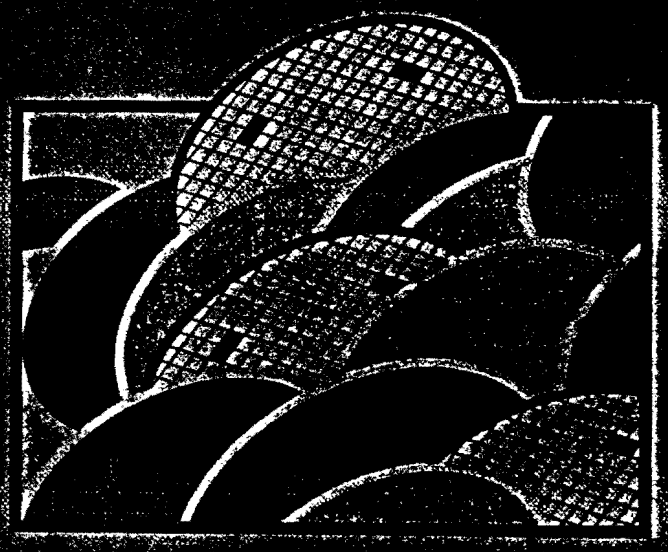

WORKSHOP NOTES

WORKSHOP PROGRAM

Consistency Driven Knowledge Elicitation within a Learning Oriented Representation of Knowledge

Gheorghe Tecuci* and Michael Hieb
Center for Artificial Intelligence, Department of Computer Science
George Mason University, Fairfax, VA 22030

Abstract

Knowledge representation in a semi-automated knowledge acquisition system should facilitate both knowledge elicitation and learning. Such a representation is implemented in the system NeoDISCIPLE. This is a concept based representation in which generalization and specialization operations are performed by simple syntactic transformations of knowledge. Thus, it is very appropriate for learning. In the same time, it also facilitates a type of knowledge elicitation that is associated with human oriented representations like, for instance, repertory grids. In this paper we discuss this learning oriented representation and present two methods for knowledge elicitation that are based on it. The methods are consistency driven in that they elicit additional knowledge from the expert in order to remove inconsistencies in the knowledge pieces learned by NeoDISCIPLE.

1 Introduction

One of the main problems in developing a method for knowledge acquisition concerns the representation of the knowledge to be acquired, and subsequently used, by an expert system. This is because what may be a suitable representation for a human, might not be adequate for the system, and vice versa. This problem becomes even more important in the context of a semi-automated knowledge acquisition method. In this case, one can no longer rely on a knowledge engineer to act as an interface between the expert and the system. Moreover, new constraints are imposed on the knowledge representation related to its suitability for learning.

Two different approaches to this representation problem are presented in the following.

One approach is to use two representations, one suitable for eliciting knowledge from the human expert and the other suitable for representing knowledge in an expert system, and to translate the knowledge from the first representation into the second one. An example of a representation that is particularly well suited for knowledge elicitation is the repertory grid which is based on research in experimental psychology [Kelly, 1955]. A repertory grid represents a person's system of cross-references between his personal observations (or experience of the world) and his personal constructs (or classifications of that experience) [Shaw and Gaines, 1987]. Experts easily understand this representation and are able to build grids without much effort. Moreover, the systems for repertory grid elicitation, like AQUINAS, ETS, PEGASUS [Boose and Bradshaw, 1988; Shaw and Gaines, 1987], guide the knowledge elicitation process by asking questions that are very easy to answer (e.g., Which feature makes two of the elements A, B, C similar and distinct from the third one? Which feature distinguishes two elements A and B? Is there an element that has the feature P without having the feature S?, etc.). However, although a repertory grid contains a significant amount of expert knowledge, this knowledge is not in a form that can be directly and efficiently used by an expert system. Therefore, much research effort has been involved in developing techniques for translating knowledge from a repertory grid representation to another representation as, for instance, inference rules, or concept hierarchies [Boose and Bradshaw, 1988; Gaines and Shaw, 1986; Shaw and Gaines, 1987]. A problem with this approach is that it is more difficult for the system to elicit additional knowledge for repairing its rules (or concept hierarchies) because knowledge is elicited in the form of repertory grids.

A very different approach to this representation problem is taken by the research in machine learning. In this case, the emphasis is on using a representation that helps the system build and develop its knowledge base (KB) through learning. Most of the current learning systems employ a knowledge representation based on extended propositional calculus or on first-order predicate calculus. This representation is suitable for learning because generalization and specialization operations, that are at the basis of most learning processes, can be performed as simple syntactic transformations of logical expressions [Michalski, 1983; Mitchell, 1978]. One problem with this approach is that the system needs a significant amount of prior knowledge for learning to occur, and this knowledge should be defined by the expert. Because the representation is not oriented toward the human expert, defining the prior knowledge is a difficult task. Moreover, this prior knowledge sets limits on the space of representation in which new knowledge is to be learned which may cause the so-called *problem of new terms* (i.e., the necessity of extending the representation language with new terms when it does not contain the concept to be learned). Without a human expert in the learning loop, this problem is very difficult, and has not yet received a satisfactory solution.

* Joint appointment with the Research Institute for Informatics, Bucharest, Romania.

Each of these approaches is strongly oriented toward one of the two contributors to a semi-automated knowledge acquisition process: the human expert and the learning system. A human oriented representation facilitates knowledge elicitation from the expert, while a learner oriented representation facilitates automatic learning.

In this paper we show that these two approaches are not antagonistic, and that it is possible to design a representation of knowledge that is suitable for both learning and knowledge elicitation. Designing such a representation was one of the goals of the system DISCIPLE [Tecuci, 1988; Tecuci and Kodratoff, 1990] and of its successor NeoDISCIPLE [Tecuci, 1992a,b]. On the one hand, the representation is based on the notion of 'concept', and supports the basic operations dealing with concepts: comparing the generality of concepts, generalizing concepts, and specializing concepts. Thus, it is very appropriate for learning. On the other hand, the representation is also suitable for knowledge elicitation, supporting a systematic elicitation of new concepts and features that is very similar in spirit with the techniques employed by repertory grid elicitation methods. Moreover, we will show that, by integrating methods for the elicitation of expert knowledge into a learning system, a natural solution to the problem of new terms is provided.

This paper is organized as follows. The next section contains a general presentation of the NeoDISCIPLE approach to the automation of knowledge acquisition. Section 3 describes the main features of the knowledge representation that makes it suitable for learning. Sections 4 and 5 present two methods for the consistency driven elicitation of expert knowledge, that are based on the knowledge representation presented. The last section points to the strengths and weaknesses of the knowledge elicitation methods presented, and of NeoDISCIPLE in general, in the context of the proposed representation.

2 General presentation of NeoDISCIPLE

NeoDISCIPLE is a learning and problem solving inference engine that supports a representation formalism in which a KB can be encoded, as well as a methodology for automatically building the KB.

The KB may contain object concepts and rules that are manipulated by a rule interpreter. The objects are described in terms of their features (properties or relationships), and are hierarchically organized according to the *more-general-than* (or 'IS-A') relationship, thus forming a hierarchical semantic network. The rules are expressed in terms of the object names and features. The meaning of the rules depends on the application domain. These rules may be inference rules for inferring new features of objects from other features [Tecuci, 1992a], general problem solving rules as, for instance, rules that indicate the decomposition of complex problems into simpler subproblems [Tecuci and Kodratoff, 1990], or even action models that describe the actions that could be performed by an agent (for instance, a robot), in terms of their preconditions, effects and involved objects [Tecuci, 1991].

To build an expert system with NeoDISCIPLE, the human expert has to first define a preliminary KB. The expert can define both objects and rules. In the current version of NeoDISCIPLE it is assumed that the object descriptions may be incomplete but correct, and the rules may be both incomplete and partially incorrect.

Then, the expert trains the system by providing new facts or specific examples of problem solving episodes. From this input, the system learns new rules, improves the existing ones, and extends the hierarchy of object concepts with new concepts and/or features. The learning method of NeoDISCIPLE integrates different learning strategies like explanation-based learning, learning by abduction, learning by experimentation, empirical generalization, and learning by instruction [Tecuci, 1992a].

The generalization language for learning is provided by the semantic network of object concepts. Because this semantic network is incomplete, the learned rules may be inconsistent, covering also some negative examples (called false positive or negative exceptions). These rules constitute the input to two knowledge elicitation methods which acquire additional knowledge from the expert in order to remove the inconsistencies from the KB by updating the semantic network and the rules.

The next section gives more details on the knowledge representation employed by NeoDISCIPLE.

3 Elements of knowledge representation

In order to facilitate learning, the objects and the rules are represented by using the following representation unit:

```
(concept-i concept-k (FEATURE-1 value-1)        (1)
                     ...
                     (FEATURE-n value-n))
```

The expression (1) defines 'concept-k' as being a specialization of 'concept-i' (from which it inherits features) and having additional features. The value of a feature may be a constant (in which case the feature is a property) or another concept (in which case the feature is a relationship). A concept represents a set of instances. Therefore, if 'concept-i' represents the set of instances Ci, then 'concept-k' represents a subset Ck of Ci. The elements of Ck have the features 'FEATURE-1', ... , 'FEATURE-n' with values that are instances of 'value-1', ... , 'value-n', respectively.

The representation unit (1) is equivalent to the following predicate calculus expression (where IS-A, FEATURE-1, ... , FEATURE-n are the names of the predicates):

```
(concept-k IS-A concept-i) &
(concept-k FEATURE-1 value-1) &
...
(concept-k FEATURE-n value-n)
```

Because of this straightforward equivalence, we previously used the predicate calculus notation in the papers describing NeoDISCIPLE, where the specific details of knowledge representation are not discussed.

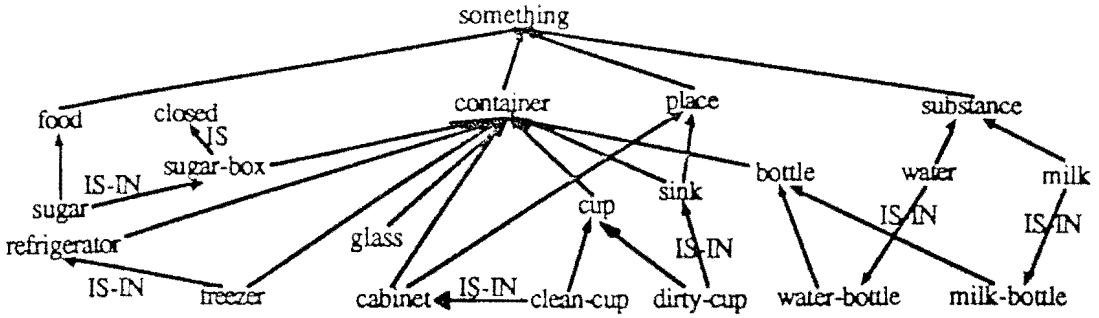

Figure 1: A semantic network of object concepts (the unlabeled arrows represent IS-A relationships).

Using (1), the concept "blue cup containing water" is represented as follows:

```
(cup  x   (COLOR      blue)
          (CONTAINS   water))
```

This concept could be generalized by simply generalizing 'cup', 'blue', or 'water', or by removing the property 'COLOR', or the property 'CONTAINS'. An example of such a generalization is:

```
(container  x  (CONTAINS y))
```

which represents the concept "container containing something". It was obtained by generalizing 'cup' to 'container', 'water' to any object, and by removing the constraint on the COLOR.

Figure 1 and Figure 2 present a sample KB of a planning system for a robot able to perform simple domestic tasks.

Figure 1 contains the semantic network of object concepts that describes the types of the objects in the world of the robot, together with their features. This semantic network is built from different descriptions of object-concepts of the form (1) as, for instance:

```
(container sugar-box (IS closed))          (2)
```

The semantic network organizes these descriptions hierarchically, along the IS-A relationship (for instance, 'cabinet' is both a 'container' and a 'place', and 'sugar' is a 'food'). In this hierarchy, any concept inherits all the features of its superconcepts. The semantic network may be incomplete in the sense that it may not contain all the relevant object names and features. For instance, (2) is obviously an incomplete description of a 'sugar-box'.

Figure 2 contains two rules for hierarchical planning. These rules have been learned by NeoDISCIPLE, from specific problem solving episodes indicated by the expert. For instance, the rule R1 has been learned from the following episode:

```
the problem                                (3)
    TAKE clean-cup
has the solution
    OPEN cabinet
    TAKE clean-cup FROM cabinet
```

The conditions of these rules are conjunctions of concept descriptions, in which each concept is represented by using the basic representation unit (1). One may notice

```
R1: IF
        (something x (IS-IN y)) & (container y)
    THEN
    the problem
        TAKE x
    has the solution
        OPEN y
        TAKE x FROM y
    with the positive examples:
        (x <- clean-cup, y <- cabinet)
        (x <- sugar, y <- sugar-box)
    with the negative exceptions:
        (x <- dirty-cup, y <- sink)
        (x <- freezer, y <- refrigerator)

R2: IF
        plausible upper bound
        (something x) & (something y (IS-IN z)) &
        (something z)
        plausible lower bound
        (container x) & (substance y (IS-IN z)) &
        (bottle z)
    THEN
    the problem
        FILL x WITH y
    has the solution
        POSITION-FOR-FILLING x
        TAKE z
        OPEN z
        POUR y FROM z INTO x
    with the positive examples:
        (x <- clean-cup, y <- milk, z <- milk-bottle)
        (x <- glass, y <- water, z <- water-bottle)
```

Figure 2: Problem solving rules.

that, while the rule R1 has a single applicability condition, the rule R2 has two conditions, called the plausible upper bound and the plausible lower bound. The plausible upper bound is a conjunctive expression that is supposed to be more general than the exact condition, and the plausible lower bound is a conjunctive expression that is supposed to be less general than the exact condition. The two bounds define a *plausible version space* [Tecuci, 1992a] for the exact condition to be learned by NeoDISCIPLE. The bounds and the version space are

called plausible because the learning process takes place in an incomplete representation language that may cause them to be inconsistent (a lower bound that covers negative examples or an upper bound that does not cover positive examples).

There are several differences between the plausible version spaces (PVS) defined in NeoDISCIPLE and the certain version spaces (CVS) defined by [Mitchell, 1978].

First, in the case of CVS, the lower bound and the upper bound are exact boundaries. Consequently, during learning, the lower bound is only generalized and the upper bound is only specialized. On the contrary, in the case of PVS, these bounds are approximations of the exact bounds. Therefore, during learning, each of them can be both generalized and specialized.

Second, CVS needs to be exhaustively searched. Consequently, each bound may consist of a large number of conjunctive expressions that may cause a combinatorial explosion. On the contrary, PVS is heuristically searched and each bound consists of one conjunctive expression.

The main difference, however, between CVS and PVS concerns the type of the rules to be learned. With CVS one can only learn consistent descriptions. Therefore, if the representation language does not contain a description that covers all the positive examples and none of the negative examples, then no rule is learned. On the contrary, by employing a plausible version space, NeoDISCIPLE is able to learn even in such situations, but the learned rules will be inconsistent, (i.e., will cover also some negative examples). Consequently, the rules in NeoDISCIPLE's KB could be inconsistent.

Each example I from which a rule R was learned is kept in the description of the rule as the substitution σ defined by 'σR=I'. For instance, the substitution 'σ=(x<-clean-cup, y<-cabinet)' corresponds to example (3) of the rule R1. Indeed, by applying σ to the 'THEN' part of R1, one obtains example (3).

All the positive examples E and the negative examples N from which a rule R was learned are partitioned into four groups: 1) *the positive examples* are the elements of E that are covered by R; 2) *the negative examples* are the elements of N that are not covered by R; 3) *the positive exceptions* are the elements of E that are not covered by R; 4) *the negative exceptions* are the elements of N that are covered by R. These examples and exceptions of the rules are the main source of knowledge for extending the representation language of the system with new object concepts or concept features, as well as for updating the rules, as will be shown in the following sections.

4. Consistency-driven knowledge elicitation

The generalization language for the rules learned by NeoDISCIPLE is provided by the semantic network of object concepts. Indeed, the conditions of the rules are conjunctions of basic representation units in which concept-i (see (1)) is an object concept from the semantic network in Figure 1, and FEATURE-1, ..., FEATURE-n, are features from the same semantic network.

Because this semantic network is incomplete, the corresponding representation space may not contain a consistent rule covering all the positive examples and none of the negative examples. In such a situation, many of the current learning systems will simply not learn any rule at all. The learning method of NeoDISCIPLE, however, is based on the assumption that the representation language may be incomplete. Consequently, while its goal is to learn a consistent rule, the system will also attempt to learn a rule that has as few inconsistencies as possible (i.e., as few covered negative examples as possible).

On the other hand, the existence of the negative exceptions in the learned rules is an indication that the hierarchical semantic network of concepts is incomplete, and may guide the elicitation of new object concepts or features from the expert. This allows one to formulate the knowledge elicitation problem in Table 1.

Table 1: The problem of consistency-driven KE

Given
- an incomplete KB;
- an inconsistent rule;
- a set of positive examples covered by the rule;
- a set of negative exceptions.

Determine
- an improved KB and an improved rule description that has fewer exceptions. The KB is improved by eliciting new object concepts or features from the human expert.

To illustrate this knowledge elicitation problem, let us consider the semantic network in Figure 1, together with the rule R1 from Figure 2 that has two positive examples and two negative exceptions. For instance, the negative exception represented by the substitution σ = (x <- freezer, y <- refrigerator) is the following incorrect problem solving episode:

the problem
TAKE freezer
has the solution
OPEN refrigerator
TAKE freezer FROM refrigerator

The negative exceptions could be turned into negative examples by introducing the new concept 'movable-object' and the concept property '(cabinet IS closed)' into the semantic network, as shown in Figure 3. With these extensions of the generalization language, the rule R1 is updated as shown in Figure 4.

For solving the knowledge elicitation problem in Table 1, we have developed, and integrated into NeoDISCIPLE, the methods that will be presented in the following sections. A distinctive feature of these methods is that they perform a type of knowledge elicitation that is very similar to

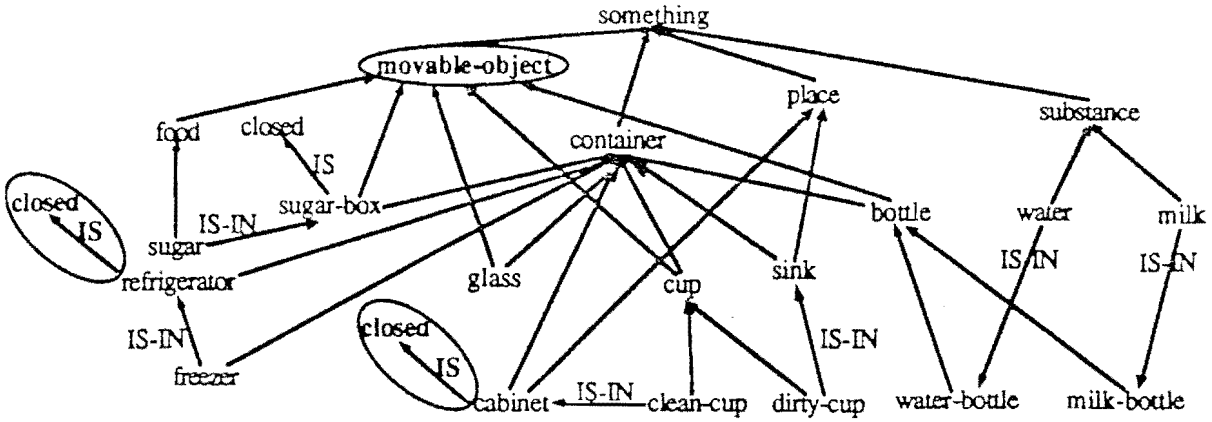

Figure 3: An improved semantic network of object concepts (the unlabeled arrows are IS-A relationships).

R1: IF
(movable-object x (IS-IN y)) &
(container y (IS closed))
THEN
the problem
TAKE x
has the solution
OPEN y
TAKE x FROM y
with the positive examples:
(x <- clean-cup, y <- cabinet)
(x <- sugar, y <- sugar-box)
with the negative examples:
(x <- dirty-cup, y <- sink)
(x <- freezer, y <- refrigerator)

Figure 4: Updated rule.

repertory grid elicitation, but operate on a representation of knowledge that is particularly suitable for learning.

5 Consistency-driven feature elicitation

A negative exception of a rule may be transformed into a negative example by identifying (or defining) a new object feature that discriminates between the positive examples and the negative exception. An outline of the method is presented in Table 2. A detailed description is given in (Hieb and Tecuci, 1992).

Let us consider, for instance, the rule R1 in Figure 2 that has the positive examples represented by

(x <- clean-cup, y <- cabinet)
(x <- sugar, y <- sugar-box)

and the negative exceptions represented by

(x <- dirty-cup, y <- sink)
(x <- freezer, y <- refrigerator)

To eliminate these negative exceptions, the system will look for a property of 'clean-cup' that may be property of 'sugar', without being a property of 'dirty-cup' or a property of 'freezer'. Or, it may look for a property of 'sugar' that may be property of 'clean-cup', without being a property of 'dirty-cup' or a property of 'freezer'. Or, it may look for a property of 'cabinet' that may be property of 'sugar-box', without being a property of 'sink' or a property of 'refrigerator'. Or, it may look for a property of 'sugar-box' that may be property of 'cabinet', without

Table 2: The method for consistency-driven feature elicitation

Let R be an inconsistent rule, E_1,..., E_k the set of its positive examples, and N_n,...,N_s the set of its negative exceptions. Each positive example E_i and each negative exception N_i is represented as a substitution $\sigma = (v_{i1} <- O_{i1}, \ldots, v_{im} <- O_{im})$ that transforms rule R into the corresponding example, where v_{ij} is a variable from R and O_{ij} is an object from the object hierarchy.

1. Determine or elicit a feature F of an object O_{ij} from one positive example E_i such that:
   - F is a feature of the corresponding objects O_{1j},...,O_{kj}, from ALL the positive examples;
   - F is not a feature of the corresponding objects O_{pj},...,O_{rj} from SOME of the negative exceptions N_p,...,N_r.
2. Refine the descriptions of the objects O_{1j},...,O_{kj} by adding the feature F.
3. Refine the descriptions of O_{pj},...,O_{rj} by adding the feature NOT-F, if such a feature makes sense for these objects.
4. Particularize the condition of the rule by adding the feature F (this feature is now shared by all the current positive examples).
5. Remove N_p,...,N_r from the list of negative exceptions and add them to the list of negative examples.
6. Repeat this procedure as long as such features can be found or elicited from the user or until there are no more negative exceptions.

being a property of 'sink' or a property of 'refrigerator'.

An illustration of the knowledge elicitation dialog is the following one (the answer 'Irrelevant' means that the property proposed by the system does not make sense for the corresponding object concept):

I know that '(sugar-box IS closed)'.
Is it true that '(cabinet IS closed)' ? *Yes*

This means that '(y IS closed)' is a property characterizing all the positive examples and may distinguish them from some of the negative exceptions.

Is it true that 'NOT (sink IS closed)' ? *Irrelevant*
Is it true that 'NOT (refrigerator IS closed)' ? *No*

As a result of this dialog, the system has found the property '(y IS closed)' which discriminates between the positive examples and the negative exception

(x <- dirty-cup, y <- sink).

By introducing the discriminating property into the rule's condition, this negative exception becomes a negative example. However, the second negative exception is still covered by the rule. This negative exception will be transformed into a negative example by using another method that is presented in the next section.

Another result of the above dialog is that the descriptions of 'cabinet' and 'refrigerator' are refined by adding the discovered property, as shown in Figure 3. This is a case of goal-driven property transfer from one concept ('sugar-box') to other concepts ('cabinet' and 'refrigerator').

It may also happen that the system cannot find a property to transfer. In such a case, it will try to elicit a new property by using a technique similar to the triad method employed in the elicitation of the repertory grids [Boose and Bradshaw, 1988; Shaw and Gaines, 1987]. If the semantic network in Figure 1 does not contain the property '(sugar-box IS closed)', the knowledge elicitation dialog would be the following one:

Consider the following two groups of objects
group1: sugar-box, cabinet
group2: sink, refrigerator
in the context of the following problem solving episode in which y may take values from group1
the problem
TAKE x
has the solution
OPEN y
TAKE x FROM y
Could you think of some property that discriminates all the objects from group1 from at least one object from group2 [Yes/No]: *Yes*
Describe this property for each object
(sugar-box IS closed)
(cabinet IS closed)
(refrigerator IS closed)
Irrelevant (sink IS closed)

A similar method exists for eliciting relationships between object concepts. Indeed, let us again consider the positive examples and the negative exceptions of rule R1 in Figure 2. To eliminate one or both of the negative exceptions the system may look for a relationship between 'sugar' and 'sugar-box' that may also hold between 'clean-cup' and 'cabinet', but cannot hold between the corresponding objects from at least one negative exception (for example, cannot hold between 'dirty-cup' and 'sink', or between 'freezer' and 'refrigerator').

6 Consistency-driven concept elicitation

Another method of transforming a negative exception of a rule into a negative example consists of eliciting a new concept that discriminates between the positive examples and the negative exception. An outline of the method is presented in Table 3. A detailed description is given in (Hieb and Tecuci, 1992).

Table 3: The method for consistency-driven concept elicitation

Let R be an inconsistent rule, E_1,..., E_k the set of its positive examples, and N_n,...,N_s the set of its negative exceptions. Each positive example E_i and each negative exception N_i is represented as a substitution $\sigma = (v_{i1} <- O_{i1}, \ldots, v_{im} <- O_{im})$ that transforms rule R into the corresponding example, where v_{ij} is a variable from R and O_{ij} is an object from the object hierarchy. Let C_j be the object-concept in the condition of R that covers the objects $O_{1j}, \ldots, O_{kj}, O_{nj}, \ldots, O_{sj}$, one from each of the positive examples and the negative exceptions.

1. Elicit a new object concept C'_j that
   - covers corresponding objects $O_{1j}, \ldots, O_{kj}$ from ALL the positive examples;
   - does not cover the corresponding objects O_{pj},...,O_{rj} from SOME of the negative exceptions N_p,...,N_r;
   - is less general than C_j;
   - may be given a meaningful name by the user.
2. Find the place of the concept C'_j in the hierarchical semantic network and introduce it there.
3. Modify the condition of R by replacing C_j with C'_j.
4. Remove N_p,...,N_r from the list of the negative exceptions and add them to the list of the negative examples.
5. Repeat this procedure as long as there are negative exceptions and meaningful concepts can be defined.

Let us consider, for instance, the positive examples

(x <- clean-cup, y <- cabinet)
(x <- sugar, y <- sugar-box)

and the negative exception

(x <- freezer, y <- refrigerator)

The concept covering 'clean-cup', 'sugar', and 'freezer', in the condition of the rule R1 is 'something'. Also, the concept covering 'cabinet', 'sugar-box', and 'refrigerator' is 'container'.

In order to remove the negative exception, one may define a concept covering 'clean-cup' and 'sugar', but not covering 'freezer', as indicated in Figure 5.

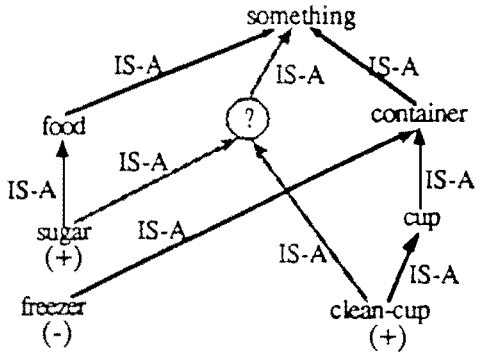

Figure 5: An object concept that would eliminate a negative exception.

Alternatively, one may define a concept covering 'cabinet', and 'sugar-box', not covering 'refrigerator', and being less general than 'container', as indicated in Figure 6.

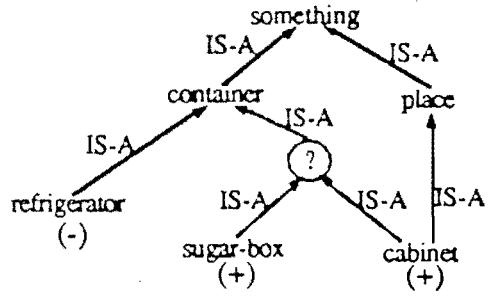

Figure 6: Another object concept that would eliminate a negative exception.

The knowledge elicitation dialog is the following one:

Consider the following problem solving episode

the problem
TAKE x
has the solution
OPEN y
TAKE x FROM y

Could you think of some concept corresponding to x that cover the objects 'sugar', 'clean-cup' without covering the object 'freezer' [Yes / No]: *Yes*

Give this concept a meaningful name: *movable-object*

Is any cup a movable-object [Yes / No]: *Yes*

Is any food a movable-object [Yes / No]: *Yes*

Is any place a movable-object [Yes / No]: *No*

. . .

The last questions are asked by the system to find the location of the new concept ('movable-object') in the hierarchical semantic network of concepts. This place is somewhere in between the covered objects ('sugar', 'clean-cup') and the corresponding concept ('something') from the condition of the rule. To this purpose, the system tries to elicit the generalization relationships that exists between the newly defined concept, on the one hand and, on the other hand:

- the ancestors of 'sugar' and 'clean-cup' that are less general than 'something' but do not cover 'freezer';
- the descendants of 'something' that do not cover the known positive and the negative examples of movable objects ('sugar', 'clean-cup' and 'freezer').

As a result of the above knowledge elicitation dialog, the new concept 'movable-object' is introduced in the semantic network, as indicated in Figure 3.

One may notice that this method is a type of goal-driven conceptual clustering in which known concepts are clustered under newly defined ones, to improve the consistency of the learned rules.

7 Related research and conclusions

In this paper we have addressed the problem of representing knowledge for supporting both learning and knowledge elicitation and we have briefly presented the knowledge representation employed by NeoDISCIPLE. The adequacy for performing learning with this representation was discussed in other papers, particularly in [Tecuci, 1988; Tecuci and Kodratoff, 1990; Tecuci, 1992a]. Therefore, in this paper, we have concentrated on the adequacy of this representation for knowledge elicitation, by presenting a method for eliciting concept features and another one for eliciting new concepts. In the following we present some of the strengths and weaknesses of these methods.

The methods are similar in spirit to those developed for the elicitation of repertory grids [Boose and Bradshaw, 1988; Shaw and Gaines, 1987] in that the human expert is asked the same kind of questions. However, these questions do not concern the constructs and elements of a repertory grid, but the concepts and features from a hierarchical semantic network of concepts. Moreover, the repertory grid elicitation is driven by a general goal of eliciting differentiations between the elements and the constructs of the expertise domain when their descriptions are not distinct enough. The methods presented in this paper are driven by the specific goal of eliciting those differentiations between the concepts of the domain that eliminate specific inconsistencies, as represented by inconsistent rules.

BLIP [Wrobel, 1989] employs an automated method for defining new object concepts that is similar to the one from Table 3. The main difference is that BLIP always defines only one concept, that discriminates between the objects from the positive examples and the corresponding

objects from the exceptions of a rule. Our method may define several concepts, each eliminating at least one exception, if they are meaningful concepts in the modeled domain. If the goal is to represent the real world as closely as possible, there is no reason to believe that one always has to define only one concept for eliminating all exceptions. Giving meaningful names to the concepts is also an important aspect that is not addressed by BLIP.

The knowledge elicitation method from Table 3 is also related to the method employed by Nanoklaus [Haas and Hendrix, 1983] in that both methods ask questions in order to find the correct place of a concept in a concept hierarchy. However, Nanoklaus simply receives a concept from the user that is not related to any particular problem solving method (besides further retrieval of the elicited knowledge). On the contrary, NeoDISCIPLE guides the expert in defining new concepts in the context of specific inconsistent problem solving rules.

There are also several weaknesses and limitations of the knowledge elicitation methods presented which indicate future research directions.

First of all, these methods elicit object knowledge that only extends the hierarchical semantic network, which is supposed to be incomplete but correct. If the semantic network may also be partially incorrect, then the presented methods need to be developed to also perform deletions and modifications of object knowledge.

The methods are also limited in that they deal only with the negative exceptions. However, a rule may also have positive exceptions. A positive exception is a positive example of a rule that cannot be covered by the rule without generalizing it to cover additional negative examples. One way to deal with the positive exceptions is simply to learn another rule that covers them. However, it may be possible to modify the semantic network to allow the rule to cover some of the positive exceptions, in the same way as with the negative exceptions.

The knowledge elicitation methods that were presented elicit knowledge in order to remove the inconsistency of a single rule. An important extension would be to analyze several inconsistent rules at the same time, and to base knowledge elicitation on a more global view of the KB's inconsistency.

Other techniques used by NeoDISCIPLE to elicit expert knowledge are described in [Tecuci, 1992b]. However, all of these techniques elicit knowledge in order to *extend* an initial KB. By this, they simplify the definition of the initial KB, which could be incomplete. Developing methods for *creating* this initial KB is an important research direction of the knowledge acquisition methodology promoted by NeoDISCIPLE.

Acknowledgments

This research was done in the GMU Center for Artificial Intelligence. Research of the Center is supported in part by NSF Grant No. IRI-9020266, in part by ONR grant No. N00014-91-J-1351, and in part by DARPA grant No. N00014-91-J-1854, administered by ONR.

References

Boose, J.H. and Bradshaw, J.M., Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge-acquisition Workbench for Knowledge-based Systems, in J.Boose and B.Gaines (eds), *Knowledge Acquisition Tools for Expert Systems*, Academic Press, 1988.

Gaines, B.R. and Shaw, M.L.G., Induction of inference rules for expert systems, *Fuzzy Sets and Systems*, 18, pp. 315-328, 1986.

Hass, N., Hendrix, G., Learning by Being Told: Acquiring Knowledge for Information Management, in R.S.Michalski, J.Carbonell, T.M.Mitchell (eds), Machine Learning vol.1, Tioga Pub. Co., 1983.

Hieb, M., and Tecuci, G., Two Methods for Consistency-driven Knowledge Elicitation, *Reports of the Machine Learning and Inference Laboratory*, Artificial Intelligence Center, George Mason University, 1992.

Kelly, G.A., *The Psychology of Personal Constructs*, New York, Norton, 1955.

Michalski, R.S., A Theory and Methodology of Inductive Learning, *Artificial Intelligence*, 20, pp.111-161, 1983.

Mitchell, T.M., Version Spaces: An Approach to Concept Learning, *PhD Thesis*, Stanford University, 1978.

Shaw, M.L.G. and Gaines, B.R., An Interactive Knowledge Elicitation Technique Using Personal Construct Technology, in A.L. Kidd (ed), *Knowledge Acquisition for Expert Systems: A Practical Handbook*, Plenum Press, 1987.

Tecuci, G., DISCIPLE: A Theory, Methodology, and System for Learning Expert Knowledge, *Ph.D. Thesis*, University of Paris-Sud, 1988.

Tecuci, G. and Kodratoff Y., Apprenticeship Learning in Imperfect Theory Domains, in Kodratoff Y., and Michalski R.S. (eds), *Machine Learning: An Artificial Intelligence Approach*, vol. III, Morgan Kaufmann, 1990.

Tecuci, G., A Multistrategy Learning Approach to Domain Modeling and Knowledge Acquisition, in *Proc. of the European Working Session on Learning*, Porto, Springer-Verlag, 1991.

Tecuci, G., Automating Knowledge Acquisition as Extending, Updating and Improving a Knowledge Base, *IEEE Transactions on Systems, Man, and Cybernetics*, 22, 4, July 1992a.

Tecuci, G., Cooperation in Knowledge Base Refinement, in D. Sleeman (ed.), *Proceedings of the Machine Learning Conference*, Aberdeen, Morgan Kaufmann, 1992b.

Wrobel, S., Demand-Driven Concept Formation, in Morik K.(ed), *Knowledge Representation and Organization in Machine Learning*, Springer Verlag 1989.