, Holte et al. (1989) experimentally proved that small disjuncts are much more error prone than large ones. To solve the problem of small disjuncts, several approaches have been proposed. The simplest approach is to remove all small disjuncts from a description, e.g., decision tree pruning (Quinlan, 1987). In some domains (especially noisy domains), this approach improved accuracy of concept descriptions. The shortcomings of the approach are obvious. First, the description generated in this approach loses the coverage of the examples covered by removed small disjuncts. When a concept is highly disjunctive, this approach seriously degrades the performance of the generated descriptions. Second, in a dynamic environment, concepts drift from time to time, so a small disjunct may gradually become large. By removing small disjuncts, a learning system loses the capability to adapt such a dynamic environment.

Michalski et al. (1986) removed all small disjuncts, then applied a flexible matching function to the remaining large disjuncts to recover some of the loss of coverage. In this way, only the loss of coverage from the small disjuncts that are close to some large disjuncts might be recovered. To recover the coverage lost from the small disjuncts that are far from all large disjuncts, the POSEIDON system described in chapter 3 of the thesis applied a more sophisticated matching procedure in which deduction is allowed to match an example with concept descriptions. The problem with this approach is the difficulty in acquiring the rules for deductive inference.

Holte et al. (1989) suggested different biases should be used for large and small disjuncts, with a maximum generality bias for large disjuncts and a selective specificity bias for small disjuncts. The hybrid representation described in next section provides an alternative approach to handle the problem of small disjuncts. Small disjuncts may occur in two cases. First, when a concept has irregular and imprecise boundaries, small disjuncts are generated to describe the boundary examples. Second, flexible concepts often include some exceptions, small disjuncts are generated to describe these exceptions. In the approach presented in this chapter, small disjuncts that cover boundary examples may merge under the hybrid representation introduced in next section, while the small disjuncts that cover exceptions are replaced by exemplars.

As discussed before, each flexible concept often has a central tendency that is described all relevant properties of the concept. Some of the relevant properties are more important than the others. An instance of the concept may only possess some of the relevant properties instead of all the relevant properties. An event with more relevant properties is more typical than one with fewer relevant properties. An event with more important properties is more typical than one with less important properties. In other words, all events that possess all the relevant properties form the center of the concept. The events close to the center are more typical than the events further from the center. The boundary of the concept with such a central tendency is often imprecise. The probability of an event's belonging to the concept decreases with its distance from the center. The classification of boundary events of the concept is highly uncertain.

The description of such a concepts should be graded so that the events near the center have higher values of match with the description than the events near the boundary. The problems are:

1. How to represent and learn the central tendency.
2. How to represent and decide the boundary.
3. How to represent and learn the importance of a relevant property.

The rest of the chapter address these problems.

5.2. Concept Representation

This section introduces the hybrid concept representation used in FCLS. FCLS represents a concept as a disjunction of extended complexes and a set of exemplars. An extended complex consists of a base complex, a set of weights, a threshold, and a similarity measure. In addition, each extended complex has a certainty value. A base complex is equivalent to a disjunct represented in VL_1 (Michalski, 1983) as a conjunction of conditions called selectors and represents a central tendency of a concept. The weights are the degree of necessity (importance) of conditions, while the threshold defines the boundary of an extended complex. The similarity measure determines the degree of fit between an event and a base complex. The similarity between an event and an exemplar is also decided by the similarity measure. The exemplars are a set of selected examples. An extended complex describes a cluster (peak) of a concept whose central tendency is described by the base complex of the extended complex. The exemplars represent exceptions.

In this representation, each extended complex is two-tiered, its base complex is the first tier,

its weights, threshold and the similarity measure constitute the second tier. Therefore, this hybrid representation can be viewed as a simple form of the two-tiered concept representation. It is simple because it does not include any inference rules. Following subsections describe each of the components of the representation in detail.

5.2.1 Base Complex

A base complex is represented in the same way as a complex in POSEIDON. To be complete, I repeat the part of the definition of a base complex. A base complex is represented as a complex by the attribute based Logic System VL_1 (Michalski, 1983). A complex in VL_1 is a conjunction of selectors. A selector is of the form:

$$[L \# R]$$

where the attribute L is called the *referee* and R is called the *referent*, which is a set of values from the domain of L. The symbol # denotes one of the relational symbols $=, <, >, \leq, \geq, \neq$. Complexes and selectors correspond to disjuncts and conjuncts of a disjunctive normal form, respectively. In the current implementation of FCLS, only nominal and linear attributes are allowed.

5.2.2 Syntactic Similarity Measure

The syntactic similarity measure (SSM) measures the degree of fit between an event and a base complex. The specific SSM used in our current implementation maps an event from the set E and a base complex from the set C to a real value between 0 and 1, which is the degree of fit of the event to the complex.

$$\text{SSM: } E \times C \text{ --> } [0..1]$$

The SSM of an event *e* and a base complex cpx is defined by a normalized distance measure DIS as follows:

$$\text{SSM}(e, cpx) = 1 - \frac{\text{DIS}(e, cpx)}{\text{MAXDIS}(cpx)}$$

where MAXDIS (cpx) is the maximum distance between events in the set E and the complex cpx. DIS(e, cpx) is defined as a weighted sum of the distances between the event *e* and all selectors of

the complex cpx:

$$DIS(e, cpx) = \sum W_i * SELDIS(e, sel_i)$$

where W_i is the weight which reflects the necessity of the selector sel_i in the complex. Weights are calculated during learning. Weight learning methods will be described in section 5.4.4. SELDIS(e, sel_i) is the distance between the event *e* and the selector sel_i. Suppose that the selector sel is [x = aj_1, v...v, aj_m]. SELDIS(e, sel) is then defined as

$$SELDIS(e, sel) = \begin{cases} 0 & \text{if x is nominal and e si covered by sel} \\ 1 & \text{if x is nomianl and e is not covered by sel} \\ \dfrac{lindis(a_k, sel)}{Max} & \text{if x is linear} \end{cases}$$

where $Max = max_{i=1,...,n} \{dis(a_i, sel)\}$,

$lindis(a_k, sel) = min_{i=j1,...jm} (|i - k|)$

It is assumed that the domain of the selector x is the ordered list (a_1, a_2,...,a_n), and a_k is the value of x of the event *e*. For example, if the domain of x is [0 .. 10], the value of x for the event *e* is 4 and the selector *sel* is [x = 7 .. 9], then $SELDIS(e, sel) = \frac{7 - 4}{7 - 0} = 3/7$.

5.2.3 Exemplars

In section 5.1, we discussed the problem of small disjuncts. In the representation proposed in this section, the small disjuncts that cover exceptions are replaced by exemplars. In this subsection, I shall discuss some of the advantages of replacing small disjuncts with exemplars, and show some empirical results.

A disjunct that covers a small area in event space often includes many more conditions (conjuncts) than a disjunct that covers a large area. It is more time consuming and error prone in learning such small disjuncts than large ones in an inductive learning system that performs general-to-specfic search. In such an inductive learning system, the search starts from the most general disjunct, and gradually adds more conditions to specialize the most general disjunct until a disjunct that satisfies some criteria is found. It is possible to select a wrong condition to add to the current disjunct in each step of the specialization process. A disjunct including more conditions needs

more steps to generate, so it is more error prone and time consuming. Many inductive learning systems, such as ID3, PL1 and CN2, FCLS, perform a general-to-specific search.

In contrast, an exemplar-based approach may work better for a small disjunct than a large one. As mentioned above a small disjunct includes many conditions, that is, many attributes are relevant and few are irrelevant. An exemplar-based approach is sensitive to the number of relevant attributes. Usually, the fewer irrelevant attributes, the better the approach works.

Two experiments have been conducted to confirm the above arguments. The domain consists of 8 nominal attributes, each of which has four values. Two concepts, positive and negative, are involved in the experiments. In the first experiment, both positive and negative concepts are described by one disjunct. The number of attributes involved in the two disjuncts is varied, the more attributes are involved, the smaller the disjuncts become. In the second experiment, only the number of attributes involved in the positive concept is varied, the negative concept is simply the complement of the positive concept. Two learning systems, an inductive learning system and an exemplar-based learning system, were applied to learn these two concepts. The learned concepts were tested on the test sets. The results are shown in Figure 5.2. The inductive learning system used in the experiments is similar to CN2 and was implemented as a part in FCLS, and the exemplar-based learning system used is similar to the Growth algorithm (Kibler and Aha, 1987). The ratio of the number of training examples of the positive class and the number of the total examples of the positive class is kept constant. Figure 5.2(a) presents the results of the first experiment. Figure 5.2(b) shows the results of the second experiment. In both experiments, FCLS was given a value 1 for *MAXSTAR* which is a parameter that limits the maximum search space. When *MAXSTAR* is equal to 1, the search is equivalent to hill climbing.

In both Figure 5.2(a) and Figure 5.2(b), the accuracy of the descriptions generated from the exemplar-based learning method sharply increases when the number of relevant attributes increases. The results are consistent with the arguments that the fewer the number of irrelevant attributes, the better the exemplar-based approach works. A small disjunct often has more relevant attributes, so the exemplar-based approach may work better for small disjuncts than large one. In Figure 5.2(b), the accuracy of induced descriptions degrades very fast. This contrasts with Figure 5.2(a), where the accuracy of induced descriptions only slightly degrades. This difference can be explained as follows. In the first experiment, the number of attributes for both concepts increases simultaneously. When more attributes are involved in these two concepts, only a small part of the whole event space is covered by the two concepts. Therefore, an overgeneralized disjunct does not cause too much error.

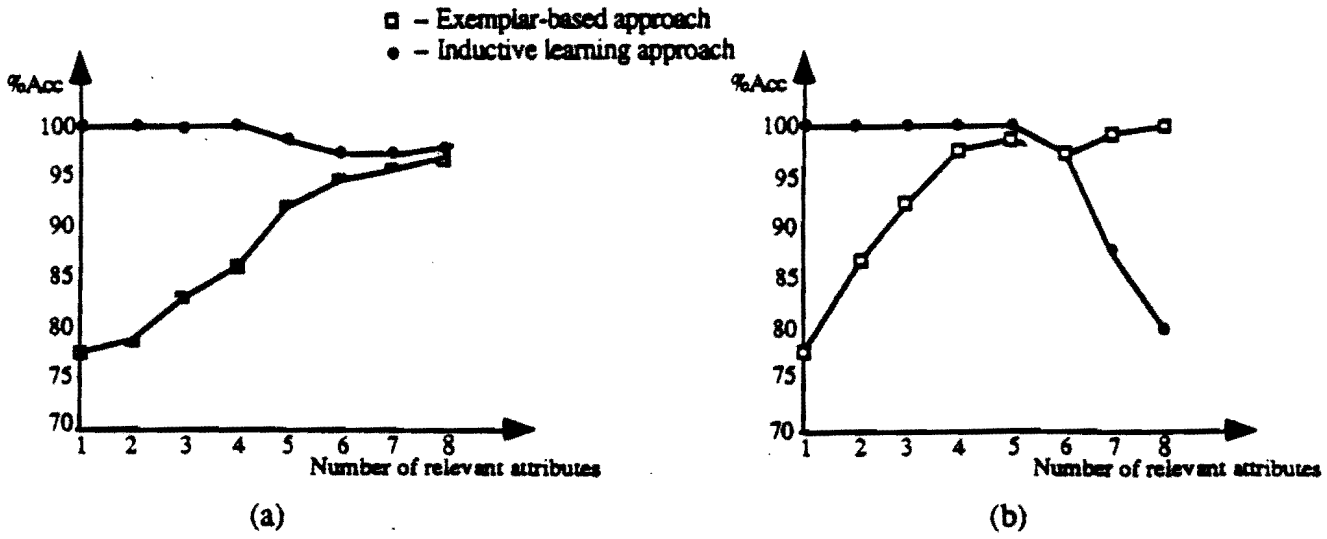


Figure 5.2. Variation of accuracy with the number of relevant attributes

Another advantage of replacing small disjuncts by exemplars involves a dynamic environment in which the meaning of a concept may drift from time to time, possibly resulting in a small disjunct gradually becoming larger. A small disjunct is generalized from a small set of examples. When a description is generalized from examples, some information is lost, especially when the description is generalized from a small set of examples. If exemplars are stored, instead of small disjuncts, as descriptions, the information lost may be less. If more examples come later, these exemplars together with new examples can be generalized to a large disjunct.

An exemplar is a special case of a base complex and is represented in the same way as a base complex. An exemplar is one of the most specific base complexes, and has a selector with only one value for each attribute. In the hybrid representation reported in this section, an exemplar is also viewed as an extended complex whose threshold is 1. Each selector of an exemplar has a weight, the method to compute the weight will be described in section 5.4.4. The similarity measure discussed in the previous section (section 5.2.2) is also applied to measure the degree of fit between an event and an exemplar.

5.2.4 Weights

Each selector of a complex is associated with a weight which reflects the degree of necessity of the selector under the complex. Its value ranges from 0 to ∞. When the weight of a selector of a complex is equal to 0, the selector is totally irrelevant, and plays no role in the complex. This occurs very seldom, because irrelevant attributes are often dropped during learning. When the weight of a selector of a complex is equal to ∞, the selector is a necessary condition of the complex and has to be satisfied. Except 0 and ∞, any other value of a selector weight reflects the relative importance of the selector in comparison with the other selector in the same complex. For example, if the weight of all selectors of a complex are equal, all the selectors are equally important regardless the value of the weights of these selectors. The importance of selectors that are in different complexes is not comparable. Weights are computed during learning. Two weight learning methods will be described in section 5.4.4.

5.2.5 Threshold

In addition to weights, each extended complex is associated with a threshold that is a real number between 0 and 1. The threshold of an extended complex defines the boundary of the complex and is used to determine the classification of events based on the degree of fit between events and the complex. An event is covered by an extended complex if its degree of fit to the complex is not smaller than the threshold.

A threshold of an extended complex partially determines the distribution of the meaning of the extended complex between its base complex and the similarity measure. When the threshold of an extended complex is equal to 1, the extended complex is equivalent to its base complex. The extended complex with the threshold 0 covers the whole event space. The smaller a threshold of an extended complex, the more important role the similarity measure plays in the extended complex. The weight reflects necessity of a condition, whereas the threshold can be viewed as the definition of the sufficiency of a group of conditions. The threshold of an extended complex is adjusted so that the optimum performance of the complex regarding training examples is achieved during learning.

At first look, it seems the boundary of an extended complex defined by a threshold is precise. In fact, it is not precise for two reasons. First, an extended complex is graded, and all events have a degree of fit to the complex. So the classification of the events whose degree of fit is close to the threshold is thought highly uncertain and imprecise. Second, the boundary defined by

a threshold is further softened by applying the flexible classification method which will be introduced in next section (section 5.3).

5.2.6 Certainty

Each extended complex has a certainty value which is defined as the inverse of the generality of the complex. The certainty of an extended complex is used in classifying the unseen events that match more than one concept descriptions, such events are called multiple match events. The certainty of an extended complex cpx C(cpx) is defined as the ratio of the logarithm of the number of positive examples covered by cpx minus the number of negative examples covered by cpx and the logarithm of the total event space covered by cpx.

$$C(cpx) = \frac{1 + \log_2 (PC(cpx) - NC(cpx))}{1 + \log_2 SC(cpx)}$$

where PC(cpx) and NC(cpx) are the number of the positive examples covered and the number of the negative examples covered by cpx, respectively. SC(cpx) is the event space covered by cpx. The more space cpx covers, the greater the inductive gain (Rendell, 1986), and the less the certainty.

One of the reasons that small disjuncts are error prone is that they are too general. To correct this, Holte et al. (1989) proposed that a selective specificity should be used for small disjuncts. It is hoped that the use of certainty can lessen the problem of small disjuncts, because a overgeneralized complex has smaller certainty so that the less generalized complex is preferred if an event matches two complexes. An exemplar has the highest certainty 1.

5.2.7 Examples

To illustrate the idea of the hybrid representation described in the previous subsections, let us consider a simple imaginary concept "R-ball" (Michalski, 1990). The meaning of the concept R-ball is defined as three disjuncts:

(SHAPE = round) & (BOUNCES = yes) or
(SHAPE = round) & (SIZE = medium v large) or
(BOUNCES = yes) & (SIZE = medium v large)

By using the hybrid representation introduced above, these three disjuncts merge into one extended complex:

[SHAPE = round]: 1
[BOUNCES = yes]: 1
[SIZE = medium v large]: 1
Threshold = $\frac{2}{3}$ = 0.67

The number following a selector is the weight of the selector. The base complex:

[SHAPE = round] & [BOUNCES = yes] & [SIZE = medium v large]

represents the central tendency of the concept R-ball, all of the three selectors are equally important. The meaning defined by the extended complex is that an object that satisfies any two or more of the three selectors is a R-ball, otherwise it is not a R-ball. Furthermore, it tells that balls that satisfy all of the three selectors are typical R-balls, while those which only satisfy two of the three selectors are less typical R-balls.

Now suppose the meaning of the concept R-ball changes a little, and all R-balls must be round. The new meaning of the concept R-ball is defined by two disjuncts:

(SHAPE = round) & (BOUNCES = yes) or
(SHAPE = round) & (SIZE = medium v large)

These two disjuncts are combined into one extended complex:

[SHAPE = round]: ∞
[BOUNCES = yes]: 1
[SIZE = medium v large]: 1
Threshold = $\frac{1}{2}$ = 0.5

In this extended complex, the selector [SHAPE = round] is necessary, and must be satisfied by all R-balls. The other two selectors are not necessary, and one of them must be satisfied by a R-ball.

Suppose that the attribute *SIZE* is linear and the order of its values is small, medium and large. The extended complex representing R-ball becomes:

[SHAPE = round]: ∞

[BOUNCES = yes]: 1
[SIZE = large]: 1
Threshold = $\frac{1}{4}$ = 0.25

Under this extended complex, the R-ball whose size is large is more typical than the R-ball whose size is medium. For example, let us consider the two R-balls Rb1 (round yes large) and Rb2 (round yes medium). The degree of fit of Rb1 and Rb2 to the extended complex is 1 and 3/4, respectively, so Rb1 is more typical than Rb2.

5.2.8. Discussions

In the sections 5.2.2 to 5.2.6, a hybrid concept representation based on the idea of the two-tiered concept representation was proposed to represent flexible concept. In the previous sections, it is simply discussed that how this hybrid representation handles the problems that flexible concepts may cause, such as imprecise and irregular boundaries, highly disjunctive concepts, small disjuncts, and events with different typicality. This subsection further discusses how these problems are tackled in the proposed representation.

Before further discussion, let us first see how a flexible concept is represented in the hybrid representation. In Figure 5.3, the area inside the irregular shape is the area covered by a flexible concept which is represented by an extended complex (the oval in the figure) and 7 exemplars (+'s in the figure). The rectangle inside the oval is the base complex that captures the central tendency of the concept.

In section 5.2.7, I showed some simple examples of the representation. In this section, some more complicated examples are given.

Let us first consider the simplest case: all attributes are nominal and all weights of selectors of a complex are equal to 1. In this case, the extended complex describes a concept that has the form "at least k of n conditions are satisfied". Such an extended complex consists of n selectors, weights of all selectors are equal to 1 and its threshold is k/n. This kind of extended complexes is similar to criteria tables which are useful as a knowledge representation for expert systems, especially in medical domains (Spackman, 1988). Such an extended complex merges n!/(k! * (n - k)!) disjuncts.

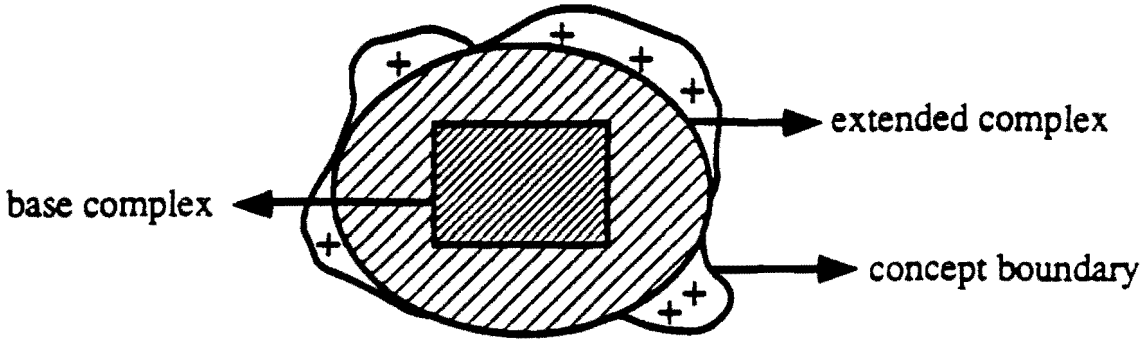


Figure 5.3: how a flexible concept is represented by the hybrid representation

The following examples was used to test the proposed representation and its associated learning algorithm, the results from this example will be reported in section 6.2.1. In this example, the target concept is defined by 7 conditions, of which 5 or more should be satisfied. This concept is described by the following extended complex:

$$[x_1 = 0 \text{ v } 1]: 1$$
$$[x_2 = 0 \text{ v } 1]: 1$$
$$[x_3 = 0 \text{ v } 1]: 1$$
$$[x_4 = 0 \text{ v } 1]: 1$$
$$[x_5 = 0 \text{ v } 1]: 1$$
$$[x_6 = 0 \text{ v } 1]: 1$$
$$[x_7 = 0 \text{ v } 1]: 1$$
$$\text{Threshold} = \frac{5}{7}$$

This target concept has to be represented by 21 disjuncts in a disjunctive normal form, so this extended complex merges 21 disjuncts.

Now a slightly more complicated case is given: all attributes are nominal and the weight of a selector may have one of two values: $+\infty$ and 1. In this case, an extended complex includes two kinds of selectors: necessary selectors and unnecessary selectors, and all unnecessary selectors are equally important. The weights of necessary selectors are $+\infty$, whereas the weights of unnecessary selectors are equal to 1. Thus, the concept described by an extended complex has the form "the m conditions $x_1,...,x_m$ must be satisfied, and at least k of n - m conditions $x_{m+1},..., x_n$ has to be

satisfied". $x_1,...,x_m$ are necessary conditions and $x_{m+1},..., x_n$ are unnecessary conditions.

A more complicated case is that all attributes are nominal and weights of selectors can be any real value between 0 and $+\infty$. In this case, an extended complex can represent more complicated concepts. For example, the following two disjuncts:

$$[x1 = 1] \text{ or}$$
$$[x2 = 1] \,\&\, [x3 = 1]$$

merge into the following extended complex:

$$[x1 = 1]: 2$$
$$[x2 = 1] : 1$$
$$[x3 = 1] : 1$$
$$\text{Threshold} = \frac{1}{2}$$

The most complicated case is that all attributes are linear and weights of selectors can be any value between 0 and $+\infty$. Let us consider a domain which has eight linear attributes x_n ($1 \le n \le 8$) and each attribute has four values 0, 1, 2 and 3. The target concept is described by the following extended complex:

$$[x_1 = 0 \text{ v } 1]: 2$$
$$[x_2 = 0 \text{ v } 1]: 2$$
$$[x_3 = 0 \text{ v } 1]: 1$$
$$[x_4 = 0 \text{ v } 1]: 1$$
$$[x_5 = 0 \text{ v } 1]: 1$$
$$[x_6 = 0 \text{ v } 1]: 1$$
$$\text{Threshold} = \frac{5}{8}$$

This domain was one of the domains used in the experiments reported in next chapter. The attributes are linear, so the concept is very complicated if it is represented as a disjunctive normal form.

Another interesting point of the hybrid representation is that conjunctive constructive induction can be applied to merge disjuncts. In a logic type representation, only disjunctive constructive induction can be applied to merge disjuncts. I show this by an example. Assume that a concept C consists of two disjuncts:

$$[x1 = 1] \,\&\, [x2 = 1] \,\&\, [x3 = 1] \text{ or}$$
$$[x1 = 1] \,\&\, [x4 = 1] \,\&\, [x5 = 1]$$

These two disjuncts cannot be combined into one extended complex unless the other four disjuncts:

$$[x1 = 1] \,\&\, [x2 = 1] \,\&\, [x4 = 1]$$
$$[x1 = 1] \,\&\, [x2 = 1] \,\&\, [x5 = 1]$$
$$[x1 = 1] \,\&\, [x3 = 1] \,\&\, [x4 = 1]$$
$$[x1 = 1] \,\&\, [x3 = 1] \,\&\, [x5 = 1]$$

are also included in the concept C. But after applying the following conjunctive constructive induction rules,

$$[x6 = 1] \text{ <-- } [x2 = 1] \,\&\, [x3 = 1]$$
$$[x7 = 1] \text{ <-- } [x4 = 1] \,\&\, [x5 = 1]$$

two new attributes x6 and x7 are constructed and the two disjuncts merge into the following extended complex.

$$[x1 = 1]: +\infty$$
$$[x6 = 1]: 1$$
$$[x7 = 1]: 1$$
$$\text{Threshold} = \frac{1}{2}$$

The hybrid representation is flexible and can be dynamically adjusted during learning by adjusting the distribution of extended complexes between their base complexes and the similarity measure, and the distribution of a concept meaning between general descriptions and exemplars according to the concepts to be learned. The distribution of an extended complex is adjusted by changing its threshold, and the adjusting of the distribution between general descriptions and exemplars is controlled by two user defined parameters which will be described in next section. By changing the weights of the selectors, one can change the shape of an extended complex. By changing the threshold, one can change the size of an extended complex.

At one extreme of the distribution, an extended complex is equivalent to a disjunct in DNF. In this case, the threshold of the extended complex is 1. At the other extreme, an extended can be viewed as an best exemplar model (or prototype). At this extreme, the base complex of the extended complex is diminished to a single event which is a center of a cluster. Between these two

extremes, an extended complex is similar to a description in probabilistic view. During learning, the optimum distribution is decided based on characteristics of the concepts to be learned.

5.3 Concept Recognition

During the classification of new events, each of the new events is matched with concep: descriptions which are represented as a set of extended complexes and a set of exemplars to see which concept descriptions cover the event. During learning, to evaluate intermediate hypotheses, each of the training examples is matched with all intermediate hypotheses. This section describes two classification (matching) methods: strict and flexible. Both methods match an event with concept descriptions, and the result of a match between an event and a concept description is based on the result of the match between the event and the extended complexes and the exemplars of the description. In both methods, an event is covered by a concept description if the event is covered by one of the extended complexes of the description or one of the exemplars of the description.

Because exemplars in the representation are treated in the same way as extended complexes, these two classification methods can be applied to match an event with both extended complexes and exemplars. In the rest of the section, we shall not distinguish exemplars from complexes, and only describe how these two classification methods are applied to match an event with an extended complex.

5.3.1 The Strict Classification Method

The strict classification method is very straightforward. In this method, an event is covered by an extend complex if the degree of fit of the event to the base complex of the extended complex is not smaller than the threshold of the extended complex. The degree of fit between an event e and a base complex cpx is the result of the syntactic similarity measure SSM(e, cpx). The examples covered by applying this method are called strictly covered examples.

The coverage of an extended complex computed by applying the strict classification method is independent of other extended complexes, so the strict method may create many no match events that are covered by no concept descriptions, and multiple match events that are covered by more than one concept description. No match events cannot be classified, while multiple match events cannot be uniquely classified. The flexible classification method is proposed to classify these no

match and multiple match events.

5.3.2 The Flexible Classification Method

A multiple match event is covered by two or more concept descriptions, that is, its degree of fit to two or more complexes is larger than or equal to the thresholds of these complexes. The event is covered by several complexes, but it may be covered in different degree of fit by these complexes. In the flexible classification method, a multiple match event is classified based on the result of the degree of fit between the multiple match event and the base complexes of extended complexes weighted by the certainties of extended complexes. The degree of fit of a no match event to all complexes is smaller than their thresholds, but some are close to thresholds, the others are far from thresholds. In the flexible classification method, a no match event is classified based on the difference between the degree of fit of no match events and the thresholds of extended complexes.

Let us first define Flexible Degree of Fit (FDF) between an event and an extended complexes. The FDF of a multiple match event e to an extended complex cpx is the normalized degree of fit between e and the base complex of cpx weighted by the certainty of cpx.

$$FDF(e, cpx) = \begin{cases} Certainty(cpx) & th(cpx) = 1 \\ Certainty(cpx) * \dfrac{SSM(e,cpx) - th(cpx)}{1 - th(cpx)} & \text{otherwise} \end{cases}$$

Where Certainty(cpx) is the certainty of cpx, th(cpx) is the threshold of cpx. Because e is strictly covered by cpx, SSM (e,cpx) ≥ th(cpx). Thus, the FDF of a multiple match event to an extended complex that strictly covers the no match event is between 0 and Certainty(cpx). The FDF of a multiple match event to an extended complex increases with the increases of the certainty of the complex and the normalized degree of fit between the event and the complex.

The FDF of a no match event e to an extended complex cpx is the normalized difference between the degree of fit of the event and the threshold (boundary) of the extended complex.

$$FDF(e, cpx) = \frac{SSM(e,cpx) - th(cpx)}{th(cpx)}$$

Because e is not strictly covered by cpx, SSM(e, cpx) < th(cpx) and the FDF of a no match event

to an extended complex is between 0 (exclusive) and -1.

In the flexible classification method, an event is covered by an extended complex if the Flexible Degree of Fit (FDF) of the event to the extended complex is the largest among all extended complexes. Such events are called flexibly covered by the complex. The flexible method does not change the classifications of the events that are uniquely classified by the strict method. The coverage of an extended complex decided by flexible classification method is dependent on other extended complexes. After some new extended complexes are generated, the coverage of the existing extended complexes may change.

In Figure 5.4, the two circles represent two different extended complexes cpx1 and cpx2, respectively. Each extended complex describes a different class. e represents an event. In the figure, the events e1 to e6 are strictly covered by cpx1, e7 to e11 are strictly covered by cpx2. e6 and e11 are multiple match events and e12 and e13 are no match events. Using the flexible classification method, e6 and e13 are classified to cpx1, and e11 and e12 are classified to cpx2.

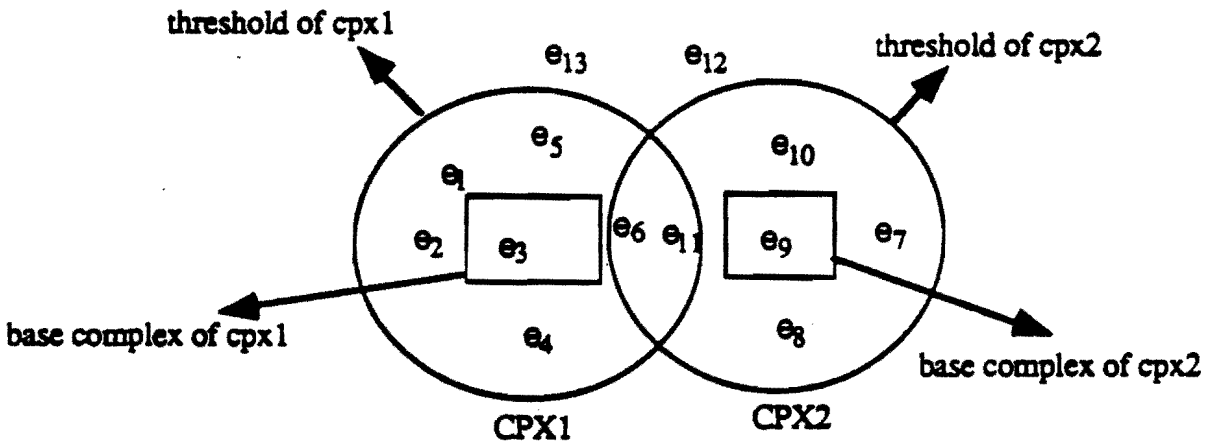


Figure 5.4: The illustration of how the strict and the flexible classification work

In almost all conducted experiments, many no match and multiple match events are correctly classified by the flexible method, the classification accuracies have been improved. The experimental results reported in next chapter show this improvement.

The strict method is applied to decide the coverage of an extended complex during learning, because the flexible coverage of a complex dynamically changes with the changes of other

extended complexes. The flexible coverage of an extended complex may decrease, after some new extended complexes are generalized. So a currently 'good' complex may become worse after some new complexes are generated.

The strict coverage of an exemplar is the exemplar itself, so the strict method is not meaningful for exemplars. No generalization is performed by the strict method. In contrast, the flexible method performs some generalization during classification.

5.3.3 Categories of Examples

For each extended complex, all training examples can be divided into five categories based on the result of the match between the examples and the complex. The examples in different categories play different roles during learning.

1. Strictly covered examples of a base complex: all examples in this categories are strictly covered by the base complex of an extended complex and satisfy all conditions of the base complex.

2. Strictly covered examples of an extended complex: the examples in this category do not match all conditions of the base complex of the extended complex, but their degree of fit to the base complex is larger than or equal to the threshold of the extended complex.

3. Flexibly covered examples of an extended complex: the examples in this category are not strictly covered by the extended complex, but are covered by the flexible classification method.

4. Nearly covered examples of an extended complex: The examples in this category are not strictly covered by the extended complex, but their degree of fit to the base complex of the extended complex is close to the threshold. The closeness is defined by the potential threshold PTH which will be discussed in Section 5.5. This category may intersect with the category of the flexibly covered examples.

5. Uncovered examples of an extended complex: The examples in this category are neither strictly covered nor nearly covered by the extended complex. This category may intersect with the category of the flexibly covered examples.

In Figure 5.5, e_1 and e_2 are strictly covered examples of the base complex, e_3 to e_6 are strictly covered examples of the extended complex, and e_7 and e_8 are nearly covered examples of the extended complex.

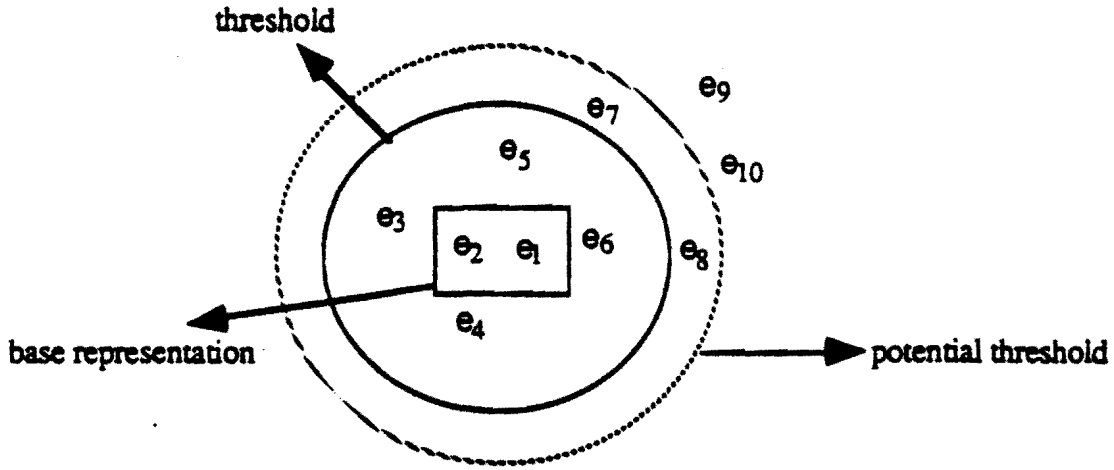


Figure 5.5: Categories of examples

5.4. The Learning Algorithm

The learning strategy used in FCLS is inductive learning (learning from examples). The task of learning a concept description in FCLS is, given a set of positive and negative examples, to generate a set of extended complexes and select a set of exemplars such that most of the positive examples and few negative examples are covered by the description. This section describes the learning algorithm of FCLS, the extended complex generation algorithm, the exemplar selection algorithm, and the weight learning algorithms, respectively.

5.4.1 The Learning Algorithm of FCLS

Figure 5.6 shows the learning algorithm of FCLS. The purpose of this algorithm is, given a set of concepts with their examples, to generate a disjunction of extended complexes and a set of exemplars for each given concept to flexibly cover most of the examples of the concept and few of

the examples of other concepts. This algorithm works in an iterative fashion. In each iteration, the concept description that has the largest error omission is generalized by either generating one extended complex or selecting a set of exemplars. The error omission of a concept description is the percentage of the number of the examples of the concept that are not flexibly covered by the descriptions. An extended complex or a set of exemplars is generated to minimize the error omission.

FCLS provides users with two parameters: *MAX-ERR-RATE* and *MIN-COVERAGE*. These two parameters are used as thresholds. *MAX-ERR-RATE* is the maximum error rate that an extended complex may have. The error rate of an extended complex is the ratio of the number of negative examples strictly covered and the number of all examples strictly covered by the extended complex, and the ratio reflects the inconsistency of the complex. *MIN-COVERAGE* is the minimum fraction of all positive examples that an extended complex must cover. A *MIN-COVERAGE* of 0 and a *MAX-ERR-RATE* of 0 will cause the algorithm to produce consistent and complete descriptions. An extended complex is acceptable if

$$(1)\ \frac{p}{pos} \geq \textit{MIN-COVERAGE}, \text{ and}$$

$$(2)\ \frac{n}{p + n} \leq \textit{MAX-ERR-RATE}.$$

where p (n) is the number of positive (negative) examples strictly covered by the extended complex, and pos is the total number of positive examples.

These two parameters control the distribution of the description between generalized extended complexes and exemplars. *MIN-COVERAGE* directly controls the distribution. Larger *MIN-COVERAGE* favours exemplars, while smaller *MIN-COVERAGE* favours extended complexes. When *MIN-COVERAGE* = 0, no exemplar is selected. When *MIN-COVERAGE* = 1, a concept description only includes exemplars unless there exists a single extended complex that covers all positive examples. *MAX-ERR-RATE* indirectly affects the distribution through *MIN-COVERAGE*. The larger *MAX-ERR-RATE*, the more negative examples can be covered by an extended complex so that it is easier to generate an extended complex that covers a large number of positive examples. *MAX-ERR-RATE* is also used to control the error rate produced by all concepts descriptions.

Each iteration of the algorithm first tries to generate an acceptable extended complex for the concept that has the largest error omission. If it fails to generate such an acceptable extended complex, it selects a set of exemplars. After an extended complex or a set of exemplars is

generated, all examples that are strictly covered by the newly generated extended complex or the newly selected exemplars are removed, and the new extended complex is added as a disjunct into the description of the selected concept, or the new exemplars are added as exceptions into the description. Then the current descriptions of all concepts are reevaluated together by applying the flexible classification method to decide the error rate of these current concept descriptions. If the error rate of the current descriptions is not larger than the parameter *MAX-ERR-RATE*, the algorithm terminates, otherwise it repeats. The error rate of a set of descriptions is the fraction of all training examples that are not correctly classified by the flexible classification method. The percentage of negative examples strictly covered by an acceptable extended complex is smaller than *MAX-ERR-RATE*, so a set of descriptions of all concepts whose error rate is smaller than *MAX-ERR-RATE* can be always generated and the worst case is that all examples are strictly covered by some extended complex. Because the flexible classification method increases the coverage of extended complexes, the final descriptions generated by applying the flexible classification method may consists of less extended complexes and less exemplars. The non-typical or uncertain examples are often flexibly covered. Because of the application of the flexible classification method, boundaries of extended complexes defined by threshold are soft.

Let DES be empty
Repeat
    Select the concept CNPT that has the largest error omission.
    Generalize CNPT by
        Either 1. Generate an acceptable extended complex CPX
        or 2. Generate a set of exemplars EXEM
    Add CPX or EXEM into DES
Until error-rate(DES) ≤ *MAX-ERR-RATE*
Return DES

Figure 5.6. The Learning Algorithm in FCLS

5.4.2 The Complex Generation Algorithm (CGA)

The complex generation algorithm generates the 'best' acceptable extended complex for a given concept from a set of positive and negative examples, if there exists some acceptable extended complex, otherwise it returns null. The process of generating the 'best' acceptable

extended complex is divided into two phases. The first phase generates a set of base complexes that satisfy the consistency requirement which is specified by the parameter *MAX-ERR-RATE*. The algorithm of the first phase is called the Base Complex Generation Algorithm (BCGA). The base complexes generated in the first phase serve as the initial extended complexes whose threshold is 1, and are optimized in the second phase to cover more positive examples through adjusting the distribution of the extended complexes between their base complexes and the similarity measure. The Extended Complex Optimization Algorithm (ECOA) serves as the algorithm in the optimization phase.

The searches in both algorithms are general-to-specific. In each step of the search, a base complex is specialized by either adding a new selector or removing an internal disjunctive element in one of its selectors. Because of the different purposes of the two algorithms, the specializations play different roles in these two algorithms. The goal of the specialization of a base complex in the first phase is to exclude more negative examples from the base complex so that the inconsistency of the base complex is reduced, while the goal of the specialization of a base complex in the second phase is to reduce the degree of fit of nearly covered negative examples so that the threshold of the extended complex can be decreased to include more positive examples. In section 5.2, it is mentioned that an extended complex can be generalized by decreasing its threshold. Decreasing the threshold of an extended complex can be viewed as relaxing the conditions that must be satisfied.

Both BCGA and ECOA algorithms resemble the *star* algorithm of AQ (Michalski, 1983) in that both perform a general-to-specific beam search. The differences between these two algorithms and the *star* algorithm are substantial. A major difference is that these two algorithms do not depend on any specific examples, both positive and negative examples. The reason for not using seeds is that the examples of a flexible concept can be divided into three types: typical, less typical and exceptions. If seeds were used, some example which should be allowed as a less typical example or an exception which should not be strictly covered by a base complex, might possibly selected as a seed, and thus would necessarily be strictly covered by the base complex created as a generalization of it. If a seed is not used, however, the space of search for a complex is larger than the space using a seed, thus these two algorithms may not be as optimum as the *star* algorithm. In practice this does not seem to be important. For the same reason, these two algorithms do not select any specific negative examples to specialize complexes in a partial star, instead they use some statistical information to select attribute values to specialize complexes.

The second difference is that these two algorithms use two different criteria to evaluate the quality and the potential quality of a complex. The quality value is used to determine the final

extended complex generated by the complex generation algorithm and the potential quality value is used to determine whether complexes can be improved. Two different evaluation functions are used because a currently 'good' (high quality) complex may not have a potential for improvement. These criteria will be discussed in section 5.5. The algorithms also differ from the *star* algorithm in the termination conditions used. In the *star* algorithm, the set of complexes in a star are specialized until all negative examples have been uncovered. In these two algorithms, large numbers of negative examples may be skipped over in a single cycle, and some negative examples may remain covered.

5.4.2.1 Phase 1: The Base Complexes Generation Algorithm

The purpose of this algorithm is to find a set of alternative base complexes which satisfy the consistency requirement given by the parameter *MAX-ERR-RATE*. Figure 5.7 gives the base complex generation algorithm. The algorithm starts with the most general base complex which strictly covers all positive and negative events. In order to find the base complexes that satisfy the consistency requirement, the strictly covered negative examples must be excluded from the most general base complex.

The technique used in the algorithm is a beam search. During each cycle, the consistency of each base complex in STAR is tested. If the consistency is high enough, the base complex is added to the set of CONSISTENT-CPXES and removed from STAR. Otherwise, the base complex is specialized by removing a value from one of selectors of the complex. This specialization is repeated for each of the selectors of the complex. The value of a selector is chosen to maximize the number of negative examples and minimize the number of positive examples excluded from the base complex after the value is removed. The selection criterion will be elaborated in section 5.5. This yields several new base complexes, each of which covers fewer negative examples. The new star is the union of these newly specialized base complexes. A certain maximum number (*MAXSTAR*) of these base complexes are selected for further processing. This set of base complexes is selected based on their potential quality as evaluated by the Base Complex Potential Quality Evaluation Function BCPQEF which will be described in section 5.5. The Algorithm terminates with a set of base complexes whose inconsistency (error rate) is smaller than *MAX-ERR-RATE* when STAR is empty. Figure 5.8(a) shows the most general base complex that the algorithm starts with and Figure 5.8(b) shows the set of consistent base complexes that the algorithm ends with. The set of consistent (or near consistent) base complexes generated in this algorithm will be optimized by the extended complex optimization algorithm in the second phase.

Let STAR be the set containing the most general complex that covers all events.
Let CONSISTENT-CPXES be empty.

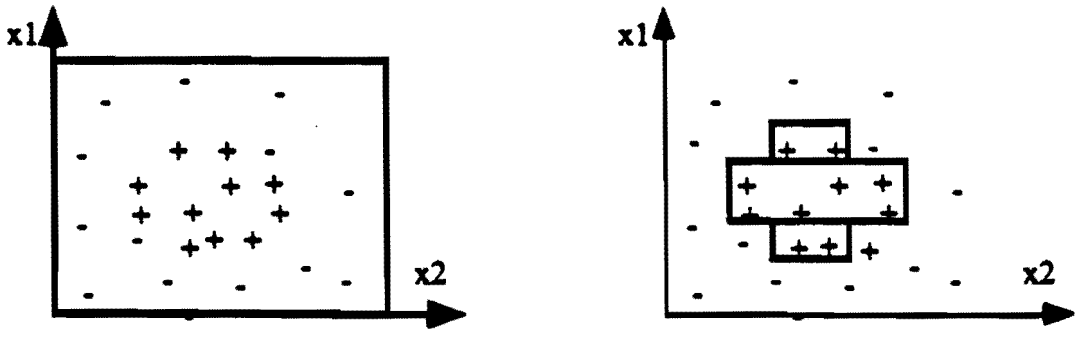
Repeat
  Let NEWSTAR be empty
  For each complex CPX in STAR
    For each attribute
      select a value to remove from CPX so that a more specific complex NEWCPX is generated.
      if error-rate(NEWCPX) ≤ *MAX-ERR-RATE* ,
        then add NEWCPX into CONSISTENT-CPXES
        else add NEWCPX into NEWSTAR

  Let STAR be MAXSTAR complexes with the largest potential quality in NEWSTAR.

until STAR is empty

Return CONSISTENT-CPXES

Figure 5.7: Phase 1: The Base Complex Generation Algorithm



(a) (b)

Figure 5.8: An illustration of the function of the phase 1

5.4.2.2 Phase 2: The Extended Complex Optimization Algorithm

The extended complex optimization algorithm starts with a set of initial extended complexes which are the base complexes generated in the first phase by setting their threshold and weights to 1. The goal of the algorithm is to optimize this initial set of extended complexes to cover more positive examples by adjusting their thresholds.

In a logic type representation, the coverage of a disjunct (complex) can be increased only by generalizing the disjunct itself. But the base complexes generated by the base complex generation algorithm are the most general base complexes that satisfy the consistency requirement, so they cannot be further generalized without degrading their consistency. Fortunately, the hybrid representation proposed in section 5.2 allows an alternative method to generalize an extended complex. The alternative method generalizes an extended complex by decreasing its threshold. In order to decrease the threshold of an extended complex without causing the increase of inconsistency, the nearly covered negative examples have to be moved further to the boundary (threshold) of the extended complex. In other words, the degree of fit of nearly covered negative examples must be reduced so that the threshold of the extended complex can be decreased without covering new negative examples. The way to reduce the degree of fit of nearly covered negative examples is to select some values of selectors to remove from the base complex. The values selected should occur on many nearly covered negative examples and few nearly covered positive examples. Thus, the process of optimization of an extended complex is the process of further specialization of its base complex and adjustment of its threshold.

The optimization of an extended complex can be viewed as finding the optimum distribution of the extended complex between its base complex and the similarity measure. The generality of the base complex together with the threshold decides this distribution. The more general the base complex, the greater the fraction of the examples covered by the corresponding extended complex are strictly covered by the base complex. The smaller the threshold, the smaller fraction of the examples covered by the corresponding extended complex are strictly covered by the base complex. If the threshold of an extended complex is 1, all examples strictly covered by the extended complex are strictly covered by its base complex, the extended complex is equivalent to its base complex. On the other hand, the base complex may be specialized to an exemplar, almost all examples covered by the extended complex are not strictly covered by the base complex. During the process of optimization, the base complexes of the initial extended complexes are gradually

specialized, at the same time their thresholds are gradually decreased. So the process shifts the concept meaning from the base complex to the similarity measure until the optimum distribution is achieved.

The extended complex optimization algorithm is similar to the base complex generation algorithm in the sense that it performs a general-to-specific search during the process of optimization. However, the goals of the two algorithm are different. The goal of the base complex generation algorithm is to generate a set of nearly consistent base complexes, whereas the goal of the extended complex optimization algorithm is to find the 'best' extended complex. Because of the different goals, there are several differences between the two algorithms. The major difference involves the negative examples that the two algorithms try to exclude. The base complex generalization algorithm reduces the number of negative examples strictly covered by the base complex, whereas the extended complex optimization algorithm reduces the number of nearly covered negative examples of the extended complex. This difference is reflected in the potential quality evaluation functions and the attribute value selection criteria used in the two algorithms. The potential quality of a base complex is computed based only on the examples strictly covered by the base complex, while the potential quality of an extended complex is computed based on the examples strictly covered by the extended complex as well as the examples nearly covered by the extended complex. In the base complex generalization algorithm, the attribute value is selected to specialize a base complex so that as many as strictly covered negative examples are excluded from the base complex as possible, while in the extended complex optimization algorithm, the attribute value is selected to specialize the base complex of an extended complex so that the degree of fit of nearly covered negative examples are reduced as much as possible.

Figure 5.9 shows the extended complex optimization algorithm. The input of the algorithm is a set of initial extended complexes that are equivalent to the base complexes generated by the base complex generation algorithm. The algorithm first optimize each of the initial extended complexes by computing its new weights and new threshold. In section 5.4.4, two different weight learning algorithms will be introduced. The threshold of an extended complex is determined so that the 'best' quality of the complex is achieved. The STAR is initialized as the *MAXSTAR* complexes with the highest potential quality. These *MAXSTAR* complexes are selected from all initial extended complexes. The potential quality of an extended complex is evaluated by the Extended Complex Potential Quality Evaluation Function ECPQEF which is different from BCPQEF used to evaluate the potential quality of a base complex and will be defined in section 5.5. Then the 'best' acceptable extended complex is selected from the set of optimized initial extended complexes as the initial 'best' complex. This 'best' extended complex is subject to replacement by a better extended

complex during the process of optimization. After the algorithm terminates, the 'best' acceptable extended complex is the extended complex generated by the complex generation algorithm. The 'best' extended complex is the complex with the highest quality which is evaluated by the Quality Evaluation Function (QEF) which will be defined in section 5.5. If no acceptable extended complex can be generated, BEST-CPX is empty when the algorithm terminates.

Like the base complex generation algorithm, the algorithm repeats the beam search until the stop condition is satisfied. In each cycle of the loop, a set of new extended complexes is generated. Each newly generated extended complex is evaluated by two function: QEF and ECPQEF. The acceptable complex with the highest value of QEF replaces BEST-CPX, if its value of QEF is larger than or equal to BEST-CPX's. The complexes which have no value to be improved or cannot be improved are removed from NEWSTAR. Two kinds of complexes, the most specific complexes and the complexes with 0 potential improvement, cannot be improved. The most specific complexes are events in event space, they cannot be further specialized. The complexes with very low quality (comparing with BEST-CPX) have no value to be improved. Issues about the potential improvement will be discussed in section 5.5. The *MAXSTAR* new extended complexes are selected for further improvement. *MAX-TRIES* is an integer parameter which controls the execution of the loop. If BEST-CPX has not been improved in *MAX-TRIES* steps, the algorithm stops.

The algorithm uses maximum specificity criterion rather than maximum generality criterion in generating an extended complex. The algorithm prefers a more specific complex over a more general one when these two complexes have equal quality. In other words, when two complexes have the same quality, the algorithm selects the one that has higher certainty. The advantages of using maximum generality criterion is that the descriptions generated have larger predictive power and are simpler. The major disadvantage of the most general descriptions is that they are less certain. On the other hand, the descriptions generated under maximum specificity criterion are often too specific to predict unseen events. In this algorithm, the most specific descriptions are generated to maximize the certainty of descriptions, and the predictive power lost is recovered by applying the flexible classification method. The events strictly covered by these descriptions are more certain than the events flexibly covered.

Figure 5.10(a) shows the initial extended complexes that the algorithm starts with and Figure 5.10(b) shows the extended complex that the algorithm ends with. In Figure 5.10(b), the circle is the extended complex, and the square inside the circle is its base complex. It can be seen that the base complex in Figure 5.10(b) is more specific than the two base complexes in Figure 5.10(a),

but the extended complex is more general than both of the two initial extended complexes that are equivalent to the base complexes.

For each complex CPX in CONSISTENT-CPXES (generated in Phase 1)
    Compute the weights and threshold for CPX
Let STAR be *MAXSTAR* complexes with the highest potential quality in CONSISTENT-CPXES
If there exist some acceptable complexes in CONSISTENT-CPXES
    then let BEST-CPX be the acceptable complex with the highest quality
    else let BEST-CPX be empty
Let NO-IMPROVEMENT be 0

Repeat
    Let NEWSTAR be empty

    For each complex CPX in STAR
        For each attribute
            select a value to remove from CPX to generate a new complex NEWCPX
            compute the weights and threshold for NEWCPX
            if NEWCPX is acceptable and has equal or higher quality than BEST-CPX
                then replace BEST-CPX by NEWCPX
                     set NO-IMPROVEMENT to 0
                else add 1 to NO-IMPROVEMENT
            add NEWCPX into NEWSTAR

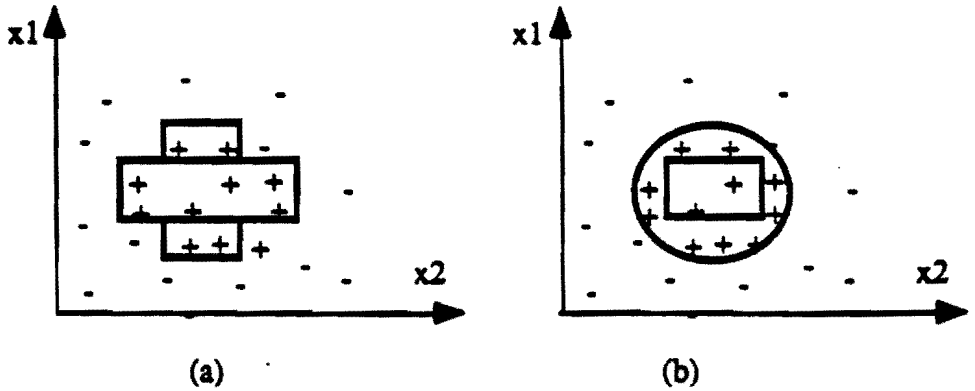
    Remove all complexes that cannot be improved from NEWSTAR
    Let STAR be *MAXSTAR* complexes with the largest potential quality in NEWSTAR.

until NO-IMPROVEMENT > *MAX-TRIES* or STAR is empty

Return BEST-CPX

Figure 5.9: Phase 2: The Extended Complex optimization algorithm

---



(a) (b)

Figure 5.10.: An illustration of the function the phase 2

5.4.3 Exemplar Selection Algorithm

When the complex generation algorithm fails to generate an acceptable extended complex for a selected concept, all remaining uncovered examples of the concept are widely dispersed in the event space. It is dangerous to generalize these dispersed examples. To not lose the coverage of these examples, an exemplar based learning algorithm is applied to them. The algorithm of selecting exemplars is similar to the best model algorithm (Kibler and Aha, 1987). All examples of the selected concept that are not correctly classified by the current concept description are the candidates for exemplars. All exemplars are selected from these candidates. It first selects a candidate as an exemplar, then it reclassifies all remaining candidates and removes all correctly classified candidates. The algorithm repeats the process of selecting a candidate and removing all correctly classified candidates until no more candidate exists. The flexible classification method is used to classify candidates. After some other concepts are generalized, previously correctly classified examples of the selected concept may no longer be correctly classified. For this reason, the exemplar selection algorithm may be executed several times for one concept, with new exemplars selected for each execution.

It is straightforward to combine the inductive learning approach with the exemplar-based approach in the hybrid representation. Because of the similarity measure, an extended complex can be viewed as an abstract exemplar, and an exemplar can be treated as an extended complex.

5.4.4 Weight Learning

In the hybrid representation described in section 5.2, each selector of an extended complex is associated with a weight which is the degree of necessity of the selector in the extended complex. A weight is a real value ranging from 0 to $+\infty$. A 0 weight means the selector is irrelevant, and a $+\infty$ weight means the selector is necessary. The larger a weight of a selector, the more necessary the selector. The weights of an extended complex are decided during learning. Each time an extended complex is generated, its weights are computed. In this section, two alternative methods for weight learning are discussed. One is based on counting the number of positive examples and negative examples that do not satisfy a selector. The other is based on the degree of fit of positive and negative examples that do not satisfy a selector. Both methods have been implemented in

FCLS.

5.4.4.1 Weight learning by counting unmatched examples

This method is similar to the method to compute Logic Necessity (LN) in STAGGER (Schlimmer, 1987). In computing the weight of a selector, the method counts the number of positive and negative examples that do not match the selector. This weight learning method is called count-based weight learning method. In the count-based method, the weight of the selector SEL w(SEL) is computed as follows:

$$w(SEL) = \frac{p(unmatched \mid NEG)}{p(unmatched \mid POS)}$$

where p(unmatched | NEG) and p(unmatched | POS) are the fraction of positive and negative examples which do not match with SEL. w(SEL) ranges from 0 to $+\infty$. When the selector SEL is satisfied by all positive examples, p(unmatched | POS) = 0 so that w(SEL) = $+\infty$ and the selector SEL is necessary. When the selector SEL is satisfied by all negative examples, p(unmatched | NEG) = 0 so that w(SEL) = 0 and the selector is totally unnecessary. This case occurs seldom, because such a selector is usually removed in the process of complex generation. The fewer negative examples satisfy the selector SEL, the larger p(unmatched | NEG) and w(SEL). The more positive examples satisfy the selector SEL, the smaller p(unmatched | POS), therefore the larger w(SEL).

The count-based weight learning method is simple and easy to be implemented. But it has a problem. The weight of a selector of an extended complex is the degree of necessity of the selector in the extended complex, the same selector occurring in two different complexes may have different degree of necessity in these two complexes. For example, suppose that the concept 'good job' is defined by two disjuncts: high-pay & interesting, and high-pay & good-location. If one cares more location than interesting, then high-pay in the disjunct high-pay & good-location should be less important than it is in the disjunct high-pay & interesting. The weights of a selector is always same regardless of complexes when it is computed by the count-based weight learning method.

Events of a concept may form several clusters. Events in different clusters have different reasons for being events of the concept, therefore the principles behind different clusters of a concept are different. In the hybrid representation described in this chapter, ideally each of the

clusters is represented by one extended complex. The weights of an extended complex should be computed based on only the examples of the cluster that the extended complex describes. Next subsection proposes an alternative weight learning method to tackle the problem.

5.4.4.2 Weight learning using degree of fit of unmatched examples

In the previous section, it is argued that the weights of an extended complex should be computed solely based on the examples from the cluster that the extended complex describes. Unfortunately it is not known in advance which examples form the cluster. In the following a method called degree-of-fit-based (df-based) weight learning is proposed. It is hoped that the method can partially solve the problem of the count-based weight learning method.

In df-based weight learning method, the weight of a selector SEL of an extended complex CPX w(SEL, CPX) is computed based the degree of fit of all examples (both positive and negative) that do not satisfy SEL to the complex CPX. The examples with larger degree of fit contribute more to w(SEL, CPX) than the examples with smaller degree of fit, because the examples with larger degree of fit have larger chance to be covered by the extended complex CPX, that is they have larger chance to belong to the cluster described by the complex. Also, because examples with larger degree of fit are more typical, more typical examples should contribute more than less typical examples. The examples with 0 degree of fit contribute nothing to w(SEL, CPX), because they are far from CPX and have little chance to belong to the cluster described by CPX. In df-based weight learning method, w(SEL, CPX) is computed as follows:

$$w(SEL, CPX) = \frac{N(SEL, CPX)}{P(SEL, CPX)}$$

$$N(SEL, CPX) = \frac{\sum_{ne \in \neg SEL \& NEG} SSM(ne, CPX)}{\sum_{ne \in NEG} SSM(ne, CPX)}$$

$$P(SEL, CPX) = \frac{\sum_{pe \in \neg SEL \& POS} SSM(pe, CPX)}{\sum_{pe \in POS} SSM(pe, CPX)}$$

The basic idea of the method is the same as in the count-based method, the difference is that the number of unmatched (all) positive (negative) examples is replaced by the summation of degree

of fit of these unmatched (all) positive (negative) examples. In general, the following arguments are still true. The fewer the negative examples that satisfy the selector SEL, the larger w(SEL, CPX) and the more necessary the selector. The more positive examples satisfy the selector SEL, the larger w(SEL, CPX) and the more necessary the selector. In addition, the larger the degree of fit of the positive examples that do not satisfy the selector, the smaller the w(SEL, CPX), and the larger the degree of fit of the negative examples that do not satisfy the selector, the larger the w(SEL, CPX). The arguments could be explained as follows. Positive examples with small degree of fit may not belong to the cluster that the extended complex describes, they should be counted less for the weights of the complex. The positive examples with large degree of fit are very typical examples, the selector that is not satisfied by many typical examples is less important. The negative examples with large degree of fit are near miss examples, the selector that distinguishes near miss examples from positive examples is more important.

The df-based weight learning method is expected to improve the performance over the count-based method. However, this improvement in performance is brought at the price of greater complexity and slower speed.

5.5 Concept Evaluation and Attribute Value Selection

The evaluation function (or preference criterion) that evaluates concept descriptions plays an important role in all learning systems, especially inductive learning systems. Without the evaluation functions, these learning systems would lack the means for deciding which of the many alternative candidate descriptions should be chosen. This section discusses the evaluation functions used in FCLS.

In most concept learning systems, the problem of learning can be viewed as a problem of search for the 'best' solutions (concept descriptions). Because of the complexity of brute force search, all learning systems search the description space heuristically. Therefore, an evaluation function is needed to decide one or more intermediate hypotheses for further improvement. Such intermediate hypotheses should have potential for improvement. And also an evaluation function is needed to determine the best description that is the final solution of the search, when the search algorithm terminates. Therefore, two evaluation functions are needed, one to determine which intermediate hypotheses have potential for improvement, the other to determine which description is currently best. Most current inductive learning systems use one evaluation function for both purposes. It means that a good description always has great potential for improvement.

Unfortunately this is not always true. In both section 5.5.2 and section 5.53, some examples will be given to show that a good description may not have potential for improvement.

In FCLS, two different evaluation functions are used for the two purposes mentioned above. One is used to evaluate the quality of a description, another evaluates the potential quality of a description. The description with the highest quality is the best description, while the description with the highest potential quality has the greatest potential for improvement. As we presented in section 5.2, a description is expressed as a disjunction of extended complexes. In the the learning algorithm, each extended complex is generated independent of the others, so the problem of evaluating a description becomes the problem of evaluating the complexes of the description. Thus, the evaluation functions introduced in this section evaluate complexes. The evaluation function that evaluates the quality of a description is called *Quality Evaluation Function* (QEF). The potential quality evaluation function consists of two subfunctions. The first evaluates the potential quality of a base complex and is called *Base Complex Potential Quality Evaluation Function* (BCPQEF), and the second evaluates the potential quality of an extended complex and is called *Extended Complex Potential Quality Evaluation Function* (ECPQEF). BCPQEF is used in the base complex generation algorithm, while ECPQEF is used in the extended complex optimization algorithm. In the following subsections, these three evaluation functions QEF, BCPQEF and ECPQEF are defined. In the final subsection, the heuristics used to select the attribute values for specialization are discussed.

5.5.1 The Quality Evaluation Function: QEF

The problem of evaluating description is not new to empirical learning systems, and a number of measures have been developed in the past. Some of them concentrate solely on the aspects of completeness and consistency (e.g. Mitchell, 1978). Other measures include additional criteria, such as the simplicity and cost of using the learned descriptions (Michalski, 1973). Broader aspects of the problem of deciding what should be included in evaluation functions for judging competing inductive hypotheses are discussed in (Mitchell, 1980; Michalski, 1983; Utgoff, 1986).

As in most inductive learning systems, the main concern of the quality of an extended complex is its completeness and consistency with regard to the training examples. The relationship between completeness and consistency is not very obvious and varies from one domain to another. But in general, one can gain completeness of an extended complex at the expense of consistency,

or one can gain consistency by sacrificing completeness. They are two competing criteria. For this reason, QEF is defined as the product of these two parts: normalized completeness and normalized consistency.

$$QEF(cpx) = NCOM(cpx) * NCON(cpx)$$

$$NCOM(cpx) = \begin{cases} 1 & \text{if } \textit{MIN-COVERAGE} = 1 \text{ and } \frac{p}{pos} = 1 \\ 0 & \text{if } \textit{MIN-COVERAGE} = 1 \text{ and } \frac{p}{pos} < 1 \\ \max\left\{\frac{\frac{p}{pos} - \textit{MIN-COVERAGE}}{1 - \textit{MIN-COVERAGE}}, 0\right\} & \text{otherwise} \end{cases}$$

$$NCON(cpx) = \begin{cases} 1 & \text{if } \textit{MAX-ERR-RATE} = 0 \text{ and } \frac{n}{p+n} = 0 \\ 0 & \text{if } \textit{MAX-ERR-RATE} = 0 \text{ and } \frac{n}{p+n} > 0 \\ \max\left\{\frac{\textit{MAX-ERR-RATE} - \frac{n}{p+n}}{\textit{MAX-ERR-RATE}}, 0\right\} & \text{otherwise} \end{cases}$$

Where *pos* represents the total number of positive examples. *p* and *n* are the number of positive and negative examples strictly covered by the extended complex cpx, respectively.

The first part of QEF, NCOM, represents the completeness of the complex. The more positive examples covered, the larger NCOM. When the completeness of an extended complex is not larger than the parameter *MIN-COVERAGE*, the extended complex is not acceptable, its NCOM is equal to 0, so is QEF. NCOM also indirectly measures the simplicity of a description, because the more positive examples an extended complex covers, the less extended complexes are required to describe a concept.

The second part of QEF, NCON, represents the consistency of an extended complex. The less negative examples covered, the larger NCON. When the inconsistency of an extended complex is not smaller than the parameter *MAX-ERR-RATE*, the complex is not acceptable, its NCON is equal to 0, so is QEF. The quality of an acceptable extended complex is larger than 0.

In FCLS, concept descriptions are graded, and the result of a match between an example and an extended complex is a value between 0 and 1, called the degree of the fit between the example and the base complex of the extended complex. The higher the degree of fit, the more typical (or

certain) the example. Many real world concepts have central tendencies (centers), the probability that an example belongs to a concept decreases with the increase in the distance between the example and the center of the concept. The centers are represented by base complexes. The degree of fit between an example and a base complex is decided by the distance between the example and the base complex, therefore the examples with less degree of fit are more possible to be negative examples than the examples with larger degree of fit. Furthermore, the boundary examples have more chance to include noisy information, and learning algorithms are more sensitive to noise included in boundary examples. The noise included in boundary examples may change the the concept that the examples belong, while the noise occurring in typical examples may only the typicality of the examples.

When inconsistency is allowed for an extended complex, the complex may cover some negative examples. From the above discussion, it can be derived that most of the covered negative examples of an extended complex should stay around the boundary, few are in the center of the concept. Therefore, an inconsistent extended complex that covers negative examples with lower degree of fit is preferred over an inconsistent extended complex that covers negative examples with higher degree of fit. In other words, given two complexes with the same number of positive and negative examples covered, the one that covers the negative examples with lower degrees of fit is better than the one that covers the negative examples with higher degrees of fit. In order to reflect this preference, the inconsistency $\frac{n}{p+n}$ is replaced by Weighted Inconsistency (WI) in NCON(cpx). In the Weighted Inconsistency of an extended complex, the negative examples covered are weighted according to their degree of fit to the complex, the larger degree of fit, the larger weight of the negative examples. The Weighted Inconsistency of an extended complex CPX is computed as follows:

$$WI(CPX) = \frac{WNC(CPX)}{WPC(CPX) + WNC(CPX)}$$

$$WPC(CPX) = \sum_{p_i \in CPX \ \& \ POS} W(p_i, CPX)$$

$$WNC(CPX) = \sum_{n_i \in CPX \ \& \ POS} W(n_i, CPX)$$

$$W(e, CPX) = (1 - c) * \frac{SSM(e, CPX) - th(CPX)}{1 - th(CPX)} + c$$

In WI(CPX), WPC(CPX) and WNC(CPX) are the Weighted Positive Coverage of CPX and Weighted Negative Coverage of CPX, respectively. p_i (n_i) are all positive (negative) examples that

strictly covered by CPX. W(e, CPX) is the weight of the covered example e.

$\frac{SSM(e,CPX) - th(CPX)}{1 - th(CPX)}$ is the normalized degree of fit of the example e. Because e is strictly covered by CPX, $\frac{SSM(e,CPX) - th(CPX)}{1 - th(CPX)}$ is larger than or equal to 0. c is a constant between 0 and 1, in the current implementation of FCLS, c is set to 0.2. The weight of e W(e, CPX) ranges from c to 1. W(e, CPX) increases with the increase of SSM(e, CPX). When SSM(e, CPX) = 1, W(e, CPX) = 1. When SSM(e, CPX) = th(CPX), W(e, CPX) = c. The Weighted Inconsistency WI(CPX) increases with an increases in the number of negative examples covered and an increase in the degree of fit of the negative examples covered.

5.5.2 Base Complex Potential Quality Evaluation Function: BCPQEF

Before describing the details of BCPQEF, let us first discuss the potential quality of both base complexes and extended complexes. The potential quality of complexes is used to decide whether a complex is worth being improved or not, whether a complex can be improved or not and how much a complex can be improved. Based on the potential quality of complexes, both the base complex generalization algorithm and the extended complex optimization algorithm determine which complexes should be selected for further improvement. The potential quality of a complex consists of two parts: current quality and potential improvement. Current quality is similar to the quality which concerns about the completeness and consistency of a complex, while the concern of potential improvement is how much improvement can be achieved on the base of current quality of the complex. The current quality of a complex measures the worthiness of the improvement of the complex. The potential improvement measures how much the complex can be improved. The complex with low current quality is not worth being improved even it has high potential improvement, while the complex with low potential improvement has little reason for further improvement.

In evaluating the potential quality of a base complex, only the examples strictly covered by the base complex are considered. The current quality of a base complex is computed based on the number of positive and negative examples strictly covered by the base complex. The potential improvement of a base complex is computed based on the distribution of the positive examples and negative examples inside the base complex. Some distributions make a base complex much easier to be improved than the others. For examples, Figure 5.11, shows two complexes CPX1 and CPX2, which cover the same number of positive and negative examples, they have the same

current quality. But CPX1 is much easier to be improved than CPX2, because the positive examples covered by CPX2 are scattered, while the positive examples covered by CPX1 are very concentrated.

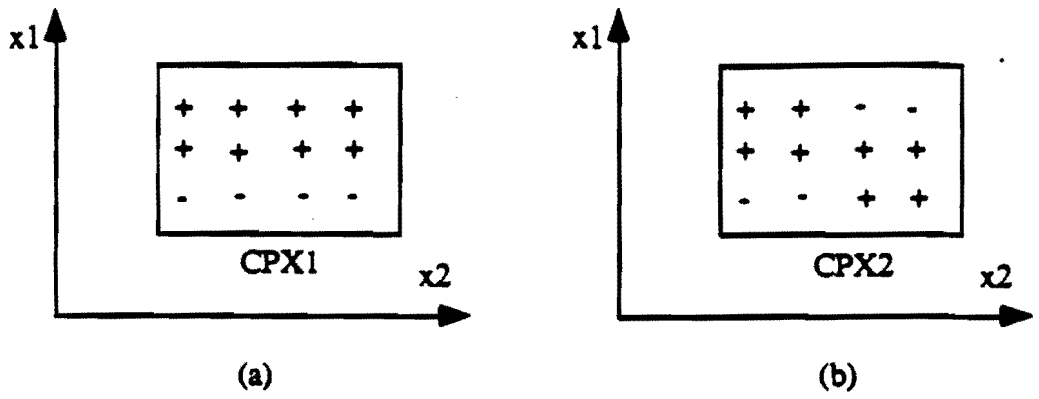


Figure. 5.11: Illustration of the difference
between the quality and potential improvement of base complexes

The base complex potential quality evaluation function of the complex cpx is defined as a product of two parts:

$$BCPQEF(cpx) = \frac{p}{pos} * PLAC(cpx)$$

where p and pos are the number of positive examples strictly covered by cpx and the number of all positive examples, respectively. $\frac{p}{pos}$ is the completeness of cpx. PLAC, Probabilistic Looking-Ahead Criterion, concerns two aspects: consistency and the distribution of positive examples and negative examples inside the base complex cpx. The rest of the subsection defines PLAC(cpx).

Consider the base complex cpx represented as:

$$[A_1 = v_{11} \vee \ldots \vee v_{11m}] \,\&\, \ldots \,\&\, [A_n = v_{n1} \vee \ldots \vee v_{nnm}]$$

which includes selectors for all attributes, both relevant and irrelevant. The selector of an irrelevant attribute includes all values of the attribute. For each attribute A_i ($1 \le i \le n$), its relevance $R(A_i)$ is computed as follow:

$$R(A_i) = \sum_{k = 1 \ldots i_m} p(A_i = v_{ik}|pos) * p(pos|A_i = v_{ik})$$

where pos represents the set of all positive examples strictly covered by cpx. The larger the conditional probability $p(pos|A_i = v_{ik})$, the fewer the negative examples that share the value v_{ik}. If the probability is equal to 1, all examples with $A_i = v_{ik}$ are positive examples. If the probability is 0, all examples with $A_i = v_{ik}$ are negative examples, so the removal of the pair $A_i = v_{ik}$ from cpx only reduces the number of negative examples covered. This probability is also used to select attribute values for specialization. $p(pos|A_i = v_{ik})$ is weighted by the conditional probability $p(A_i = v_{ik}|pos)$. The larger $p(A_i = v_{ik}|pos)$, the greater the proportion of positive examples sharing the value v_{ik}.

The $R(A_i)$ value of a cpx ranges from the consistency of cpx (the ratio of the number of positive examples covered by the base complex and the number of examples covered by the base complex) to 1. We can see that $R(A_i)$ of a base complex is dependent on the consistency of the base complex which is the smallest value of $R(A_i)$. When all $p(pos|A_i = v_{ik})$ ($k = 1, \ldots, i_m$) are equal to the consistency of the complex, all positive and negative examples are equally distributed over all values of the attribute A_i and $R(A_i)$ has the smallest value: the consistency of the base complex. When all $p(pos|A_i = v_{ik})$ ($k = 1, \ldots, i_m$) are equal to either 1 or 0, both positive and negative examples are highly concentrated and $R(A_i) = 1$. For an attribute whose R value is 1, one can obtain a consistent complex by removing all of its values whose probability $p(pos|A_i = v_{ik})$ is equal to 0 without losing any coverage of positive examples.

Generally speaking, when all positive and negative examples are scattered on all values of an attribute A_i, $R(A_i)$ has a small value, and when all positive and negative examples are concentrated on some of the values of an attribute A_i, $R(A_i)$ has a large value. So $R(A_i)$ reflects the relevance of the attribute A_i in the complex cpx. The larger value of $R(A_i)$, the more relevant the attribute.

When all $R(A_i)$ ($1 \leq i \leq n$) of a base complex are equal to the consistency of the base complex, the positive examples covered by the base complex are very dispersed and the complex is very difficult to improve. When some of $R(A_i)$ have larger value, the complex is easier to improve. When one $R(A_i)$ is equal to 1, the complex can be improved to be consistent without losing any positive coverage. The difference between the consistency of a base complex and its $R(A_i)$ reflects the potential improvement of the base complex. The larger the difference, the more potential improvement. $R(A_i)$ of a complex is also dependent on the consistency of the complex. $R(A_i)$ of a complex with higher consistency is more likely to have a larger value than an $R(A_i)$ of a complex with lower consistency. $R(A_i)$ of a consistent base complex is always equal to 1. A consistent base

complex has no potential improvement, because it cannot be improved.

The Probabilistic Looking Ahead Criterion (PLAC) is defined using the relevance of attributes $R(A_i)$. In the base complex generation algorithm, a base complex is specialized in order to achieve greater consistency, and reduce its completeness as little as possible. A base complex with highly concentrated covered positive examples are highly and high consistency has high potential quality, that is a base complex with some very large $R(A_i)$ has high potential quality. Following, three definitions of PLAC(cpx) are introduced.

First, PLAC(cpx) can be defined as the average of $R(A_i)$ on all attributes.

$$PLAC(cpx) = \sum \frac{R_i}{n}$$

The base complex with a higher average value has a higher potential quality. The problem with this definition is that it is too sensitive to irrelevant attributes. If many of the attributes are irrelevant, the average will be very low even some attributes are highly relevant. For instance, if one of $R(A_i)$ is equal to 1, and the others have very low $R(A_i)$, the average is very low. The base complex, however, has very a large potential improvement, because it can be improved to a consistent complex without loss of any completeness.

Second, one can take the probabilistic sum of all $R(A_i)$. The probabilistic sum of $R(A_1)$ and $R(A_2)$ is defined as follows:

$$PLAC(cpx) = R(A_1) + R(A_2) - R(A_1) * R(A_2)$$

The advantage of the probabilistic sum is that if one of R_i is equal to 1, the result of the probabilistic sum is equal to 1 too. This method still suffers when there exist many irrelevant attributes. When a large number of attributes are present in a complex, the result of the above formula is always close to 1. One way to overcome the problem is to set a threshold for probabilistic sum. If the value of $R(A_i)$ is less than the threshold, it is set to 0.

The simplest method is that only the largest R_i is taken into account. This method is called one step looking-ahead. It is the method that is currently implemented in FCLS.

$$PLAC(cpx) = Max(R_i)$$

It is mentioned at the beginning of the section that the potential quality consists of two parts: current quality and potential improvement. These two parts are not explicitly expressed in the

above formula. The value of PLAC(cpx) ranges between the consistency of the base complex: $\frac{p}{p+n}$ and 1, so the potential quality of a base complex cpx can be decomposed as follows:

$$\frac{p}{pos} * \frac{p}{p+n} + \frac{p}{pos} * [PLAC(cpx) - \frac{p}{p+n}]$$

The first part of the decomposed formula, $\frac{p}{pos} * \frac{p}{p+n}$, is the current quality and the second part can be thought as the potential improvement.

5.5.3 Extended Complex Potential Quality Evaluation Function: ECPAEF

The extended complex potential quality evaluation function ECPQEF evaluates the potential quality of an extended complex and is used in the extended complex optimization algorithm. The goal of this algorithm is to increase the coverage of positive examples of extended complexes through decreasing the threshold of the extended complexes. The uncovered positive examples that most probably become covered are the nearly covered positive examples. The more nearly covered positive examples an extended complex has, the higher the potential improvement of the extended complex. If all nearly covered examples of an extended complex are negative, the extended complex is surrounded by negative examples and has no potential improvement. The potential improvement of an extended complex is computed based on the nearly covered examples which will be defined more precisely later. The current quality of an extended complex is computed based on the examples strictly covered by the extended complex.

In Figure 12, inside circles are thresholds of the two extended complexes, and the outside circles are the potential thresholds of the two extended complexes. The potential threshold of an extended complex is used to decide nearly covered examples. All examples between threshold and potential threshold are nearly covered examples. Potential threshold will be defined later. + and - are positive and negative examples, respectively. In this figure, CPX1 has higher current quality than CPX2, but has little potential improvement, since it is surrounded by negative examples. In contrast, CPX2 has lower current quality, but has larger potential improvement.

Before presenting the evaluation function, I first precisely define nearly covered examples. To do this, the potential threshold of an extended complex cpx PTH(cpx) is introduced first. The potential threshold is defined by:

$$PTH(cpx) = TH(cpx) - n * ADS(cpx)$$

where TH(cpx) is the threshold of cpx, and n is a positive integer. The larger the value of n, the greater number of nearly covered examples. If n is too large, all examples become nearly covered examples. If n is too small, only few examples are nearly covered. The experimental results show that 2 is a reasonable number for n. ADS(cpx) is the Average Decrease of the degree of fit between examples and the base complex of cpx on all possible specializations of the complex. Each specialization of a complex decreases the degree of fit of some examples. Suppose the value i of attribute A is removed from the base complex of an extended complex, the degree of fit between all examples with the value i of attribute A and the base complex decreases. The amount of decrease is the ratio of the weight of the selector whose corresponding attribute is A and the possible largest distance MAXDIS(cpx) which was introduced in section 2 of this chapter. This amount of decrease is only an approximation because the weights of all selectors change after the specialization is done. ADS(cpx) is computed as follows:

$$ADS(cpx) = \sum \frac{w_i}{MAXDIS(CPX)} * \frac{1}{m}$$

Where w_i ($1 \leq i \leq m$) is the weight of the selector i in cpx, m is the number of selectors in cpx.

Nearly covered examples of an extended complex cpx are all examples whose degrees of fit to the base complex of cpx are smaller than TH(cpx) and larger than or equal to PTH(cpx).

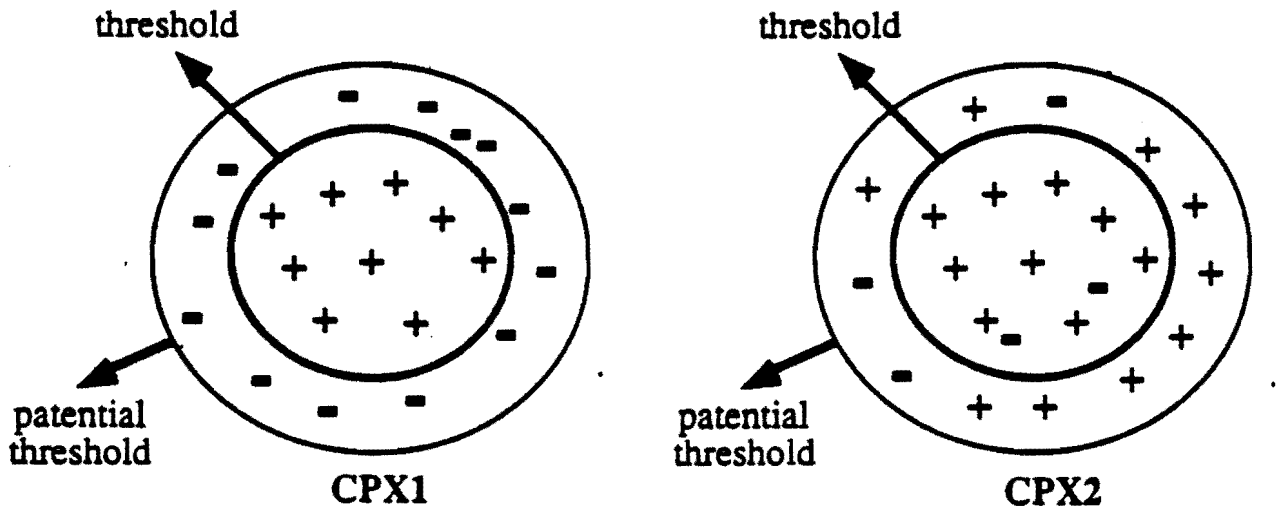


Figure 5.12: Illustration of the difference between the current quality and potential improvement of complexes in phase 2

The potential improvement of an extended complex is evaluated from nearly covered examples of the complex. The more positive nearly covered examples, the higher potential improvement. Each of the nearly covered examples is associated with a value that ranges from 0.5 to 1. This value is used in evaluating the potential improvement. Examples closer to threshold get larger value than examples further from threshold, because the nearly covered examples closer to threshold have more chance to become covered examples than the examples farther from threshold. The value of all covered examples can be considered as 1. Thus, most of the nearly covered examples get a smaller value than covered examples. This is reasonable, because nearly covered examples only have a chance to become covered, but they are not covered yet. The extended complex potential quality evaluation function ECPQEF of the complex cpx is defined as:

$$\frac{PC(cpx)}{pos} * \frac{PC(cpx)}{PC(cpx) + NC(cpx)}$$

$$PC(cpx) = p + \sum_{ncp_i \in \text{nearly-covered \& POS}} VNC(ncp_i, cpx)$$

$$NC(cpx) = n + \sum_{ncn_i \in \text{nearly-covered \& NEG}} VNC(ncn_i, cpx)$$

PC(cpx) and NC(cpx) are the positive coverage and negative coverage of cpx. The positive coverage is the sum of two parts: the number positive examples covered and the sum of the values of all nearly covered positive examples. The negative coverage is the sum of the two corresponding parts. This evaluation function is similar to the quality evaluation function, the difference is that the number of positive and negative examples in the quality evaluation function is replaced by the positive and negative coverage. $VNC(ncp_i, cpx)$ and $VNC(ncn_i, cpx)$ are the value of the nearly covered positive example ncp_i and the nearly covered negative example ncn_i, respectively. pos is the total number of positive examples. p and n are the number of positive and negative examples covered by the complex, respectively.

The value of a nearly covered example e of the extended complex cpx VNC(e, cpx) is computed as follows:

$$VNC(e, cpx) = 1 - 0.5 * \frac{TH(cpx) - SSM(e, cpx)}{TH(cpx) - PTH(cpx)}$$

SSM(e, cpx) is the degree of fit of the example e to the complex cpx. because e is a nearly covered example, PTH ≤ SSM(e, cpx) < TH.

5.5.4 The Criterion for Selecting Specialization Operations

In both the base complex generation algorithm and the extended complex optimization algorithm, a general-to-specific beam search is performed. In each step of the specialization process, some systems, e.g. CN2 (Clark and Niblett, 1989), generate and evaluate all possible specializations of a complex. This approach increases the complexity of the learning algorithm dramatically when the domains of some attributes are very large. FCLS does not generate all possible specializations, instead it selects one value for each attribute as the specialization operations of a complex.

Because the purposes of the specializations in the base complex generation algorithm and the extended complex optimization algorithm are different, the criteria for selecting the specialization operations are different. We first introduce the criterion used in the base complex generation algorithm. The specialization of a base complex in this algorithm tries to reduce the number of negative examples strictly covered by the base complex as much as possible and the number of positive examples strictly covered by the base complex as few as possible. Therefore, the value of an attribute that appears the most frequently in the negative examples strictly covered by the base complex and the least frequently in the positive examples strictly covered by the base complex is selected as one specialization operation of the base complex. Specifically, the value v_{ik} of the attribute A_i with the smallest conditional probability $p(pos|A_i = v_{ik})$ is selected. Where pos is the set of all positive examples strictly covered by the base complex.

In the extended complex optimization algorithm, the criterion is more complex. The purpose of specialization of an extended complex is to remove nearly covered negative examples as much as possible and just covered positive examples as few as possible. Just covered positive examples of a extended complex are the examples that are covered by the complex and near the threshold. As nearly covered examples, each of the just covered examples is also associated with a value ranged from 0.5 to 1. It is computed in a similar way as nearly covered examples.

$$VJC(e, cpx) = 1 - 0.5 * \frac{SSM(e, cpx) - TH(cpx)}{TH(cpx) - PTH(cpx)}$$

The reason that a value is assigned to each just covered example is stated below. When a base

complex is specialized by removing a value of an attribute, the degree of fit of all examples that have this value is reduced. If such examples are just covered, it is very possible that their degree of fit drops below the threshold so that they are excluded from the extended complex. The just covered examples whose degree of fit is closer to the threshold are more likely to be excluded from the extended complex during the specialization.

The best value of an attribute for specialization is the value that appears in many nearly covered negative examples and few just covered positive examples. Specifically, for each attribute A, the value v_i is selected which has the smallest result of the formula

$$\frac{sp_i}{sp_i + sn_i}.$$

Where sp_i is the sum of the value of all nearly covered and just covered positive examples which have the value v_i, sn_i is the sum of the value of all nearly covered and just covered negative examples which have the value i.

5.6 Unknown Attribute Values

In real-world applications it is not unusual to encounter examples, some of whose attribute values are not known. Unknown values influence the learning system introduced above in the following three ways:

1. The evaluation of quality and potential quality of a complex may require computing the degree of fit of an event which has some unknown attribute values.
2. Unknown values may be involved in computing weights of selectors.
3. The description may be used to classify an unseen event that has unknown values.

FCLS's hybrid representation allows it to handle unknown values in a very simple way. This method is different from some other methods (e.g., Quinlan, 1989). In FCLS the result of a match between an attribute value of an event and a selector is either 1 (matched) or 0 (unmatched) for a nominal attribute, a value between 0 and 1 for a linear attribute. The result of the match between an unknown attribute value and a selector is n/m, where n is the number of values involved in the selector, and m is the total number of values in the domain of the attribute.

Chapter 6

EXPERIMENTS WITH FCLS

To evaluate the approach proposed in chapter 5 and compare it with other inductive learning approaches, a number of experiments were conducted on various domains with the Flexible Concept Learning System (FCLS) and the decision tree learning system C4.5. This chapter first outlines the experimental methods and the domains used in the experiments, then reports the details of the experimental results.

6.1 Experimental Design

The experiments described in this chapter were designed along with three dimensions. First, the experiments were performed on various domains, some of which are favorable for the approach, the others are not. Second, different learning algorithms were applied to learn concept descriptions from examples in the experiments. Finally, the performance was evaluated on the classification accuracy on unseen events and the complexity of generated concept descriptions. This section discusses these three dimensions.

6.1.1 Experimental Domains

To thoroughly test FCLS, eight domains were selected for the experiments. The domains can be divided into three categories: domains favorable to FCLS, domains unfavorable to FCLS, and real world domains. The domains in the first two categories, favorable and unfavorable domains,

are artificial. FCLS is expected to achieve better performance on both classification accuracy and description complexity than other inductive learning methods on favorable domains. Favorable domains were designed to test the effectiveness of the approach and if the approach improved performance. This category includes three domains. The domains and their experimental results are described in section 6.2.

When the FCLS approach was designed, it is hoped that the approach has the ability to automatically adjust its representation to adapt different problems so that it can be applied to a wide range of learning problems. Unfavorable domains serve as a demonstration of the applicability of FCLS. Three domains were chosen as unfavorable domains, Two of them involve small DNF formulas, and the other is 11-multiplexor. The domains and their experimental results are discussed in section 6.3.

All domains in the first two categories are artificial. The ultimate goal of this research, however, is to develop a learning approach that can be applied to solve some real world problems. Two simple and widely used real world problems, Congress voting records and Lymphatic cancer, were selected to test FCLS. Section 6.4.gives the descriptions of the problems and the experimental result on them.

6.1.2 Learning Methods

The core of the learning approach introduced in Chapter 5 consists of two algorithms: the base complex generation algorithm, and the extended complex optimization algorithm. The extended complex optimization algorithm optimizes an extended complex by adjusting its threshold and learning its weights. To deeply understand the performance of the approach, these algorithms should be evaluated independently, and compared with each other. Four methods, the base-cpx, the no-weight, the c-weight, and the d-weight, were implemented in FCLS, and involved in all experiments. The base-cpx method was realized as the base complex generation algorithm. When the base complex generation algorithm terminates, the 'best' base complex is selected. The base-cpx method generates a disjunction of base complexes as a concept description that is equivalent to the descriptions generated by AQ15 (Michalski, et al., 1986). The differences between the base complex generation algorithms and AQ algorithm were discussed in section 5.4. The base-cpx method provides the performance baseline for other methods.

The no-weight method is realized as the base complex generation algorithm plus threshold adjusting. That is, an extended complex is optimized by threshold adjusting only, no weight

learning is involved. Thus, a concept description generated by the no-weight method is represented as a set of extended complexes whose selectors are equally weighted.

The c-weight and d-weight methods were realized by the two algorithms: the base complex generation algorithm and the extended complex optimization algorithm. That is, these two methods generate an extended complex with both threshold adjusting and weight learning. Thus, each selector of an extended complex generated by these two methods has a weight that can be any value between 0 and ∞. The difference between these two methods is in the weight learning algorithms used. The c-weight method uses the count-based weight learning method, while the d-weight method uses the degree-fit-based weight learning algorithm.

In the experiments conducted, FCLS was run with all these four methods on all domains. These four methods are compared based on the experimental results to see how each method performed. To compare FCLS with other inductive learning methods, the decision tree learning system C4.5 (Quinlan, 1987) was run on the same domains.

In the domain of congressional voting records, FCLS was also run with different setting of the parameter *MIN-COVERAGE* which controls the distribution of the representation between generalized extended complexes and exemplars.

6.1.3 Performance Evaluation

The performance of FCLS was evaluated on two aspects: classification accuracy and description complexity. Classification accuracy was measured as the percentage of correct classifications made by the concept description generated by FCLS on a set of test events. The test set was either the entire space of events or a set of randomly selected events which are different from the training set. Two classification methods, the strict method and the flexible method, were used to classify test events, so the accuracy of a description was reported for both methods. Accuracy of the strict method was called S-accuracy, and accuracy of the flexible method was called F-accuracy.

Description complexity was measured by three parts: the number of extended complexes, the number selectors, and the number of exemplars involved in a description. When the parameter MIN-COVERAGE was set to 0, no exemplar was generated and the complexity was only measured by the number of extended complexes and the number selectors. The complexity of decision trees is measured by the number of leaves in a tree.

6.1.4 Experimental Method

In all experiments, FCLS was run on randomly generated training sets of various sizes. The sizes of training sets varied according to the particular learning problem; in all artificial domains, for example, training sets were of 100, 200, 300, and 400 examples. For each training set size, FCLS was run on four different randomly generated training sets. The final descriptions produced from these four runs were tested for accuracy on a set of test events, and were measured for complexity respectively. The results reported in the following sections are the average of the four runs. The parameter *MAXSTAR* that determines the width of the beam search in learning was set to 3 for all experiments. The decision tree algorithm C4.5 (Quinlan, 1987) was run on the same training sets as FCLS with and without pruning, and the decision trees produced were tested on the same set of test sets. C4.5 was run with default values of all parameters.

The experimental results are presented in the following sections as tables. Each table reports the result from one domain. Each table contains six columns: *size*, *method*, *S-accuracy*, *F-accuracy*, *#cpx/lvs*, and *#sel*. *size* is the size of a training set. In the tables, the experimental results are shown for six *methods*: C4.5 without pruning, C4.5 with pruning, FCLS with four methods: the base-cpx, the no-weight, the c-weight and the d-weight. *S-accuracy* and *F-accuracy* are the percentage of the number of correctly classified test events using the strict and the flexible classification methods, respectively. All events can be uniquely classified by a decision tree, so flexible classification method is not meaningful for C4.5. *#cpx* and *#sel* are the number of extended complexes and selectors involved in the concept descriptions of all concepts in a domain produced by FCLS. *#lvs* is the number of leaves of a decision tree produced by C4.5.

6.2 Experiments on The Domains Favorable for FCLS

The experiments described in this section were performed on three specially designed domains, called designed problem I to III. These domains were specially designed to test the novel features in FCLS, e.g., threshold adjusting and weight learning methods. It is expected that the learning approach described in Chapter 5 improves the performance over other inductive learning approaches on these domains. The following subsections describe the details of each domain, and report the experimental results.

In the experiments performed on these three domains, four training sets sizes 100, 200, 300, and 400 were used. During testing, the set of 1000 events was used as the test set. Both *MIN-COVERAGE* and *MAX-ERR-RATE* were set to 1, so no exemplar was generated and the descriptions generated by FCLS were complete and consistent.

6.2.1 Designed Domain I

This domain contains two classes, positive and negative, and 10 nominal attributes each of which has four values: 0, 1, 2, and 3. More than 1 million events were distributed between the two classes: positive and negative. The rule for distinguishing positive class from negative class has the general form of "at least k of n conditions are satisfied." Specifically, the rule is "if any 5 or more of the first 7 attributes of an event are equal to 0 or 1, then the event belongs to positive class, otherwise it belongs negative class". This rule is compactly represented by one extended complex:

$$[x_1 = 0 \vee 1]: 1$$
$$[x_2 = 0 \vee 1]: 1$$
$$[x_3 = 0 \vee 1]: 1$$
$$[x_4 = 0 \vee 1]: 1$$
$$[x_5 = 0 \vee 1]: 1$$
$$[x_6 = 0 \vee 1]: 1$$
$$[x_7 = 0 \vee 1]: 1$$
$$\text{Threshold} = 5/7 = 0.71$$

where the number following a selector is the weight of the selector. In a traditional logic type representation, 21 disjuncts are needed to describe the rule. In addition, the rule can be viewed as graded, the events with more 0 or 1 values for all first 7 attributes are more typical than the events with fewer 0 or 1 values of the first 7 attributes. For example, e_1 and e_2 are two events of positive class, and described by two vectors:

$$e_1: (0\,0\,0\,0\,0\,0\,0\,2\,3\,2)$$
$$e_2: (0\,0\,0\,0\,0\,2\,3\,3\,1\,0)$$

e_1 is a typical event of positive class, while e_2 is a less typical event, because e_1 satisfies all conditions of the positive class, while e_2 only satisfies 5 of the 7 conditions. The flexible degree of fit between e1 and the extended complex is 1, and the flexible degree of fit between e2 and the extended complex is 0. The flexible degree of fit reflects typicality of an event.

The advantage of the hybrid representation over conventional logic type representations on this type of problems is obvious. The experiments on this domain were designed to demonstrate the effectiveness of the learning algorithm that learns concept descriptions represented in the hybrid representation.

Table 7.1 shows the experimental results of this domain. A significant improvement was achieved on both classification accuracy and description complexity by the no-weight, the c-weight, and the d-weight methods over the base-cpx method and C4.5 at all training set sizes. Descriptions generated by the no-weight, the c-weight, and the d-weight methods are represented as extended complexes which are appropriate for this domain than base complexes that are used by the base-cpx method. The complexity of the descriptions generated by the base-cpx method increases with the increase in the size of training sets. In contrast, the complexity of the descriptions generated by the other three methods decreases with the increase of the size of training sets. The performance improvement obtained by the the no-weight, the c-weight, and the d-weight methods shows the effectiveness of the learning algorithm described in Chapter 5 in generating concept descriptions that have the form of "at least k of n conditions are satisfied."

The results, summarized in Table 7.1, show that the no-weight method achieved the highest classification accuracy and the lowest average complexity among all methods. This result is reasonable, because all conditions of the target concept description are equally important, that is, all selectors of the extended complex describing the target concept are equally weighted. In the no-weight method, the weights of all selectors were set to 1 without any change during learning. In the c-weight and the d-weight methods, weights were adjusted based on the training examples during learning. Although all weights learned were close to each other by the c-weight and the d-weight methods, they were not exactly equal so that the final description did not exactly describe the target concept. In real world domains, concepts are often imprecise, and weights of conditions of a concept are seldom exactly equally important, so this should not be a problem for learning imprecise concepts in the real world.

Size	Method	S-accuracy	F-accuracy	#cpx/#lvs	#sel

100	**C4.5**				
	no pruning	67.3±1.0		34.0±3.0	
	pruning	69.3±3.2		7.0±6.0	
	FCLS				
	base-cpx	67.9±4.3	73.0±4.0	6.0±1.0	33.0±9.0
	no-weight	87.5±4.7	88.3±5.9	2.3±0.7	17.8±7.2
	c-weight	80.8±4.1	83.5±3.8	5.3±1.3	39.5±10.5
	d-weight	83.1±5.4	85.9±4.9	2.5±0.5	21.8±3.2
200	**C4.5**				
	no pruning	69.6±4.1		54.3±6.7	
	pruning	72.1±2.7		16.0±9.0	
	FCLS				
	base-cpx	76.0±2.3	76.4±2.7	11.3±1.3	51.0±12.0
	no-weight	89.4±5.4	92.0±3.9	3.3±1.3	27.3±9.3
	c-weight	89.6±3.1	92.3±3.8	3.0±1.0	26.3±9.3
	d-weight	84.5±7.8	86.3±4.5	5.5±2.5	46.8±17.8
300	**C4.5**				
	no pruning	72.4±2.1		112.0±9.0	
	pruning	73.4±2.6		15.3±6.7	
	FCLS				
	base-cpx	79.9±1.7	80.5±1.9	19.5±2.5	97.5±11.5
	no-weight	99.2±2.5	99.2±2.6	2.8±2.2	18.3±12.7
	c-weight	97.4±1.9	98.2±1.8	3.5±1.5	25.5±11.5
	d-weight	98.9±1.1	99.2±0.9	2.5±0.5	16.5±3.5
400	**C4.5**				
	no pruning	74.0±3.2		157.8±9.8	
	pruning	72.9±2.8		17.5±13.5	
	FCLS				
	base-cpx	83.7±1.4	84.2±0.5	27.0±3.0	141.5±8.5
	no-weight	100.0±0.0	100.0±0.0	2.0±0.0	14.0±0.0
	c-weight	100.0±0.0	100.0±0.0	2.0±0.0	14.0±0.0
	d-weight	99.9±0.2	100.0±0.0	2.0±0.0	14.3±0.7

Table 7.1: Results from designed problem I

In all experiments, accuracy of the flexible classification method was not lower than the accuracy obtained by the strict classification method. It means that the flexible classification method correctly classified many of the multiple-match and no-match events.

The base-cpx method achieved higher accuracy than C4.5 at all sizes but the lowest size. In the training set of 100 examples, the accuracy of the strict classification method is lower than the accuracy of C4.5 with pruning. The accuracy of C4.5 shows little improvement with the increase in the size of the training set.

The no-weight method produced the exact target concept descriptions for both the positive and negative classes at three of four training sets of 300 examples, and all four training sets of 400 examples. The base complexes of the extended complexes of the two descriptions produced by the two methods with weight learning are the same as the target concepts, but their weights and thresholds are slightly different from the target concepts. Following is one of the descriptions produced by FCLS with the d-weight method.

Positive class:

[x1 = 0 v 1] : 3.42
[x2 = 0 v 1] : 3.55
[x3 = 0 v 1] : 3.18
[x4 = 0 v 1] : 3.08
[x5 = 0 v 1] : 3.27
[x6 = 0 v 1] : 2.98
[x7 = 0 v 1] : 3.48
Threshold = 0.69

Negative class:

[x1 = 2 v 3] : 3.00
[x2 = 2 v 3] : 3.20
[x3 = 2 v 3] : 2.83
[x4 = 2 v 3] : 2.68
[x5 = 2 v 3] : 3.02
[x6 = 2 v 3] : 2.94
[x7 = 2 v 3] : 2.68
Threshold = 0.41

6.2.2 Designed Domain II

Two classes, positive and negative, and eight linear attributes are involved in this domain. The domains of the eight linear attributes are same and include four values 0, 1, 2 and 3. The positive class is described by six conditions, two of which are as twice important as the other four conditions. Specifically, the positive class is expressed by one extended complex:

$[x_1 = 0 \vee 1]$: 2

$[x_2 = 0 \vee 1]$: 2

$[x_3 = 0 \vee 1]$: 1

$[x_4 = 0 \vee 1]$: 1

$[x_5 = 0 \vee 1]$: 1

$[x_6 = 0 \vee 1]$: 1

Threshold = 5/8 = 0.625

The classes are more complicated than the first designed domain in three ways. First, all attributes are linear in this domain. Second, the weights of all selectors are not all equal. Third, negative class are described by more than one extended complex. Therefore, it is much more difficult to learn the class descriptions in this domain than in the designed domain I. This domain was designed to show the effectiveness of weight learning methods, and the ability to handle linear attributes.

Table 7.2 shows the results of the experiments in this domain. Many of the results shown in Table 7.2 are similar to the results in Table 7.1. First, the accuracy of the descriptions generated by the no-weight, c-weight, and d-weight methods was an improvement over the descriptions generated by the base-cpx method, but not as large on difference as in Table 7.1. Second, the descriptions generated by the no-weight, c-weight, and d-weight methods are simpler than the descriptions generated by the base-cpx method, but not as much as in Table 7.1. Finally, when the flexible classification method was used to classify no match and multiple match events, the accuracy shows an improvement over the strict classification method, the improvement in this domain is larger than in the first domain.

The important difference between the results in Table 7.2 and the results in Table 7.1 is that the improvement in accuracy was achieved by the two methods with weight learning methods: the c-weight and d-weight. The improvement increases with the increase in the size of the training sets. The descriptions generated by the c-weight and d-weight methods are also simpler than the descriptions generated by the no-weight method. These improvements were expected and demonstrated that the weight learning methods worked properly. The results of both accuracy and complexity are close for the two different weight learning methods: the count-based and the degree-of-fit-based.

Size	Method	S-accuracy	F-accuracy	#cpx/#lvs	#sel
100	C4.5				
	no pruning	75.0±4.6		40.0±3.0	
	pruning	73.8±2.5		10.8±3.8	
	FCLS				
	base-cpx	70.7±2.1	75.6±2.3	7.8±1.2	43.0±7.0
	no-weight	77.5±2.8	81.4±4.3	4.8±0.8	33.0±4.0
	c-weight	80.2±2.8	83.8±3.0	4.5±1.5	33.5±11.5
	d-weight	78.2±3.2	83.9±4.5	4.5±1.5	35.0±12.0
200	C4.5				
	no pruning	79.3±4.0		74.5±15.5	
	pruning	81.0±5.4		13.0±3.0	
	FCLS				
	base-cpx	77.4±1.8	81.6±2.4	6.3±1.7	44.0±8.0
	no-weight	80.9±1.9	85.5±1.4	7.8±0.8	51.8±8.2
	c-weight	84.0±2.3	87.7±2.9	6.3±1.7	44.0±8.0
	d-weight	83.8±2.3	87.9±2.4	6.0±1.0	44.3±9.7
300	C4.5				
	no pruning	81.6±1.4		98.8±15.2	
	pruning	82.9±3.0		18.3±5.3	
	FCLS				
	base-cpx	78.1±2.1	81.9±1.3	14.8±1.2	84.0±6.0
	no-weight	82.8±1.0	86.3±1.1	10.8±1.2	68.3±9.7
	c-weight	87.1±1.3	90.9±1.8	7.3±0.7	49.8±8.2
	d-weight	87.3±0.6	90.9±1.7	7.8±1.2	52.5±4.5
400	C4.5				
	no pruning	82.3±2.2		123.3±13.0	
	pruning	84.7±0.3		21.5±6.5	
	FCLS				
	base-cpx	80.4±1.1	83.5±3.0	19.0±2.0	104.5±6.5
	no-weight	82.2±2.0	86.2±1.8	13.0±2.0	83.5±13.5
	c-weight	89.2±2.5	92.9±0.8	9.0±1.0	63.0±10.0
	d-weight	87.5±2.2	92.4±1.8	8.8±0.8	60.3±3.7

Table 7.2: Results from designed problem 2

S-accuracy of descriptions generated by the base-cpx method is lower than the accuracy of decision trees, and their F-accuracy is about the same as the accuracy of decision trees. As discussed in section 6.2.1, it is almost impossible to learn the exact weights of the target concept, thus the generated description is only an approximation of the target concept description. It is believed that the examples misclassified from inexact weights are usually the examples near the boundary of the concept. In the real world, the classification of these examples are often uncertain. Following is the one of the description generated by FCLS for the positive class.

[x1 = 0 v 1] : 2.76
[x2 = 0 v 1] : 2.82
[x3 = 0 v 1] : 1.56
[x4 = 0 v 1] : 1.51
[x5 = 0 v 1] : 1.28
[x6 = 0 v 1] : 1.44
[x7 = 0 v 1 v 3] : 1.11
Threshold = 0.68

The base complex of the extended complex is similar to the target concept, it includes an extra selector [x7 = 0 v 1 v 3] which has the least weight. The weights of the first two selectors are about twice that of the other four selectors.

6.2.3 Designed Domain III

Designed domain III contains 15 nominal binary attributes, and two classes: positive and negative. The events of the positive class are described by two extended complexes, each of which consists of 6 selectors, two of which are as twice important as the other four. The positive class is described by the disjunction of the following two extended complexes:

Complex 1:
[x_1 = 0]: 2
[x_2 = 0]: 2
[x_3 = 0]: 1
[x_4 = 0]: 1
[x_5 = 0]: 1
[x_6 = 0]: 1
Threshold = 5/8 = 0.625

Complex 2:
[x_7 = 0]: 2
[x_8 = 0]: 2
[x_9 = 0]: 1
[x_{10} = 0]: 1
[x_{11} = 0]: 1
[x_{12} = 0]: 1
Threshold - 5/8 = 0.625

Size	Method	S-accuracy	F-accuracy	#cpx/#lvs	#sel
100	C4.5				
	no pruning	72.9±3.7		14.8±2.8	
	pruning	75.8±1.6		8.3±1.7	
	FCLS				
	base-cpx	73.4±2.9	77.3±3.4	11.3±1.3	46.0±5.0
	no-weight	76.9±2.5	79.3±3.1	6.5±0.5	52.3±7.3
	c-weight	80.4±3.3	82.8±3.2	5.8±1.2	49.5±7.5
	d-weight	79.4±3.2	82.2±3.2	5.3±0.7	45.3±7.7
200	C4.5				
	no pruning	78.9±3.5		25.3±5.3	
	pruning	79.2±2.8		12.3±3.7	
	FCLS				
	base-cpx	77.6±0.6	79.3±0.6	18.0±2.0	80.5±12.5
	no-weight	80.0±1.0	82.6±2.9	12.0±1.0	93.3±17.7
	c-weight	86.6±3.1	89.3±4.2	11.5±3.5	95.3±31.7
	d-weight	84.4±6.2	88.2±4.5	9.5±2.5	85.0±21.0
300	C4.5				
	no pruning	81.7±1.5		35.3±5.3	
	pruning	81.6±1.0		19.0±4.0	
	FCLS				
	base-cpx	80.0±0.9	81.8±1.8	22.5±1.5	103.0±16.0
	no-weight	82.1±1.6	85.4±2.9	16.0±2.0	126.3±15.7
	c-weight	88.8±0.9	90.8±0.8	12.3±6.7	105.8±61.2
	d-weight	86.6±2.4	89.1±1.8	14.3±6.3	127.0±57.0
400	C4.5				
	no pruning	83.4±2.5		42.8±2.8	
	pruning	82.9±1.4		21.5±4.5	
	FCLS				
	base-cpx	81.4±0.4	83.2±2.3	26.3±2.7	123.5±13.5
	no-weight	84.0±1.2	87.5±1.3	21.5±4.5	170.8±37.8
	c-weight	88.6±2.4	91.8±1.2	21.0±6.0	185.5±51.5
	d-weight	88.5±2.4	91.4±2.5	19.5±8.5	174.3±70.3

Table 7.3: Results from the designed domain III

This domain was designed to test if FCLS works well when a target concept is described by more than one extended complexes, and also to evaluate the two weight learning methods: the count-based and the degree-of-fit-based. Table 7.3 gives the experimental results from this domain.

As shown, the accuracy (both S-accuracy and F-accuracy) are improved with the no-weight, c-weight, and d-weight methods over the base-cpx method. As expected, the accuracy of the descriptions generated by the two methods with weight learning was also improved over the no-weight method. The number of complexes involved in descriptions is reduced in the order of the base-cpx method, the no-weight method, and the methods with weight learning, but the number of selectors in descriptions slightly increases with the same order. Because the optimization phase adds more selectors to an extended complex, the extended complexes produced by the no-weight, c-weight and d-weight methods involve more selectors than the base complexes generated by the base-cpx method.

Both the accuracy and complexity with the two different weight learning methods, count-based and degree-of-fit-based, are similar. The degree-of-fit-based method was designed to improve the performance when target concepts include two or more extended complexes. The experimental results from this domain showed that the d-weight learning method does not work as it is expected, and suggest that other weight learning methods should be developed.

At the training size 300 and 400, the descriptions generated on some of the training sets do include the complexes that are similar to the two extended complexes of the target concept. For two reasons, the accuracy of the descriptions generated is difficult to improve when it is over 90%. First, the weights of the two extended complexes in the target concept are defined precisely, but the weight learning methods can only generate weights that are close to these precisely defined weights. Second, the description of the negative class is highly disjunctive, and is very difficult to learn, but in classification of test events, both the positive and negative descriptions were applied. As discussed in section 6.2.2, the misclassified examples are usually near the boundaries of the concepts, and are less typical, and their classification is fuzzy for a real world problem.

6.2.4 Summary of The Experiments on Favorable Domains

This section described the experiments on three specially designed domains on which the hybrid representation and its associated learning algorithm described in Chapter 5 are expected to

perform better than some other inductive learning methods, e.g., decision tree learning method, and the methods using a pure logic-based representation. The experimental results from all of the three domains are consistent with the expectation, and show the improvement on both accuracy and complexity over the decision tree learning algorithm C4.5 and the base-cpx method, a method using a pure logic-based representation. As expected, the weight learning methods improved the accuracy over the method without weight learning on the domain II and the domain III, but the two weight learning methods, count-based and degree-of-fit-based, show no difference. The flexible classification method achieved higher accuracy than the strict classification method on all experiments.

6.3 The Domains Unfavorable to FCLS

This section describes the experiments from three domains: 11-multiplexor, 3-term 3DNF and 4-term 3DNF for which FCLS was not specially designed. The hybrid representation has no advantage over logic type representations in representing the concepts involved in these domains. Adversely, the hybrid representation increases difficulty to learn these concepts because of the less representational bias enforced by the representation. When FCLS was designed, it was hoped that it could be applied to a wide range of problems. That is, it should adjust its representation to adapt to the given problem. The adjusting of its representation in FCLS is to find the optimum distribution between the base complex and the similarity measure of an extend complex. In these domains, the optimum distribution is that each extended complex should be equivalent to its base complex, that is, its threshold should be equal to 1. The experiments on these domains are designed to test the applicability of FCLS.

Four training sets sizes 100, 200, 300 and 400 were used in the experiments on these domains. During testing, the entire event space was used as the test set in the problem 11-multiplexor, and 1000 testing events were used for the problems of 3-term 3DNF and 4-term 3DNF.

6.3.1 11-Multiplexor

The "Multiplexor" problems were first used by Wilson (1987a, b) to test his system Boole which learns concepts expressed as classification rules. Later, this family of tasks was used by Quinlan (1988) to compare genetic classifiers with decision-tree classifiers.

The "Multiplexor" is a family of tasks in which an object consists of n "address" bits and 2^n "data" bits. An object belongs to the positive concept if the particular data bit indicated by the address bits is "on". A member of this family of tasks is named by the total number of bits (number of "address" + number of "data") involved. For the task F_6 the address bits are x_0 and x_1 and the data bits x_2 through x_5. If x_0 and x_1 are both off, for instance, the address is 0 and the object is a member of the positive class iff the 0th data bit (x_2) is on. The rule for the positive concept is:

$$[x_0 = 0]\,[x_1 = 0]\,[x_2 = 1] \vee$$
$$[x_0 = 0]\,[x_1 = 1]\,[x_3 = 1] \vee$$
$$[x_0 = 1]\,[x_1 = 0]\,[x_4 = 1] \vee$$
$$[x_0 = 1]\,[x_1 = 1]\,[x_5 = 1]$$

The rule for negative concept is:

$$[x_0 = 0]\,[x_1 = 0]\,[x_2 = 0] \vee$$
$$[x_0 = 0]\,[x_1 = 1]\,[x_3 = 0] \vee$$
$$[x_0 = 1]\,[x_1 = 0]\,[x_4 = 0] \vee$$
$$[x_0 = 1]\,[x_1 = 1]\,[x_5 = 0]$$

F_{11} has 3 "address" bits and 8 "data" bits. The size of the instance space is 2048. Each target concept description of two concepts consists of 8 disjuncts (32 conjuncts).

The results of the experiments on 11-multiplexor are summarized in Table 7.4. The base-cpx method performed much better than the two methods with weight learning at the sizes of 100, and 200, but the methods with weight learning performed better than the base-cpx method at size of 300, and 400. The above behavior is explained below. As indicated above, the hybrid representation is more powerful than conventional logic based representations, therefore, enforces less representational bias. For small training sets, because the set of training examples is not large enough, many disjuncts that should not merge are incorrectly merged into extended complexes that have higher quality than the unmerged disjuncts. Large training sets prevent extended complexes from incorrectly merging disjuncts. The reason for that the methods with weight learning performed better on large training sets will be discussed later.

Except for the size of 100, similar accuracy has been achieved by the base-cpx method and the no-weight method. At the sizes of 300, and 400, the descriptions generated by the two methods are almost same and very similar to the target concept descriptions. In other words, the extended complexes generated by the no-weight method are equivalent to their base complexes, and their threshold is 1. The no-weight method works well in adjusting representation for a given problem.

Size	Method	S-accuracy	F-accuracy	#cpx/#lvs	#sel
100	C4.5				
	no pruning	66.8±9.8		17.5±3.5	
	pruning	65.3±9.7		12.0±4.0	
	FCLS				
	base-cpx	74.7±4.4	76.7±6.2	14.5±0.5	64.5±3.5
	no-weight	69.(±2.4	71.6±1.5	16.0.±4.0	101.0±30.0
	c-weight	68.9±1.8	70.8±3.0	13.5±2.5	90.0±18.0
	d-weight	68.2±1.3	70.4±1.3	13.0±3.0	96.3±21.3
200	C4.5				
	no pruning	76.3±10.1		35.0±6.0	
	pruning	76.4±3.3		26.0±6.0	
	FCLS				
	base-cpx	92.0±5.9	92.3±4.5	16.3±1.3	68.0±7.0
	no-weight	89.3±6.1	93.5±5.9	18.5±1.5	93.5±22.5
	c-weight	82.7±2.4	84.6±3.6	21.3±6.3	131.8±39.8
	d-weight	82.6±4.3	85.5±4.4	18.8±3.8	125.5±30.5
300	C4.5				
	no pruning	90.9±6.0		37.5±8.5	
	pruning	87.1±6.7		29.5±11.5	
	FCLS				
	base-cpx	93.8±2.0	93.4±1.7	16.3±1.3	67.3±6.3
	no-weight	93.4±4.1	93.0±3.5	16.3±1.3	68.3±7.3
	c-weight	94.9±4.3	95.9±3.9	22.3±2.3	124.5±22.5
	d-weight	92.1±4.6	92.5±5.0	20.8±4.8	129.0±37.0
400	C4.5				
	no pruning	93.8±5.5		51.3±16.7	
	pruning	91.0±5.9		27.5±10.5	
	FCLS				
	base-cpx	94.9±3.5	93.6±4.8	16.5±1.5	69.5±9.5
	no-weight	94.9±3.5	93.5±4.9	16.5±1.5	69.5±9.5
	c-weight	97.5±2.5	97.7±3.2	20.8±5.2	105.0±36.0
	d-weight	96.5±1.7	96.7±2.7	22.0±4.0	125.0±24.0

Table 7.4: Results from 11-multiplexor

About half of the 8 disjuncts involved in the target concept were correctly generated by FCLS with the c-weight and d-weight methods at the size 400. The other disjuncts were generated as extended complexes whose threshold is smaller than 1. It is surprising that 100% accuracy (for both the strict and flexible classification methods) was produced for one training set, while even more than half of the 8 disjuncts of the targets were represented by extended complexes with the threshold smaller than 1. One interesting point is that some extended complexes with a threshold less than 1 are equivalent to some of the 8 disjuncts. For example, the following extended complex was generated by FCLS.

[x1 = 1] : 0.95
[x2 = 1] : 1.02
[x3 = 1] : 1.05
[x6 = 0] : 0.86
[x11 = 1] : 1.31
threshold = 0.83

The strict coverage of this extended complex is the same as the disjunct (base complex):

[x1 = 1] [x2 =1] [x3 = 1] [x11 =1]

In the extended complex, the weight of the selector [x11 = 1] is 0.86 which is the least among the 5 selectors. The summation of the five weights is equal to 5.19, and the threshold is 0.83 = 0.86/5.19. This means that if only the selector [x6 = 0] is not satisfied by an event, the degree of fit of the event to the complex is 0.83, so it is still strictly covered by the complex, but if any of the other four selectors is not satisfied by an event, the degree of fit of the event to the complex is less than the threshold, so the event is not strictly covered by the complex. The selector [x11 = 1] is redundant when the strict classification method is applied. This example shows another advantage of the hybrid representation, that is it can remove redundant conditions from a disjunct by assigning a small weight to the condition. This advantage is important in learning algorithms in which backtracking is not allowed. This advantage is also one possible explanation for the fact that the accuracy of the descriptions generated by the methods with weight learning is better than the accuracy of the descriptions generated with the base-cpx method at the size 400.

Unlike the three domains described in section 6.2, the descriptions generated by the methods with weight learning are more complicated than the ones generated by the base-cpx and the no-weight methods. There are three possible reasons for this. First, this domain leaves no room for the hybrid representation to merge any disjuncts. Second, extended complexes usually include

more selectors. Third, the extended complexes generated at the beginning of learning cover more examples than a disjunct of the target concept, these extended complexes broke the event space into small pieces so that more extended complexes have to be generated to cover the remain uncovered examples.

The results in the table show that the accuracy of the descriptions produced by FCLS with all methods are higher than the accuracy of the decision trees produced by C4.5. At the sizes of 100, and 200, the accuracy of the descriptions produced by FCLS is much better than the accuracy of the decision trees produced by C4.5, at the sizes of 300, and 400, the accuracy of the descriptions generated by FCLS is slightly better.

6.3.2 3-term 3DNF Boolean Function

A l-term kDNF boolean function (Pagallo and Haussler, 1988) consists of l disjunctive terms of at most k conjuncts each. This section describes the experiments on a 3-term 3DNF boolean function, and next section describes the experiments on a 4-term 3DNF boolean function. The 3-term 3DNF boolean function involved in the experiments consists of three disjuncts, each of which has exactly 3 conjuncts. No attribute is shared by two disjuncts, and the total number of attributes in this domain is 12.

Table 7.5 shows the experimental results. As shown in the table, the accuracy of descriptions generated by the two methods with weight learning is worse than the other two methods the base-cpx and the no-weight method, especially at small training sizes. As in the domain of 11-multiplexor, this decreased accuracy is due to the week representational bias enforced by the hybrid representation. In spite of the problem, the accuracy of the methods with weight learning is still comparable with the accuracy of C4.5, and at size 400, the target concept description was generated on some training sets by the methods with weight learning. The results suggest that stronger bias should be applied during learning especially when the training set is small.

Another important result is that the accuracy of descriptions generated by the base-cpx method is much higher than the accuracy of the decision trees obtained from C4.5. This may be partially attributed to the Probabilistic Looking Ahead Criterion used in the base complex generation algorithm. In the base-cpx method, only the base complex generation algorithm is applied to generate a complex. In this algorithm, each complex is evaluated by the base complex potential quality evaluation function in which the probabilistic looking ahead criterion is used. The potential quality of a complex involves the current quality of the complex as well as the potential

improvement of the complex. The probabilistic looking ahead criterion computes the potential improvement of a complex based on the relevance of the attributes involved in the complex. The more relevant the attributes, the more potential improvement. In this way, the relevance of an attribute is not evaluated independently, instead it is evaluated together with other attributes. When a concept has a few distant disjuncts, it is not reliable to evaluate the relevance of attributes independently, they must be evaluated jointly. A one-step looking-ahead criterion has been implemented in FCLS. That is, the relevance of attributes is evaluated based on only two joint attributes. When each disjunct consists of only a few conjuncts, e.g., 3 in this domain, this one step looking ahead criterion may work well. Another possible reason for the higher accuracy achieved by the base-cpx method is that a decision tree learning algorithm performs the hill climbing search, while the base complex generation algorithm performs the beam search with the width 3.

At size 100, the target concept was generated on three of the four training sets by the base-cpx method, the description generated from the other training set is very close to the target concept, and at the sizes 200, 300, and 400, the target concept was always produced by the base-cpx method. From the table, one can see that the accuracy is not 100%. This is because in a rule learning algorithm, the descriptions for both positive and negative concepts are learned and used to classify unseen events. In this domain, the description of negative concept is highly disjunctive and includes 27 disjuncts, so it is very difficult to learn the description for the negative concept. This is why it is so difficult to achieve 100% accuracy. If only the description of the positive concept were used for classification, the accuracy would be 100%.

Another important and interesting result is that the accuracy obtained through the no-weight method is close to the accuracy of the base-cpx method, the F-accuracy of the no-weight method is even slightly better than the base-cpx at the sizes 200, 300, and 400. As the base-cpx method, the no-weight method generated the exactly same description of the target concept at sizes 200, 300, and 400 as well. This interesting result suggests that weight learning methods should not be applied to learn well and precisely defined concepts. This result also shows that the algorithm works fine in adjusting thresholds of complexes to adapt to the learning problems given.

Finally, the flexible classification method has significantly improved the accuracy over the strict classification method, and the averaged improvement is larger than 5%. Many no match and multiple match events were created by the generated incorrect description of the negative concept. These no match and multiple match events cannot be classified by the strict method, while many of them are correctly classified by the flexible method.

Size	Method	S-accuracy	F-accuracy	#cpx/#lvs	#sel
100	C4.5				
	no pruning	82.9±2.4		12.5±1.5	
	pruning	82.7±2.8		9.3±1.7	
	FCLS				
	base-cpx	85.5±1.9	91.3±4.1	9.3±0.7	31.3±3.7
	no-weight	83.0±5.5	89.1±6.9	8.0±0.0	45.3±4.7
	c-weight	80.7±7.5	82.9±6.0	6.8±0.8	50.8±12.8
	d-weight	79.7±5.4	82.2±5.5	6.5±0.5	47.8±9.2
200	C4.5				
	no pruning	88.1±3.4		21.5±4.5	
	pruning	85.8±2.4		13.8±1.8	
	FCLS				
	base-cpx	92.2±1.0	97.1±2.1	11.5±0.5	40.0±3.0
	no-weight	88.4±1.2	98.5±2.6	10.8±0.8	65.8±5.2
	c-weight	88.6±0.6	91.9±3.7	9.3±1.7	63.8±15.8
	d-weight	88.3±1.7	90.0±3.2	8.5±1.5	57.3±7.3
300	C4.5				
	no pruning	94.0±2.9		25.3±4.7	
	pruning	89.6±5.0		16.3±4.7	
	FCLS				
	base-cpx	94.1±0.5	97.5±1.3	12.8±0.8	44.3±2.7
	no-weight	90.0±2.2	98.9±2.3	12.8±0.8	74.5±9.5
	c-weight	90.7±1.9	95.3±1.6	9.5±1.5	60.8±22.2
	d-weight	90.7±1.8	94.0±3.2	10.3±1.7	72.5±23.5
400	C4.5				
	no pruning	95.7±1.6		30.0±4.0	
	pruning	94.6±1.8		24.5±0.5	
	FCLS				
	base-cpx	94.8±1.0	98.3±1.9	12.8±0.8	43.5±3.5
	no-weight	92.4±1.6	99.1±0.7	13.2±0.7	64.3±11.7
	c-weight	91.3±2.1	95.8±3.4	11.0±1.0	77.0±17.0
	d-weight	93.3±1.3	95.9±3.2	11.3±1.7	71.5±9.5

Table 7.5: Results from the 3-term 3DNF boolean function

6.3.3 4-term 3DNF Boolean Function

The target concept of the experiments described in this section is a 4-term 3DNF boolean function that consists of four disjuncts, each with three conjuncts. No attribute is shared by different disjuncts, the total number of attributes is 16.

Table 7.6 gives the results of these experiments. The results are consistent with the results from the 3-term 3DNF reported in the section 6.3.2. The accuracy of the descriptions generated by the base-cpx and the no-weight methods is much better than the other methods. The accuracy of the descriptions generated by the methods with weight learning is slightly better C4.5. The accuracy is much greater with the flexible classification method than the strict classification method. The description of the target concept was generated on many training sets with the base-cpx and no-weight methods. The explanations for these results could be found in the section 6.3.2.

6.3.4 Summary of The Experiments on Unfavorable Domains

The hybrid representation used in FCLS has no advantage over conventional logic based representations on all three of the domains described in this section. Instead it increases the difficulty in learning the descriptions for the concepts in these domains because of the weaker bias enforced by the hybrid representation. All experimental results in this section show this difficulty, especially for the small number of training examples. It seems the main source of the difficulty is weight learning, because the results obtained from the no-weight method is similar to the base-cpx method and are much better than the methods with weight learning. The no-weight method works well in adjusting the representation for a given problem, but the weight learning methods does not. It suggests that the methods with weight learning should be to learn flexible or imprecise concepts.

The base-cpx method works very well on all of the three domains, this may be partially attributed to the quality evaluation function used in the base complex generation algorithm. The flexible classification method significantly improved the accuracy in the domains of 3-term 3DNF and 4-term 3DNF.

It seems that the approach described in Chapter 5 is less sensitive to the number of disjuncts involved in a target concept description than the inductive learning systems that use conventional logic type representations. The performance of FCLS does not degrade with an increase in the number of disjuncts of a concept description.

Size	Method	S-accuracy	F-accuracy	#cpx/#lvs	#sel
100	C4.5				
	no pruning	72.6±9.6		15.8±2.8	
	pruning	76.3±5.7		9.8±3.8	
	FCLS				
	base-cpx	81.4±2.4	92.5±4.5	11.3±1.7	41.5±6.5
	no-weight	76.8±6.8	83.6±15.2	9.0±2.0	64.8±15.2
	c-weight	75.0±4.5	77.4±6.4	9.5±2.5	74.3±24.7
	d-weight	71.2±2.0	74.5±0.9	6.3±0.7	58.0±10.0
200	C4.5				
	no pruning	77.8±2.7		24.5±2.5	
	pruning	78.2±1.1		16.3±5.3	
	FCLS				
	base-cpx	85.4±0.9	96.3±1.1	16.8±2.2	71.8±14.2
	no-weight	84.7±1.6	95.6±6.0	17.0±6.0	82.0±44.0
	c-weight	80.2±2.6	83.8±3.0	9.0±1.0	76.3±7.3
	d-weight	78.4±0.6	82.0±1.1	9.3±1.3	82.3±7.7
300	C4.5				
	no pruning	83.7±2.1		31.8±4.8	
	pruning	83.3±3.4		21.5±2.5	
	FCLS				
	base-cpx	87.5±2.3	98.6±3.6	20.5±0.5	94.5±9.5
	no-weight	84.6±0.4	98.4±4.1	13.3±0.7	90.8±4.8
	c-weight	85.2±0.3	90.6±4.0	11.3±1.3	89.0±7.0
	d-weight	83.5±4.7	89.9±5.8	11.3±0.7	86.8±16.2
400	C4.5				
	no pruning	87.5±5.0		39.0±3.0	
	pruning	86.4±2.8		24.0±2.0	
	FCLS				
	base-cpx	89.5±1.5	100.0±0.0	22.3±1.3	104.3±9.3
	no-weight	85.8±4.0	98.4±4.4	16.3±0.7	103.8±16.2
	c-weight	85.5±0.8	91.5±2.0	13.5±1.5	115.0±23.0
	d-weight	84.5±2.8	90.5±2.1	13.0±1.0	109.3±26.7

Table 7.6: Results from the 4-term 3DNF boolean Function

6.4 Real World Domains

This section reports the experimental results from two real world domains: congressional voting records and lymphatic cancer. These two domains are selected to test the applicability of FCLS to real world problems. In addition to the experiments using different methods, the experiments were done with varying values of the parameter MIN-COVERAGE in the domain of congressional voting records so that the performance could be compared between the descriptions with different distributions between generalized extended complexes and exemplars.

6.4.1 Congressional Voting Records

The data used in this section is different from the congressional voting data used in chapter 4, and includes more examples. This domain includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes in the 1984. Each key votes is a binary attribute, the two values of each attribute is *voted for and voted against*. The goal of the learning task is to generate rules to distinguish Republican Representatives from Democratic Representatives. The total number of examples is 435 of which 232 have all the attribute values and 203 have at least one missing attribute value. Four training sets for each of four training sizes (50, 100, 150, and 200) were formed by randomly drawing examples from 300 examples, the other 135 examples were used as testing set. The results reported in Table 7.7 represent averages of tests run on all four training sets.

Accuracy with all methods including C4.5 is similar. One method may be slightly better than some other methods at one training size, but worse at some other training size. The reason for such results is that the examples in one concept are very concentrated so that it is easy to generate one or two disjuncts to describe the target concept from training examples. For example, at the size 50, C4.5 generated a decision tree with only 3 nodes on all four runs. A decision tree with 3 nodes is equivalent to two disjuncts, one for republican and one for democratic, and each consists of only one condition. The averaged accuracy of such simple descriptions is close to 90%.

The number of complexes generated as descriptions is close for all methods, but the number of selectors involved in descriptions generated by the no-weight, the c-weight, and the d-weight methods is larger than the number of selectors generated by the base-cpx method. As stated in chapter 5, the base complex of an extended complex captures the basic principles, and includes all relevant conditions, while the descriptions generated by C4.5 are discriminant descriptions, and only include the conditions that are enough to discriminate two classes.

Size	Method	S-accuracy	F-accuracy	#cpx/lvs	#sel
50	**C4.5**				
	no pruning	87.4±7.9		2.0±0.0	
	pruning	88.7±6.7		2.0±0.0	
	FCLS				
	base-cpx	86.9±7.3	91.9±6.2	2.5±1.5	5.3±3.7
	no-weight	90.4±4.9	90.6±4.7	2.0±0.0	6.3±4.3
	c-weight	90.1±5.2	88.7±6.6	2.0±0.0	5.8±3.8
	d-weight	86.6±5.9	88.6±2.4	2.0±0.0	7.8±0.8
100	**C4.5**				
	no pruning	90.5±3.3		3.8±0.8	
	pruning	92.6±5.4		2.8±1.2	
	FCLS				
	base-cpx	88.9±2.8	94.1±2.5	3.8±1.8	12.0±9.0
	no-weight	90.2±2.4	93.0±3.1	2.0±0.0	13.5±5.5
	c-weight	91.2±5.1	89.2±6.5	4.0±2.0	22.0±15.0
	d-weight	90.7±5.0	91.9±7.3	3.0±2.0	18.5±7.5
150	**C4.5**				
	no pruning	89.8±7.0		5.0±3.0	
	pruning	93.3±6.1		2.3±0.7	
	FCLS				
	base-cpx	90.6±4.0	92.8±7.1	5.0±3.0	16.3±11.7
	no-weight	92.3±1.8	91.0±2.5	3.3±1.3	20.8±7.8
	c-weight	91.0±1.9	91.0±1.6	4.8±1.8	27.8±12.2
	d-weight	92.3±1.8	91.0±2.5	4.5±0.5	28.8±6.8
200	**C4.5**				
	no pruning	92.6±1.7		6.0±2.0	
	pruning	94.6±1.5		3.8±1.8	
	FCLS				
	base-cpx	90.4±4.7	92.0±3.7	4.3±0.7	12.3±5.7
	no-weight	92.0±1.9	92.7±2.6	4.8±3.2	27.5±13.5
	c-weight	93.1±3.7	93.9±3.0	5.3±1.3	31.5±5.5
	d-weight	93.4±4.0	94.1±2.7	5.5±1.5	36.0±15.0

Table 7.7: Results from the congressional voting records

An experiment was conducted to evaluate the performance of the concept descriptions with different distributions between generalized extended complexes and exemplars. In this experiment, the parameter *MIN-COVERAGE* was set to different values 0.0, 0.05, 0.1 and 1. This parameter controls the distribution of descriptions between extended complexes and exemplars. When *MIN-COVERAGE* = 0, descriptions are totally represented by extended complexes. When *MIN-COVERAGE* = 1, a description is represented by either one extended complex or a set of exemplars.

Table 7.8 gives the result of this experiment. The exemplar-based method worked very well in this domain, and the descriptions represented as a set of exemplars have about the same accuracy as the pure generalized descriptions. Furthermore, only a small number of exemplars was selected for each description. When some small extended complexes of the generalized descriptions were replaced by exemplars, the accuracy always improved. The reason for this improvement is that these small extended complexes were overgeneralized and committed more errors than large complexes. In such mixed descriptions, each concept is described by one extended complex and a few exemplars. The extended complex covers most of the positive examples, and the exemplars can be considered as exceptions.

Size	MIN-COVERAGE	F-accuracy	#cpx	#sel	#exem
100	0.0	89.2±6.5	4.0±2.0	22.0±15.0	0.0±0.0
	0.05	91.8±5.4	2.3±0.7	17.8±6.8	0.5±0.5
	0.1	92.1±4.9	2.0±0.0	14.8±4.8	1.0±0.0
	1	90.0±5.7	0.0±0.0	0.0±0.0	9.0±3.0
150	0.0	91.0±1.6	4.8±1.8	27.8±12.2	0.0±0.0
	0.05	93.8±3.7	2.0±0.0	18.3±2.7	3.5±0.5
	0.1	94.5±1.8	2.0±0.0	20.3±6.3	4.8±4.2
	1	90.8±4.1	0.0±0.0	0.0±0.0	10.0±3.0
200	0.0	93.9±3.0	5.3±1.3	31.5±5.5	0.0±0.0
	0.05	94.4±2.8	2.5±0.5	25.0±10.0	6.3±2.7
	0.1	93.1±7.8	2.0±0.0	17.3±4.3	8.0±5.0
	1	93.5±4.8	0.0±0.0	0.0±0.0	13.8±4.2

Table 7.8: Results of the congressional voting records with different values of *MIN-COVERAGE*

6.4.2 Lymphatic Cancer

The lymphatic cancer data is characterized by 18 attributes and 4 diagnostic classes. Data of 148 patients were available. The set of attributes was complete, i.e., was sufficient for having all learning examples consistent. This means that examples of any two classes were always different. 25, 50, 75, and 100 examples were randomly selected for learning respectively, the remaining 48 examples were used as testing set. All experiments were performed four times. The results describing the average of the four runs are presented in Table 7.9.

Both the "c-weight" and "no-weight" descriptions gave the better accuracy and are simpler than the "base-cpx" descriptions except the size of 25. All methods of FCLS have achieved much better accuracy (F-accuracy) than C4.5 especially on the sizes of 75 and 100. From the table one can see that the flexible classification method significantly improved accuracy in all experiments over the strict classification method. The flexible classification method classifies non-typical events that are strictly covered by either more than one concept descriptions or no concept description. One reason for this behavior is that the descriptions generated are the most specific, and leave much event space uncovered. Another possible reason seems to be that the domain includes many non-typical examples that cannot be classified by the strict method and are correctly classified by the flexible method.

Another striking result is that the concept descriptions generated by the c-weight method for all training sets are described by one extended complex. These descriptions strictly cover most of the positive examples but not all, the remaining positive examples being flexibly covered by these descriptions. It seems that these descriptions capture the basic principles of these concepts. Such one extended complex descriptions leave no room to apply an exemplar-based approach.

6.4.3 Domains Used to Test POSEIDON

POSEIDON was experimentally applied to two different problem domains: labor management contracts and congressional voting data. The congressional voting data is different from the data used in section 6.4.1. To empirically compare POSEIDON and FCLS, FCLS was run on these two domains with the exactly same training and testing data. Table 7.10 shows the results. In the table, the terminology used in POSEIDON is used here. ACC is the classification accuracy on testing data, in FCLS the flexible classification method was used to classify testing examples. #rules (#conds) is the number of rules (conditions) involved in the descriptions generated. Rules and conditions are equivalent to complexes and selectors, respectively.

Size	Method	S-accuracy	F-accuracy	#cpx/#lvs	#sel
25	**C4.5**				
	no pruning	66.5±11.2		15.8±11.8	
	pruning	69.7±14.4		6.0±2.0	
	FCLS				
	base-cpx	66.1±9.4	77.1±12.2	6.0±0.0	16.3±1.3
	no-weight	63.7±5.3	71.5±5.7	4.0±0.0	18.5±2.5
	c-weight	63.4±12.7	73.8±3.9	4.0±0.0	15.0±1.0
50	**C4.5**				
	no pruning	78.3±4.4		23.8±12.8	
	pruning	72.5±5.5		7.5±2.5	
	FCLS				
	base-cpx	71.2±0.6	79.4±5.4	7.3±1.7	26.0±7.0
	no-weight	75.1±2.6	83.0±3.0	4.0±0.0	24.8±2.8
	c-weight	75.9±10.0	81.0±9.3	4.0±0.0	26.8±5.2
75	**C4.5**				
	no pruning	76.0±8.6		34.3±11.7	
	pruning	76.5±4.9		5.5±1.5	
	FCLS				
	base-cpx	77.4±3.4	83.5±6.5	9.5±1.5	42.8±11.8
	no-weight	79.3±13.7	83.8±7.9	5.0±0.0	34.0±7.0
	c-weight	79.5±6.8	83.5±1.1	4.0±0.0	26.3±3.3
100	**C4.5**				
	no pruning	76.4±6.0		50.0±8.0	
	pruning	76.5±7.1		10.8±11.2	
	FCLS				
	base-cpx	80.4±6.9	84.1±5.9	10.5±1.5	53.0±9.0
	no-weight	78.2±14.2	86.0±7.9	6.0±1.0	46.3±7.3
	c-weight	78.0±7.1	85.7±12.0	4.0±0.0	29.3±3.7

Table 7.9: Results of the Lymphatic cancer

POSEIDON achieved higher accuracy in the domain of labor management contracts, while FCLS improved in accuracy over POSEIDON in the domain of congressional voting records. The higher accuracy achieved by POSEIDON in the domain of labor management contracts may be

partially due to the ICI rules generated in POSEIDON. One of the reasons for the higher accuracy obtained by FCLS is the representation and the learning method used in FCLS. In both POSEIDON and FCLS, a description is generated in two phase: description generation phase and description optimization phase. The major difference between these two systems is the algorithms used in the optimization phase. In POSEIDON, a description is optimized by removing some of its components (selectors or complexes), while in FCLS a complex is optimized by adding more selectors. That is why the descriptions generated by FCLS include more selectors, even they include fewer complexes. In the domain of congressional voting records, most of the fourteen attributes are relevant to distinguish republicans from democrats, but only some of the conditions involved in these relevant attributes need to be satisfied in order to distinguish a republican congressman from a democratic congressman. In the description generated by FCLS, an extended complex includes all of these relevant conditions, while in the description generated by POSEIDON, only some of the conditions are included. It seems that FCLS is more appropriate for this domain.

	Labor Contract			Congress Voting		
	ACC	#rules	#conds	ACC	#rules	#conds
POSEIDON	90%	9	12	92%	10	21
FCLS						
base-cpx	83%	7	19	89%	5	29
no-weight	85%	2	13	96%	2	26
c-weight	86%	2	13	93%	3	20

Table 7.10: Results from two domains used in POSEIDON

6.4.4 Summary of Real World Domains

This section described the experiments on two real world domains: congressional voting records and Lymphatic cancer. In the domain of congressional voting records, there is little difference in accuracy between all methods, but the description generated with the no-weight, the c-weight, and the d-weight methods are more concise than the base-cpx method. An experiment was conducted on the congressional voting records with the different settings of the parameter *MIN-COVERAGE*. In this experiment, the descriptions that consists of both extended complexes and exemplars obtained higher accuracy than the descriptions represented as only exemplars or

extended complexes. Both accuracy and conciseness have been improved with the no-weight and c-weight methods on the domain of Lymphatic cancer.

6.5 Summary of Experimental Results

The results of all experiments from three types of domains are summarized in the following paragraphs. In general, the results support the approach presented in chapter 5.

In the domains favorable to FCLS, all results are consistent with what were expected. The methods with weight learning improved the accuracy and complexity on the method with only threshold adjusting that always resulted in a large improvement in both accuracy and conciseness over the base-cpx method. The flexible classification method showed an improvement in accuracy. These experimental results empirically proved the claim that the hybrid representation and its associated algorithm described in chapter 5 work better than many current existing inductive learning methods on concepts with irregular and imprecise boundaries. The results showed no difference between the two weight learning methods.

The results from the domains unfavorable to FCLS showed that the base-cpx and the no-weight methods achieved the best accuracy among all methods. The base-cpx method performed much better than C4.5 because of the use of the two evaluation functions. The performance of the method with only threshold adjusting (the no-weight method) is close to the base-cpx method, and the descriptions generated by the no-weight method are very similar to the descriptions generated by the base-cpx method. This behavior shows that the no-weight method works well in adjusting the distribution of descriptions between the base_complex and the similarity measure. The two methods with weight learning degraded the performance, especially on the small number of training examples. It seems that it is not appropriate to apply weight learning methods to learn well defined concepts.

The result from the two real world domains demonstrated the applicability of FCLS on real world problems. In Lymphatic cancer domain, the no-weight and c-weight methods did improve both accuracy and conciseness over the base-cpx method and C4.5. In the domain of congressional voting records, all methods performed almost equally well in accuracy, the no-weight, c-weight, and d-weight methods generated simpler descriptions.

One result that is consistent in almost all domains is that the flexible classification method always improved the accuracy. In some domains, this improvement was significant.

CHAPTER 7

RELATED WORK

The research described in this thesis is related to previous work from three areas: flexible concept learning, simplification of concept descriptions, and the emerging problem of learning with hybrid representations. The work in learning flexible concepts is related to this work, because it shares the same goal as the research reported in this thesis: to explore methods to represent and learn flexible concepts. The work in simplifying concept descriptions is related, because the concept descriptions generated by the methods proposed in this thesis are often simpler than traditional methods. Finally, because the representations used in this thesis are hybrid the work of learning with hybrid representation is related. This chapter discusses the work in these three related areas.

7.1 Learning Flexible Concepts

The problem of learning flexible concepts is not new, and has been the subject of much research. This work includes fuzzy set approaches, probabilistic approaches, and exemplar-based approaches. These approaches are discussed in Michalski (1990) which is rephrased below.

Some of the most prominent work in the learning flexible concepts is the work on fuzzy sets by Zadeh and his collaborators and many followers (e.g., Zadeh, 1976; Mumdani and Gaines, 1981). This approach has concentrated primarily with representing the imprecision of concept boundaries, and has proposed to associate with an imprecise concept a set membership function

that defines the degree to which an instance represents the concept. This is usually a continuous numeric function, which expresses a subjective view of the concept variability. One way to interpret the set membership function is as a representation of the typicality of instances. It has been shown that such a function is useful for computationally representing the influence of linguistic modifiers, such as 'very', 'more' or 'less', and 'slightly' on the meaning of concepts. The fuzzy set approach has been widely studied and has found a number of applications, in particular, in the control of complex systems.

This approach does not address, however, a number of issues relevant to representing flexible concepts. The membership function must be defined by a person and for every context; the approach does not offer methods for automatically deriving such a function. The membership function is usually defined as one argument function; it is difficult to characterize in this way concepts whose boundaries depend on many arguments. The fuzzy set approach does not seem to provide adequate mechanisms for capturing concept extensions, representing multiple but interrelated meaning of a concept, reasoning about the context-dependence, or employing background knowledge for interpreting a concept. A set membership function is not sufficient for handling such problems.

In cognitive science literature, the inadequacy of representing human concepts by context-independent, logic-style definitions (the classical view), has been widely recognized (McCloskey and Glucksberg, 1978; Barsalou and Medin, 1986; and Lakoff, 1987). There have been other views advanced, such as the probabilistic view and the exemplar view (e.g., Smith and Medin, 1981).

The probabilistic view represents concepts by prototypes and uses the so-called family resemblance principle (e.g., Rosch and Mervis, 1975), while the exemplar view claims that concepts are represented by means of examples (e.g., Smith and Medin, 1981; Bareiss, Porter and Craig, 1990). Both views can be criticized on various grounds. The probabilistic view, which formally is based on the idea of linear separability, disregards the existence of correlations between the attributes, the context-dependency, and other information that has been shown relevant to human concept understanding (e.g., Estes, 1986; Flannagan, Fried and Holyoak, 1986).

The exemplar view promotes the idea of using similarity-based and context-sensitive matching, which has received support in the cognitive science literature. It ignores, however, the importance of general concept descriptions, that clearly play an important role in human concept formation. Such general descriptions are useful, for example, for comparing different concepts, for recognizing them from partial information, for identifying exceptions, handling context-

dependency, recording concept changes or for efficiently storing invariant information about concepts. The above operations are difficult to perform, if concepts are represented only by examples. In some work using the exemplar view, general aspects of concepts are captured under the idea of "category structure," which is a network of domain knowledge that specifies the relevance of exemplars to the concept they define (Bareiss, Porter and Craig, 1990). Some recent work has advocated a knowledge-based view, which emphasizes the need to define concepts through their role in theories in which they exist as interrelated components (Carey, 1985; Hofstadter, 1985; Schank, Collins and Hunter, 1986; Medin, 1989).

The two-tiered (TT) representation constitutes a significant departure from the existing approaches, although it has a relationship to most of them. The TT approach assumes that concepts have a certain central tendency, and proposes to describe this tendency explicitly, as the "first approximation" of the concept. On the other hand, it assumes that concepts' variability and context-dependency are best represented implicitly, by appropriate matching methods and context- and background knowledge-dependent rules of inference.

Thus, in the sense that it recognizes that concepts have a central tendency, and that there are typical and less typical concept examples, the TT approach is similar to the probabilistic view and the fuzzy set representation. It has also a relationship to the exemplar view, as it postulates the use of sophisticated matching procedures and inference rules in classifying new instances, and recognizes the usefulness of storing individual examples. The TT approach is also closely related to the knowledge-based view, as it stresses the role of background knowledge (and inference) in matching concept with instances, especially, non-typical or borderline instances.

7.2. Simplifying Concept Descriptions

The learning methods presented in this thesis relates to several efforts on simplifying concept descriptions, in particular, to the methods learning pruned decision trees (e.g. Quinlan, 1987; Cestnik et al, 1987; Fisher and Schlimmer, 1988). In decision tree learning methods, a concept description is a single tree structure that is supposed to account for all concept examples. Such decisions trees often suffer from excessive complexity, are not human understandable, and perform poorly in noisy domains. In a noisy or imprecise domain, classification is not exact. So at the lower levels of an exact tree, a decision tree construction algorithm tries to split a small subset of training examples randomly by choosing attributes that have no classification power. Decision tree pruning is the mechanism that aspires to prevent the construction of subtrees if they are

unreliable. Tree pruning replaces one or more subtrees that only cover few examples with leaves. Tree pruning involves finding a trade-off between the complexity of the decision tree and estimates of the classification error of the tree on training examples. While, a pruned decision tree is often much simpler than its unpruned version, it loses the coverage of training examples.

The HILLARY system developed by (Iba et al. 1988) uses a measure to determine the trade-off between the simplicity and accuracy of a description. This trade-off measure is somewhat similar to the GDQ measure used in POSEIDON system that was described in section 3.4. The GDQ measure, however, considers more factors. In addition to taking into account the typicality of the examples covered by the description, it considers the type of matching between an example and a description. Moreover, the simplicity measured by the GDQ depends not only on the number of disjuncts in the description as in (Iba et al. 1988), but also on the different syntactic features of the terms in the description.

The major difference between the work presented in this thesis and related research in simplifying descriptions is that TT methods can yield a simpler description without losing coverage. TT methods can compensate for the lack of coverage of one tier (BCR) by an application of the second tier (ICI), either by using flexible matching or deductive inference rules. In contrast, the simplified descriptions generated by other methods do not cover some of the training examples, they always produce some error on the training examples, and are only approximations of the target concepts if the uncovered training examples are not noise.

7.3 Hybrid Representations

No single representation is the best choice for all concept learning tasks. Different problems need different representations, or combinations of several representations. The best representation for a given problem depends on the characteristics of the concept to be learned and the data to learn from. It is often not known that which kind of representation is the most suitable for a given learning problem before learning starts. Thus, it would increase the autonomy and effectiveness of an inductive learning system if it were able to make its own choices regarding selection of representation. One way to do so is to use a hybrid representation and an associated learning algorithm that is able to adjust the hybrid representation to adapt the given learning problem. Two-tiered concept representation is a hybrid representation, because each concept is represented by two parts, and these two parts may use different representations. For example, in POSEIDON, this first part is a disjunctive normal form, and the second part includes a similarity matching function,

and a set of production rules.

Schlimmer (Schlimmer, 1987) has constructed a hybrid representation and associated learning algorithm embodied in his STAGGER program. STAGGER is designed to incrementally learn a concise and comprehensible description of a concept given a series of potentially noisy examples. The program maintains a pair of weights for each boolean term in his concept description. One corresponds to logical sufficiency, the other to logical necessity. By adjusting the weights and by adding or removing boolean terms, the program searches for a consistent concept description. Learning in STAGGER is accomplished by using three cooperative components. One of the learning components modifies the weight of boolean terms by counting examples (both positive and negative) that match or do not match each boolean term. The second component adds new boolean connectives to the concept description to repair overly general or overly specific hypotheses. The third component turns an aggregation of real-values into a few discrete ones.

The representation used in STAGGER can be viewed as a simple form of two-tiered representation. Boolean terms (both primitive and constructed) serve as the first tier, and the weights and the function used to compute Odds (degree of fit of an example) are the second tier.

Utgoff (Utgoff, 1988) described a hybrid representation called Perceptron Trees, and a perceptron tree learning algorithm called the Perceptron Tree Error Correction Procedure. A perceptron tree is a mixing of decision trees and linear threshold units. A perceptron tree is defined as either a linear threshold unit, or an attribute test with, for each value the attribute can take on , a branch to a perceptron tree. The perceptron tree error correction procedure incrementally updates the perceptron tree until it is consistent and complete. The initial perceptron tree consists of a single empty node. All leaves of the final tree are linear threshold units and all internal nodes are decision nodes.

CHAPTER 8

CONCLUSIONS

In this final chapter, I first summarize the research reported in this thesis, then discuss the contributions of this research, and finally limitations and future research are stated.

8.1 Summary

Flexible concepts play important roles in the real world life. This thesis is a study of mechanisms that automatically acquire flexible concepts from a set of given examples. The first chapter discussed the background and motivations of the research. The problem investigated in this thesis is learning flexible concepts from examples that is relatively less explored and emerges as an important direction in the field of machine learning.

Any concepts learned must be described in some representation language. Thus, an important issue is how to represent flexible concepts. In chapter 2, the idea of the two-tiered concept representation which is proposed to represent flexible concepts was introduced. In the TT representation, the concept meaning is defined by two components: the base concept representation (BCR), and the inferential concept interpretation (ICI). The BCR is an explicit description of basic concept properties, while the ICI characterizes allowed modifications of the concept meaning and its possible variations in different contexts. Thus, the ICI defines concept boundaries implicitly, by the results of matching procedures and inference processes. The approaches described in the rest of the thesis are based the idea of the two-tiered concept representation.

In the third chapter, a flexible concept learning approach and its implementation, POSEIDON system, was described. In contrast to existing learning methods in which concepts are represented as single knowledge structures, POSEIDON assumes a two-tiered concept representation. The BCR is a disjunctive normal form represented in the attribute-based *variable-valued logic system* VL_1 (Michalski, 1983). The ICI includes a flexible matching function, and a set of deductive rules. In POSEIDON, the BCR is obtained in a two-phase process. First, a complete and consistent description is learned from a conventional learning program (AQ15). Next, this description is optimized according to a general description quality measure (GDQ). This is done by a double-level search process that uses both generalization or specialization operators. The GDQ takes into account not only the properties of the BCR, but it also involves the ICI (e.g., complexity and accuracy of the total description).

The ICI has two components, one, which specifies a flexible matching function, and the second, which specifies a rule base for handling exceptions and context-dependency. The rules can be of two types. The rules of the first type extend the meaning of the concept, and rules of the second type contract the meaning defined by BCR. The first type of rules are employed when an instance does not satisfy the BCR, nor is covered by the flexible matching function. The rules of the second type are used when an instance is covered by a BCR of more than one concept, or when there is a need for confirming the concept membership. In both cases the rules are used deductively. An advantage of the rules over other methods of matching is that they provide an explanation why a given instance is or is not a member of the concept.

The experimental results reported in Chapter 4 have supported the hypothesis that TT concept descriptions can be simpler, more accurate and easier to understand than "single-tier" descriptions. For example, the obtained TT descriptions for the acceptable labor managements contracts gave the performance of over 90% correct and used only about 9 rules. In contrast, the best performance of a simple exemplar based method gave the 80% correct predictions on new examples and used 27 rules, and a pruned decision tree gave the performance of 86% and was equivalent to 29 rules. The method also outperformed the previous method based on the TRUNC procedure in terms of the performance (80%), but at the cost of a more complex concept description.

Chapter 5 described an alternative approach to learning flexible concepts. This approach has been implemented in the Flexible Concept Learning System (FCLS). The concept representation used in this approach is a simple form of the two-tiered concept representation, in this simple form, the ICI does not consists of inference rules. The ICI in FCLS includes a syntactic similarity measure that decides the degree of fit between an event and a disjunct (complex) of a concept

description. In addition to the similarity measure, each complex has a set of weights, and a threshold. Each weigh expresses the necessity of an individual selector of the complex, while threshold defines the boundary of the complex, and reflects the joint sufficiency of the selectors of the complexes. The weights are used in the similarity measure to determine the degree of fit. A complex plus its weights and threshold is called an extended complex. In addition to a set of extended complexes, a concept description may also include a set of exemplars which represent exceptions or rare cases. Two classification methods, the strict method and flexible method, were introduced to decide concept membership of an event based on the descriptions generated.

An extended complex is generated in two phases: base complex generation and extended complex optimization. The phase of base complex generation produces a set of base complexes that satisfy the consistency requirement and are the most general. A base complex is a complex without weights and threshold. The phase of extended complex optimization optimizes the base complexes generated in the phase of base complex generation to cover more positive examples by decreasing the thresholds of these complexes. In the optimization phase, the weights of a complex are computed, and the threshold is adjusted. Two weight learning methods, the count-based and degree-of-fit-based methods, were proposed to compute weights. In generating a complex, A pair of evaluation functions, Quality Evaluation Function (QEF) and Potential Quality Evaluation Function (PQEF), were applied to evaluate complexes. QEF decides the 'best' complex, while PQEF decides the complexes that are the most promising for improvement.

FCLS was tested on three different types of problems: the problems favorable for FCLS, the problems unfavorable for FCLS, and real world problems. In each experiment, FCLS was run with four different methods. One is the method without threshold adjusting and weight learning, the second is the method with threshold adjusting, but without weight learning, and the third and forth are the methods with both threshold adjusting and weight learning. To compare the method with other inductive learning method, the decision tree learning system C4.5 was also run on the same domains with and without pruning. The experimental results were reported in Chapter 6. As expected, the method with threshold adjusting and weight learning significantly improved both the accuracy and complexity of concept descriptions over the method without threshold adjusting and weight learning and C4.5 on the favorable problems. On the unfavorable problems, the method without threshold adjusting and weight learning and the method with threshold adjusting and without weight learning outperformed the method with weight learning and C4.5, and the results of the method with weight learning were comparable with C4.5. The results from the real world domains were less consistent, but generally speaking, the method described in Chapter 5 slightly improved both accuracy and complexity of descriptions over C4.5 and the method with threshold

adjusting and weight learning. The experimental results showed support for the proposed flexible concept learning method, and suggest some improvements which will be discussed in section 8.3.

Finally, in Chapter 7, I surveyed related work that addresses the tasks of learning flexible concepts from examples, simplifying concept descriptions , and hybrid representations.

8.2. Contributions

It has been stated that the primary goal of this research is to explore approaches to representing and learning flexible concepts. Accordingly, the major contribution of this work is that two representation formalisms were proposed to represent flexible concepts based on the idea of the two-tiered concept representation, and that their associated learning algorithms were designed to automatically acquire descriptions of flexible concepts. These two approaches were respectively implemented in: POSEIDON and FCLS, and were empirically evaluated on various domains. More specific contributions of these two systems are summarized below.

8.2.1 POSEIDON

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

Another important advance is the development of a heuristic double-level search procedure, called SG-TRUNC, which explores the space of two-tiered descriptions in order to determine a globally optimized description. The search employs both generalization and specialization operators, and is guided by a new criterion, the *general description quality* measure *(GDQ)*. This measure takes into consideration the description accuracy, its cognitive comprehensibility and the

computational cost of both tiers of a description (BCR and ICI). By introducing such a general description evaluation measure, the meaning of concept learning can be defined in a new way. Namely, concept learning can be viewed as a process of improving initial descriptions according to a given description quality criterion. The initial description can be in the form of positive and negative examples, a CC concept description, a teacher-supplied tentative description, or others.

Other advances include the use of a more powerful accuracy measurement in GDQ, which is able to take into consideration the typicality of training instances (when it is known), and the introduction of a rule base for implementing the ICI. The thesis presents also a body of experimental results on the performance of the proposed method, and its comparison to several other methods, such as various variants of exemplar-based learning, decision tree learning, and the earlier method based on the simple TRUNC procedure.

8.2.2 FCLS

FCLS uses a novel hybrid representation to describe a concept. First, the hybrid representation combines a symbolic representation with a numerical one. The symbolic representation is a disjunctive normal form, and the numerical representation includes a set of weights and thresholds. Second, an exemplar model is incorporated into the hybrid representation. This hybrid representation can be viewed as a combination of exemplar-based and logic based representations from two aspects. First, both generalized descriptions and exemplars are used together to describe a concept. Second, because of the use of the similarity measure, each base complex can be viewed as a generalized exemplar and treated in the same way as an exemplar. Under such a hybrid representation, proximate disjuncts may merge, concepts with irregular and imprecise boundaries can be represented more compactly, concept descriptions are graded and represent different degrees of typicality of examples, and exceptions are represented as exemplars.

Another accomplishment is the development of a technique for generating concept descriptions that are represented in the hybrid representation. This technique generates a concept description by adjusting the distributions between the symbolic and numeric representations, and between the generalized descriptions and exemplar-based descriptions to achieve the 'best' performance for a given problem. This algorithm consists of two phases. The first a symbolic description is generated, and the second phase optimizes the symbolic description by adjusting the distribution between the symbolic description (base complex) and the numeric one (weights and threshold). In the learning algorithm, the most specific description is generated for each concept to

Other important future work involves constructive induction (Michalski 1983). In general, constructive induction produces descriptions that are easier to understand, and capture the salient features of the concept. It would be useful to apply constructive induction to generate a BCR that captures the ideal principles of a concept. The effects of this type of induction are even more relevant for TT approaches. Constructive induction folds several disjuncts into a single one. Moreover, the use of constructive rules will usually merge relevant conjunctive descriptions of the concept, regardless of how many examples they cover. This feature of constructive induction may prevent removal of small relevant complexes, or replace them with exemplars.

Another open issue is the ability of the system to self-reorganize. This issue is also related to the problem of batch incremental learning. The distribution of knowledge between the BCR and the ICI may need to be adjusted when a set of new examples become available. So the incremental learning in a TT approach involves not only creating new BCR or ICI, but also adjusting the distribution between the BCR and ICI. This incremental learning feature is especially important when concept meaning drifts. Some exceptions-handling ICI rules are used very often to classify new examples, they could be compiled into explicit BCR assertions. Some exemplars may be replaced by generalized disjuncts, after more examples are seen. This interesting research problem merits further investigation.

In POSEIDON, the typicality of examples can be used in optimizing a complete and consistent concept description generated by AQ15. As mentioned earlier, the information of typicality could be very helpful in learning a TT concept description. The typicality of examples can be used to reduce the search space and find the optimum distribution of concept meaning between the BCR and ICI. In the future, FCLS will be extended to incorporate the algorithm to take advantage of the typicality of examples in learning TT concept descriptions. In this algorithm, typical examples will be used to learn the base complexes, less typical examples will be used later to relax the conditions included in the base complexes, that is to optimize the base complexes.

As we know the ICI define a concept implicitly. It is, therefore, very difficult for a user to understand a classification of an event when the ICI participates in the classification. It would be very useful if an explanation can be generated for such a classification. This explanation can serve as a qualitative result of a classification. The feature of generating explanation of a classification will be included in next version of FCLS.

Flannagan, J. J., Fried, L. S., and Holyoke, K. J., "Distributional Expectations and the Induction of Category Structure." In *Journal of Experimental Psychology: Learning, Memory and Cognition*, no. 12, 1986.

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

Hofstadter, D. R., "Analogies and Roles in Human and Machine Thinking." In *Metamagical Themas: Questing for the Essence of Mind and Pattern*, Basic Books, Inc., 1985.

Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975

Holte, R.C., Acker, L.E., and Porter, B.W., "Concept Learning and the Problem of Small Disjuncts." In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

Iba, W., Wogulis, J., and Langley, P., "Trading off Simplicity and Coverage in Incremental Concept Learning." In *Proceedings. of the Fifth International Conference on Machine Learning*, Ann Arbor, 1988.

Kedar-Cabelli, S. T. and McCarthy, L. T., "Explanation-based Generalization as Resolution Theorem Proving." In *Proceedings of the forth International Workshop on Machine Learning*, Irvine, 1987.

Kibler, D. and Aha, D., "Learning Representative Exemplars of Concepts." In *Proceedings of the Forth International Workshop on Machine Learning*, Irvine, 1987.

Kodratoff, Y and Michalski, R. S., *Machine Learning: An Artificial Intelligence Approach Volume III*, Morgan Kaufmann Publishers, 1990.

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

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

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

McCloskey, M. and Glucksberg, S., "Natural Categories: Well-defined or Fuzzy Sets?" In *Memory and Cognition*, no. 6, 1978.

Medin, D. L. and Smith, E. E., "Concepts and Concept Formation." In *Annual Review of Psychology*, M. R. Rosenzweig and L. W. Porter (Eds.), no. 35, 1984

Medin, D. L., "Concepts and Conceptual Structure." In *American Psychologist*, vol. 44, 1989.

Michalski, R. S., "AQVAL/1--Computer Implementation of a Variable-Valued Logic System VL_1 and Examples of its Application to Pattern Recognition." In *Proceedings of the First International Joint Conference on Pattern Recognition*, Washington, D.C., 1973.

Mooney, R. and, Ourston, D., "Induction Over the Unexplained: Integrated Learning of Concepts with Both Explainable and Conventional Aspects." In *Proceedings of Sixth International Workshop on Machine Learning*, Ithaca, NY, 1989.

Mumdani, E. H. and Gaines, B. R., *Fuzzy Reasoning and its Applications*, Academic Press, 1981.

Pagallo, G. and Haussler, D., "Feature Discovery in Empirical Learning." Technical Report UCSC-CRL-88-08, University of California at Santa Cruz, Santa Cruz, CA, 1988.

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

Quinlan, J. R., "Learning Efficient Classification Procedures and Their Application to Chess End Games." In *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds), Tioga, Palo Alto, Calif., 1983

Quinlan, J. R., "Simplifying decision trees." In *International Journal of Man-Machine Studies*, vol. 27, 1987.

Quinlan, J. R., "An Empirical Comparison of Genetic and Decision-Tree Classifiers." In *Proceedings. of the Fifth International Conference On Machine Learning*, Ann Arbor, 1988.

Quinlan, J. R., "Unknown Attribute Values in Induction." In *Proceeding of the Sixth International Workshop on Machine Learning*, 1989.

Rendell, L., "A New Basis for State-space Learning Systems and A Successful Implementation." In *Artificial Intelligence*, 20, 1983

Rendell, L., "A General Framework for induction and a study of selective induction." In *Machine Learning*, 1(2), 1986.

Rendell, L., "Learning Hard Concepts." In *Proceedings of the Third European Working Session on Learning*, 1988.

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

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

Schank, R. C., Collins, G. G, and Hunter, L. E., "Transcending Induction Category Formation in Learning." In *The Behavioral and Brain Sciences*, 1986.

Schlimmer, J. C., *Concept Acquisition Through Representational Adjustment*. PhD thesis, Department of Information and Computer Science, University of California, Irvine, 1987.

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

Spackman, K. A., "Learning Categorical Decision Criteria in Biomedical Domains." In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, June, 1988.

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

VITA

Jianping Zhang was born on March 18, 1956 in Wuhan, China. He grew up there and attended Wuhan University in Wuhan, People's Republic of China, where he received his Bachelor of Science degree in computer science in January 1982. After graduation he spent one and half years as a teaching assistant in the same university. In the summer of 1983 he began his graduate studies in Computer Science at the University of Illinois at Urbana-Champaign. He was a research assistant in the Department of Computer Science at the University of Illinois at Urbana-Champaign from May 1984 to January 1988, and in the Artificial Intelligence Center at George Mason University from January 1988 to August 1990. His Ph.D. was conferred in October 1990. He is currently an assistant professor at Utah State University.