P91-23

91-23

# AUTOMATING KNOWLEDGE ACQUISITION AS EXTENDING, UPDATING, AND IMPROVING A KNOWLEDGE BASE

Gheorghe Tecuci

MLI 91-7

# AUTOMATING KNOWLEDGE ACQUISITION AS EXTENDING, UPDATING, AND IMPROVING A KNOWLEDGE BASE

Gheorghe Tecuci*
Artificial Intelligence Center, Department of Computer Science
George Mason University, 4400 University Drive, Fairfax, VA 22030-4444
email: tecuci@aic.gmu.edu

## ABSTRACT

The paper presents an approach to the automation of knowledge acquisition for expert systems. The approach is based on several general principles emerging from the field of machine learning: expert system building as a three phase process, understanding-based knowledge extension, knowledge acquisition through multistrategy learning, consistency-driven concept formation and refinement, closed-loop learning, and synergistic cooperation between the human expert and the learning system. In this approach, an expert system is built by a human expert and a learning system. The human expert defines the framework for the expert system and provides an incomplete and partially incorrect knowledge base. The learning system incrementally extends, updates, and improves the knowledge base through learning from the human expert. This approach is illustrated by the learning system shell NeoDISCIPLE

* Joint appointment with the Research Institute for Informatics, 71316, Bd.Mrs.Averescu 8-10, Bucharest 1, Romania

# I. INTRODUCTION

Automating the process of building expert systems is one of the major goals of artificial intelligence. An expert system has two basic components, a knowledge base (which contains knowledge relevant to a particular domain of expertise) and an inference engine (which provides the control and inference mechanisms for applying the knowledge from the knowledge base). This characteristic architectural feature of the expert systems determined two main approaches to the automation of the expert system building process: building expert system shells and building tools for knowledge acquisition.

An expert system shell is a system that consists of an inference engine for a class of tasks, and supports representation formalisms in which a knowledge base can be encoded. If the inference engine of an expert system shell is adequate for a certain expert task, then the process of building the expert system is reduced to the building of the knowledge base. The expert system shells could be characterized by the generality of their inference engine. The range of such systems contains very general shells, like OPS (Cooper and Wogrin, 1988) and KEE (IntelliCorp, 1988), general shells for a certain type of expertise task like, for instance, diagnosis in the case of EMYCIN (van Melle et al., 1981), and even quite specific shells for role-limiting problem solving methods as, for instance KNACK (Klinker, 1988) and SALT (Marcus, 1988). The different types of expert system shells trade the generality of the inference engine (and thus their domain of applicability) against the assistance given to the building of the knowledge base. Very general shells give little assistance besides the encoding of knowledge in rules or objects. On the contrary, the shells implementing role-limiting methods provide considerable assistance in building a knowledge base (Marcus, 1988). A role-limiting method is characterized by a very simple control structure that is independent of the peculiarities of any particular task performed. Also, it defines clearly the roles that the required task knowledge plays and the form in which that knowledge can be represented (McDermott, 1988). Research on the identification of problem solving methods for generic tasks (Chandrasekaran, 1983, 1986; Clancey, 1984) aims at defining suitable shells for building expert systems.

A tool for knowledge acquisition provides assistance in building a knowledge base. In general, one may distinguish three stages of the knowledge acquisition process: systematic elicitation of expert knowledge, knowledge base refinement, and knowledge base reformulation. During systematic elicitation, the basic terminology and the conceptual structure of the knowledge base is acquired. Most often this is done through a structured interview with a human expert (Boose and Bradshaw, 1988; Gammack, 1987; Shaw and Gaines, 1987). The result of the systematic elicitation is an initial imperfect knowledge base that is refined and improved during the next stages. During knowledge refinement, the knowledge base is debugged and extended. Knowledge-refinement tools use the problem-solving abilities of the expert system to identify failures (i.e. inability to solve some problem or generation of a wrong solution). When problem-solving fails, the tool elicits knowledge from the human expert in order to eliminate the cause of the failure (Bareiss, Porter and Murray, 1989; Tecuci, 1988; Wilkins, 1990). During reformulation, the knowledge base is reorganized and/or compiled to solve problems more efficiently (Mitchell, Mahadevan and Steinberg, 1985; Minton, 1989).

The above classification of expert system building tools into expert system shells and knowledge acquisition tools reflects also the traditional distinction made between problem solving and learning. However, as new and more powerful learning methods are developed, this distinction appears to be more and more artificial. Indeed, for learning methods like explanation-based learning, abductive learning, or learning by analogy (Kodratoff and Michalski, 1990; Birnbaum and Collins, 1991) problem solving is often part of learning. Based on this observation, we define a *learning system shell* as a learning and problem solving inference engine that supports representation formalisms in which a knowledge base can be encoded, as well as a methodology for automatically building the knowledge base. Thus, a learning system shell is an expert system building tool that incorporates both the capabilities of an expert system shell and those of a knowledge acquisition tool.

In this paper we present the learning system shell NeoDISCIPLE that is based on several general principles emerging from the field of machine learning: expert system building as a three phase process, understanding-based knowledge extension, knowledge acquisition through multistrategy learning, consistency-driven concept formation and refinement, closed-loop learning, and synergistic cooperation between the human expert and the learning system shell. By identifying and implementing these principles we intended to provide a general framework for building more specialized learning system shells, in which these principles would enhance domain-specific methods, as advocated by (Marcus, 1988).

It should be noticed that, although NeoDISCIPLE aims at automating the entire process of building an expert system, this paper concentrates on the support provided by NeoDISCIPLE for extending, updating and improving a knowledge base.

This paper is organized as follows. Section II briefly presents the principles on which NeoDISCIPLE is based. Then, sections III through VII illustrate its use with the help of an example of building a question-answering system in the area of geography. Section VIII presents conclusions from several experiments with NeoDISCIPLE. Section IX relates NeoDISCIPLE to other approaches and the last section outlines the most promising directions of the future research.

## II. PRINCIPLES FOR AUTOMATING THE KNOWLEDGE ACQUISITION PROCESS

### *A. Expert system building as a three phase process*

NeoDISCIPLE is a learning system shell that provides a general framework for building an expert system. In this framework, knowledge has to be represented as objects and rules that are manipulated by a rule interpreter. The objects are described in terms of their properties and relationships, and are hierarchically organized according to the "more-general-than" (or "isa") relationship, thus forming a hierarchical semantic network. The rules are expressed in terms of the object names, properties and relationships. The meaning of the rules depends of the application domain. They may be inference rules for inferring new properties and relationships of objects from other properties and relationships, 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, 1991a).

With NeoDISCIPLE, an expert system is built in three phases.

In the first phase the human expert has to define, within the general framework provided by NeoDISCIPLE, a preliminary knowledge base (KB). There are two main goals of this phase: a) to allow the human expert to introduce into the KB whatever knowledge pieces s/he may easily express; b) to provide the system with some background knowledge that would support it in learning new knowledge. The result of this phase will be an incomplete and partially incorrect KB that will mainly consist of object descriptions.

In the second phase, the system extends, updates, and improves the KB through learning from new input information provided by the human expert. That is, it extends the hierarchy of object concepts with new properties, relationships, and concepts, learns new rules, and improves the existing ones. The result of this phase will be a KB that is complete enough and correct enough for providing correct solutions to the solved problems.

In the last phase, the knowledge base is reorganized for improving the efficiency in problem solving. The result of this phase should be an efficient expert system.

## *B. Understanding-based knowledge extension*

The imperfect KB provided by the human expert allows the learning system to react to new input information with the goal of extending, updating, and improving the KB so as to consistently integrate the input.

In general, the input may be any piece of knowledge. However, in the current version of NeoDISCIPLE, the input is supposed to represent a specific fact, an example of a concept, or an example of a problem solving episode (consisting of a specific problem and its solution). Usually, the result of learning from a specific fact will be an improved KB implying the input fact and similar ones. Also, the result of learning from a specific problem solving episode will be an improved KB allowing the system to solve similar problems.

The general learning method of NeoDISCIPLE is shown in Figure 1, and is based on the "understanding" of the input* (Tecuci and Michalski, 1991b). That is, the system will try to show that the input is a plausible consequence of the system's knowledge. To build such a plausible proof, it may need to hypothesize new knowledge, which is added into the KB. Moreover, in order to learn as much as possible from the input, the system will generalize the plausible proof (and thus the hypothesized knowledge), will analyze the instances of these generalizations, and will correct them accordingly. In the case the input represents entirely new knowledge that cannot be related to the current KB (i.e, cannot be understood by the system), it is added as such into the KB. As shown in Figure 1, all this process is supervised by the human expert.

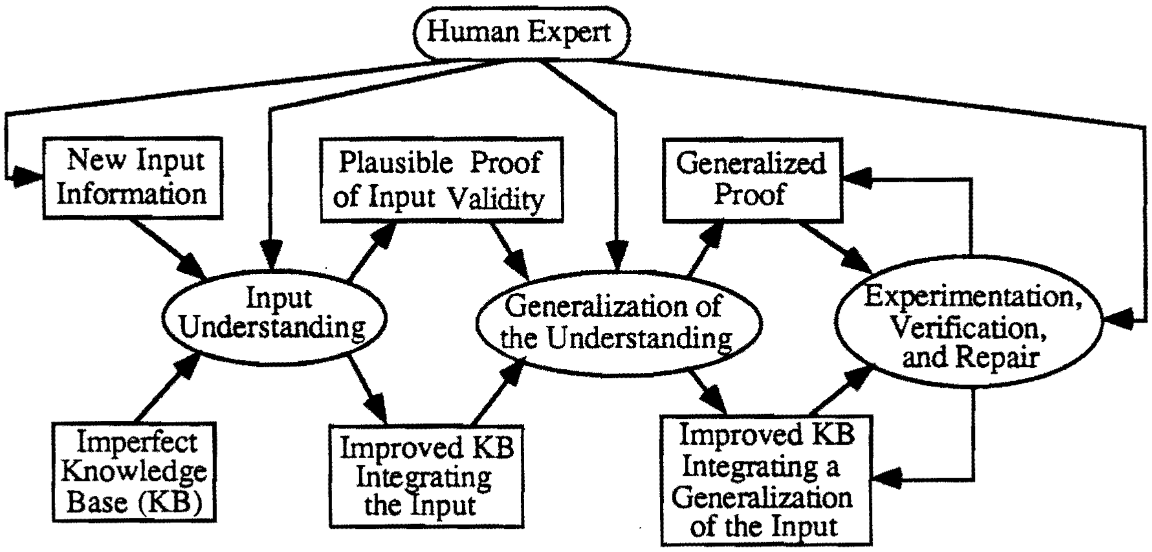

Human Expert
New Input Information
Plausible Proof of Input Validity
Generalized Proof
Input Understanding
Generalization of the Understanding
Experimentation, Verification, and Repair
Imperfect Knowledge Base (KB)
Improved KB Integrating the Input
Improved KB Integrating a Generalization of the Input

Figure 1: Incremental development of the knowledge base

## *C. Knowledge acquisition through multistrategy learning*

One of the reasons that research in machine learning (Shavlik and Dietterich, 1990) has not yet had a significant impact on the automation of knowledge acquisition consists of the simplifying assumptions made by the single strategy learning methods. These include the availability of many input examples (in the case of empirical induction), or the availability of a complete theory of the application domain (in the case of explanation-based learning). These assumptions are most often false in the case of real-world knowledge acquisition applications. NeoDISCIPLE illustrates how one could overcome the limitations of the single-strategy learning methods, by employing a synergistic combination of such methods. It applies explanation-based learning (to attempt building a plausible proof of the input, and to generalize it), learning by abduction (to complete the proof), learning by experimentation (to generate instances of the generalized proof), empirical

* The understanding of the input (i.e. explaining the input to itself) is performed by the problem solver which is part of the learner.

generalization (to generalize the generated instances), and learning by instruction (to acquire new knowledge from the user). Multistrategy learning is now more and more accepted as a main approach to the development of the learning systems (Michalski and Tecuci, 1991).

### *D. Consistency-driven concept formation and refinement*

NeoDISCIPLE starts learning with a preliminary KB which usually consists of incomplete descriptions of object concepts. The defined object concepts constitute an initial incomplete terminology for representing and learning new object concepts, facts, rules etc. Because of this incompleteness, the general knowledge pieces learned by NeoDISCIPLE may have exceptions. For instance, a learned problem solving rule may cover some invalid problem solving episodes, or a learned inference rule may imply some false facts. In order to eliminate these exceptions, new concepts may need to be defined, or the definitions of the existing concepts may need to be refined. For instance, one may eliminate the negative exceptions of a rule by defining a new object concept discriminating between the positive examples and the negative exceptions, and by introducing it into the applicability condition of the rule (Wrobel, 1989; Tecuci, 1991a). Alternatively, one may refine the definition of some object concept with a new feature or relationship shared only by the positive examples of the rule (Tecuci, 1991a). In this way, the hierarchy of object concepts is iteratively developed with the goal of improving the consistency of the learned rules.

### *E. Closed-loop learning*

As shown in Figure 1, the knowledge learned from an input becomes background knowledge which is used in the subsequent learning process, increasing the quality of learning. Therefore, NeoDISCIPLE illustrates a general case of closed-loop learning (Michalski, 1990).

### *F. Synergistic cooperation between the human expert and the learning system*

The learning method of NeoDISCIPLE is based on a cooperation between the human expert and the learning system (see Figure 1) which exploits their complementary abilities. That is, each contributes to the knowledge acquisition process with what the partner can do best.

The human expert, for instance, provides an initial imperfect elementary description of his domain. He is particularly good at providing suitable solutions to problems. He may judge if a solution to a problem is good or not, or if a fact is true or false. He is less adept at providing an explanation of why a particular solution to a problem is good or not, but can easily accept or reject tentative explanations proposed by the system. What is particularly difficult for the human expert is to provide general pieces of knowledge (as, for instance, general problem solving or inference rules) and to continuously monitor the consistency of the knowledge base.

On the other hand, NeoDISCIPLE suggests justifications of the observed facts or examples of problem solving episodes, generalizes them, and iteratively develops and updates the KB, so as to consistently integrate the learned knowledge.

## III. ILLUSTRATION OF THE PROPOSED METHODOLOGY FOR BUILDING EXPERT SYSTEMS: QUESTION-ANSWERING IN GEOGRAPHY

### *A. Definition of the preliminary knowledge base*

We shall illustrate our approach to the automation of knowledge acquisition with the help of an example - building an expert system able to answer questions about geography.

NeoDISCIPLE provides a framework for such a system that consists of a backward chaining theorem prover, and a knowledge base that contains a hierarchy of object concepts (describing explicitly properties and relationships of the geographical objects) and a set of inference rules (for inferring new properties and relationships of objects from other properties and relationships).

To answer a question of the form

Does (wheat GROWS-IN Tunisia) ?

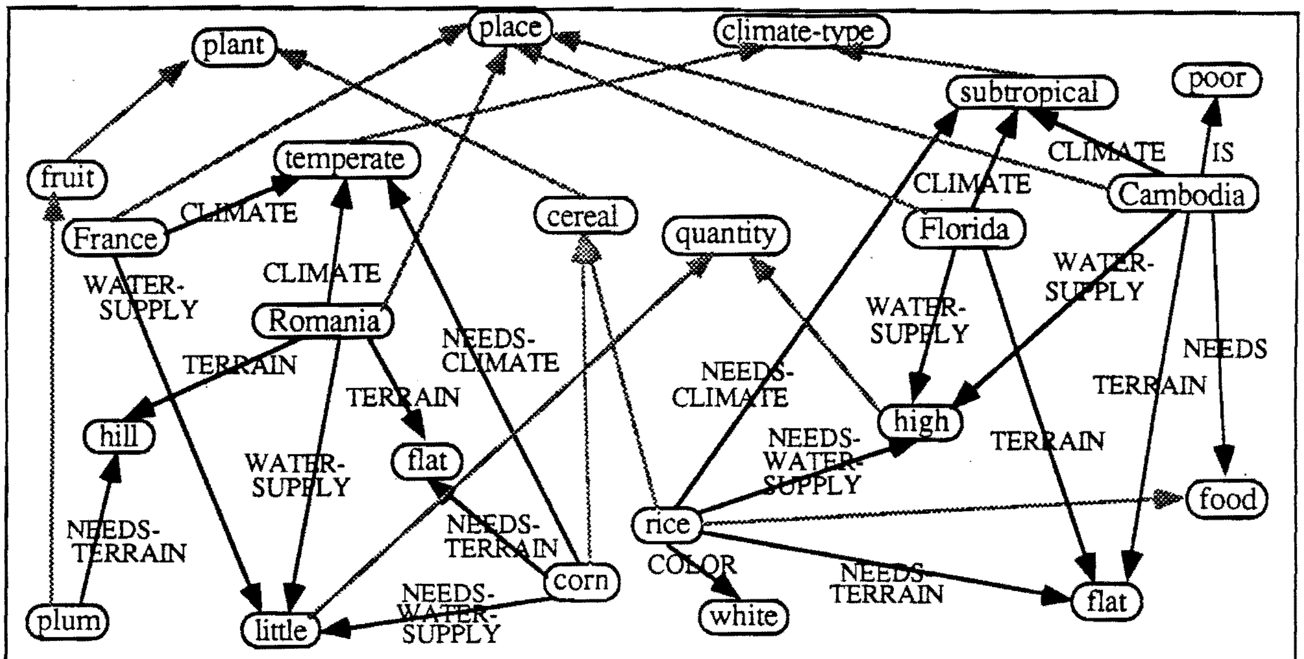
the system will first look into the hierarchy of object concepts. If the above fact is not explicitly represented, then the system will try to infer it from the explicitly represented facts, by building a proof tree like the one in Figure 3.

In the first phase of building the expert system, the human expert has to provide whatever domain knowledge he may easily express, without any interaction with NeoDISCIPLE.

The knowledge provided by the user represents the initial (incomplete and imperfect) knowledge base of the question-answering system. This KB will be extended, updated, and improved by NeoDISCIPLE, through successive interactions with the human expert. During these interactions, the human expert provides new geographical facts, and the system improves the object descriptions and the rules from the KB, or learns new ones, so as to consistently integrate into the KB the information contained in the input.

A sample of the KB is presented in Figure 2. The top part of the figure contains the hierarchy of object concepts which describes the types of the geographical objects, together with their properties and relationships. These concepts are hierarchically organized along the IS-A relationship (for instance, "rice" is both a "cereal" and a "food", and "cereal" is a "plant") which implies that any concept inherits all the properties of its superconcepts. In order to simplify the figure, each IS-A relationship is represented by a grey arrow, and the name of the relationship is no longer attached to the arrow.

In the current version of NeoDISCIPLE it is supposed that the initial object knowledge provided by the human expert is incomplete but correct. This knowledge will be extended during learning by adding new properties and relationships, or even new objects.



**R1:** IF

```
      (y HAS-METEO-COND-FOR x) &          ; if y has meteorological
      (y HAS-TERRAIN-COND-FOR x)          ; and terrain conditions for x
   THEN
      (x GROWS-IN y)                      ; then x grows in y
```

**R2:** IF

*upper bound*

```
     (x IS-A something)&(y IS-A something)&                ; the water supply of y
     (t IS-A something)&(u IS-A something)&                ; is that needed by x,
     (y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&          ; and the climate of y
     (y CLIMATE u)&(x NEEDS-CLIMATE u)                     ; is that needed by x
```

*lower-bound*

```
     (x IS-A fruit)&(y IS-A place)&                        ; the water supply of the place y
     (t IS-A little)&(u IS-A temperate)&                   ; is little, as needed by the fruit x,
     (y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&          ; and the climate of y is temperate
     (y CLIMATE u)&(x NEEDS-CLIMATE u)                     ; as needed by x
  THEN
     (y HAS-METEO-COND-FOR x)                              ; y has meteorological cond for x
```

*with the positive examples*

(x<-plum,y<-Romania,t<-little,u<-temperate) & (x<-grape,y<-France,t<-little,u<-temperate)

**R3:** IF

*upper bound*

```
     (x IS-A something)&(y IS-A something)&(z IS-A something)& ; the terrain of y
     (y TERRAIN z)&(x NEEDS-TERRAIN z)                         ; is that needed by x
```

*lower-bound*

```
     (x IS-A fruit)&(y IS-A place)&(z IS-A hill)&              ; the terrain of a place y is hill,
     (y TERRAIN z)&(x NEEDS-TERRAIN z)                         ; as needed by the fruit y
  THEN
     (y HAS-TERRAIN-COND-FOR x)                                ; y has terrain conditions for x
```

*with the positive examples*

(x<-plum,y<-Romania,z<-hill) & (x<-grape,y<-France,z<-hill)

Figure 2: A sample KB

The bottom part of Figure 2 contains three rules for inferring new properties and relationships of objects from other properties and relationships. Such rules could be provided by the user or could be learned by the system. Therefore, the minimum knowledge that the human expert is required to provide consists of an incomplete hierarchical semantic network of objects. The rules R2 and R3 in Figure 2 are incompletely learned. Instead of an exact condition they specify a plausible version space (Mitchell, 1978; Tecuci, 1988) for the condition, represented by a conjunctive expression that is supposed to be more general than the exact condition (the plausible upper bound), and a conjunctive expression that is supposed to be less general than the exact condition (the plausible lower bound). If the lower bound condition of a rule is satisfied then the system "considers" the conclusion as being true. If the upper bound condition is satisfied, but the lower bound condition is not satisfied, then the conclusion is considered only plausible, needing further evidence in order to be accepted. The reason for calling such a version space plausible is that the learning process takes place in an incomplete and/or partially incorrect representation language (which corresponds to Horn clauses formed with the concepts and the features from the current hierarchical semantic network of objects). Therefore, the bounds may not be strict, allowing for exceptions (negative examples that are covered by the lower bound or positive examples that are not covered by the upper bound). During learning, NeoDISCIPLE will improve the existing rules and will learn new rules. The system keeps all the positive and negative examples and exceptions of the incompletely learned rules. These instances are the main source of knowledge for extending the representation language of the system with new concepts or relationships (Tecuci, 1991a), as well as for accordingly updating the rules.

## *B. Knowledge refinement*

After the initial KB has been provided, NeoDISCIPLE refines it by learning from new input information received from the human expert.

The knowledge refinement problem of NeoDISCIPLE, in this geographical domain, is formulated and illustrated in Table 1.

The human expert has told the system that "(rice GROWS-IN Cambodia)", and NeoDISCIPLE has refined the knowledge base in several respects, so as to consistently integrate, not only this new piece of knowledge, but also similar ones.

First of all, the system has learned two new relevant geographical relationships: "SOIL" and "NEEDS-SOIL". These relationships extend the representational capabilities of the system and are, in fact, used to reexpress the version space of the rule R3.

Secondly, the system has learned new geographical facts like "(Cambodia SOIL fertile-soil)" and "(rice NEEDS-SOIL fertile-soil)".

**Table 1: The knowledge refinement problem**

**Given**

• *Input:* a new fact as, for instance, (rice GROWS-IN Cambodia)

• *Imperfect knowledge base:* rules and facts to be used in understanding of the input as, for instance, those represented in Figure 2.

**Determine**

• *Refined knowledge base:* A KB consistently integrating the information contained in the input.
For instance, as a result of receiving the new input (rice GROWS-IN Cambodia), NeoDISCIPLE refined the KB in Figure 2 by:

*Acquiring new relevant geographical relationships:*
SOIL and NEEDS-SOIL

*Acquiring new basic geographical facts:*
(Cambodia SOIL fertile-soil), (rice NEEDS-SOIL fertile-soil), (Florida SOIL normal-soil),
(Romania SOIL normal-soil), (plum NEEDS-SOIL normal-soil),
(France SOIL normal-soil), (grape NEEDS-SOIL normal-soil)

*Improving the rules R2 and R3 by updating their applicability conditions:*

R2: IF
*plausible upper bound*
(x IS-A something)&(y IS-A something)&
(t IS-A something)&(u IS-A something)&
(y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&
(y CLIMATE u)&(x NEEDS-CLIMATE u)

*plausible lower-bound*
(x IS-A plant)&(y IS-A place)&
(t IS-A quantity)&(u IS-A climate-type)&
(y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&
(y CLIMATE u)&(x NEEDS-CLIMATE u)
THEN
(y HAS-METEO-COND-FOR x)
*with the positive examples*
(x<-plum, y<-Romania, t<-little, u<-temperate)
(x<-grape, y<-France, t<-little, u<-temperate)
(x<-rice, y<-Cambodia, t<-high, u<-subtropical)

R3: IF
*plausible upper bound*
(x IS-A something)&(y IS-A something)&(z IS-A something)&
(y TERRAIN z)&(x NEEDS-TERRAIN z)&
(v IS-A something)&(y SOIL v)&(x NEEDS-SOIL v)

*plausible lower bound*
(x IS-A plant)&(y IS-A place)&(z IS-A terrain-type)&
(y TERRAIN z)&(x NEEDS-TERRAIN z)&
(v IS-A soil-type)&(y SOIL v)&(x NEEDS-SOIL v)
THEN
(y HAS-TERRAIN-COND-FOR x)
*with the positive examples*
(x<-plum, y<-Romania, z<-hill, v<-normal-soil)
(x<-grape, y<-France, z<-hill, v<-normal-soil)
(x<-rice, y<-Cambodia, z<-flat, v<-fertile-soil)
*with the negative example*
(x<-rice, y<-Florida, z<-flat)

Thirdly, it has improved the rules R2 and R3. Indeed, it has discovered a new positive example of the rule R2 and has generalized the plausible lower bound so as to cover this example (the generalized literals are underlined). It has also discovered one positive example and one negative example of the rule R3, and has modified the version space so as to cover the positive example and to reject the negative example. One should notice that, as opposed to the standard version space method (Mitchell, 1978), where the lower bound is never specialized, in this case the lower bound has been specialized by adding the conjunction of literals:

(v IS-A soil-type) & (y SOIL v) & (x NEEDS-SOIL v)

This specialization shows also how the new relationships "SOIL" and "NEEDS-SOIL" extend the representation language of the system.

What is not directly observable in Table 1 is that the improved knowledge base allows the system not only to derive the input fact "(rice GROWS-IN Cambodia)", but also other related facts as, for instance, "(corn GROWS-IN Romania)".

The general knowledge refinement strategy of NeoDISCIPLE could be synthesized as follows:

- the KB contains *explicit object knowledge,* which we call *basic object knowledge* (in the form of a hierarchical semantic network), and *implicit object knowledge* (in the form of inference rules).

- when the system receives a new fact from the human expert, it will try to extend and update its KB so that the current input fact is inferable from the KB. If this is not possible, then the system will interpret the input fact as representing basic object knowledge, and will introduce it explicitly into the semantic network of objects.

The knowledge refinement method of NeoDISCIPLE is presented in the next section and illustrated in the subsequent ones.

### *C. Knowledge reformulation*

When the KB of the system is complete enough and correct enough for providing correct solutions to most of the problems that the system is supposed to encounter, the main emphasis of learning changes from knowledge refinement to knowledge reformulation. The goal of knowledge reformulation is to improve the performance of the expert system.

As will be shown in section VII, an interesting and powerful feature of NeoDISCIPLE is that its knowledge refinement method becomes a knowledge reformulation one, when the KB of the system is complete.

## IV THE KNOWLEDGE REFINEMENT METHOD

The knowledge refinement method of NeoDISCIPLE follows the steps indicated in Figure 1 and detailed in the following (these steps will be illustrated in sections V and VI):

• *Understand the input*

**1.** Build a plausible justification tree T which shows that the input I is a consequence of the knowledge from the KB. The top of this tree is the input I, and the leaves are the facts $F_p,...,F_q$ from the KB that plausible imply I.

**2.** If, in order to build the tree T, the system needed to abduce facts or inference steps (Tecuci and Kodratoff, 1990), then introduce this new knowledge into the KB.

**3.** Let $R_i,...,R_j$ be the rules from the KB that have been used to build the tree T. Generalize (if necessary) the plausible lower bounds of these rules, as little as possible, so as to cover the corresponding inferences from the tree T, and to remain less general then the plausible upper bounds.

• *Generalize the understanding*

**4.** Build the most general plausible generalization $T_u$, of the tree T, by using the upper bound conditions of the rules $R_i,...,R_j$. The top of this tree will be the generalization Ig of the input I, and the leaves will be the generalizations $F_{pu},...,F_{qu}$, of the facts $F_p,...,F_q$.

**5.** Build the most general deductive generalization $T_l$, of the tree T, by using the lower bound conditions of the rules $R_i,...,R_j$. The top of this tree will be the generalization Ig of the input I, and the leaves will be the generalizations $F_{pl},...,F_{ql}$, of the facts $F_p,...,F_q$.

**6.** Build the following plausible version space VP, synthesizing the inferential capabilities of the system with respect to inputs similar with I:

```
IF
        plausible upper bound: $F_{pu}\&...\&F_{qu}$
        plausible lower bound: $F_{pl}\&...\&F_{ql}$
THEN
        $I_g$
```

**While**
The the two bounds of the version space VP are not identical
and
The KB contains an instance of the upper bound that is not an instance of the lower bound
**Do steps 7 through 24**

• *Experiment*

**7.** Find, in the KB, an instance of the plausible upper bound of VP which is not an instance of the plausible lower bound. Let $(F_{px}\&...\&F_{qx}) = \sigma(F_{pu}\&...\&F_{qu})$, where $\sigma$ is a substitution, be this instance. $(F_{px}\&...\&F_{qx})$ is not covered by the plausible lower bound.

**8.** Generate a fact similar with the input I, by applying the substitution $\sigma$ to $I_g$: $\sigma(I_g)$

**9.** Generate the instance of the tree $T_u$, corresponding to the fact $\sigma(I_g)$, by applying $\sigma$ to $T_u$: $\sigma(T_u)$.

• *Verify*

**10.** Ask the human expert if $\sigma(I_g)$ is true. If the answer is Yes then go to step 11. Otherwise, if the answer is No, go to step 13.

**11.** The tree $\sigma(T_u)$ shows new positive instances of the rules $R_i,...,R_j$. Generalize (if necessary) the plausible lower bounds of these rules, as little as possible, so as to cover the corresponding inferences from the tree $\sigma(T_u)$, and to remain less general then the plausible upper bounds.

**12.** Generalize the plausible lower bound of the version space VP, as little as possible, so as to cover $(F_{px}\&...\&F_{qx})$, and to remain less general then the plausible upper bound. Then go to 24.

• *Repair*

**13.** $\sigma(T_u)$ is a wrong proof tree: the leaf predicates are true and the top predicate is false. Ask the user to identify the wrong inference step. Let this step be $A_{kx}$-->$C_{kx}$.

**14.** Let $R_k$ be the rule in the KB the instance of which is $A_{kx}$-->$C_{kx}$. If the plausible lower bound of the rule Rk does not cover $A_{kx}$ then go to 15. Otherwise go to 17.

**15.** Specialize the plausible upper bound of the rule $R_k$, as little as possible, so as no longer to cover Akx and to remain more general then the plausible lower bound.

**16.** Specialize the plausible upper bound of the version space VP, as little as possible, so as no longer to cover ($F_{px}$&...&$F_{qx}$) and to remain more general then the plausible lower bound. Then go to 24.

**17.** Let $A_k$-->$C_k$ be the inference step from the tree T, corresponding to the wrong inference step $A_{kx}$-->$C_{kx}$. Ask the user to correct the inference step $A_k$-->$C_k$, by adding additional left hand side predicates. If the user indicates that the correct inference is $A_k$&$B_k$-->$C_k$ then go to step 18. Otherwise, if the user cannot correct the inference step $A_k$-->$C_k$, go to 23.

**18.** Introduce $B_k$ into the KB.

**19.** Let $B_{ku}$ be the inductive generalization of $B_k$ that is obtained by replacing each object in $B_k$ with a variable. Let $B_{kx1}$,...,$B_{kxn}$ be the instances of $B_{ku}$ corresponding to the known positive examples of $R_k$. Let $B_{kl}$ be a least general generalization of these instances that is less general than $B_{ku}$. Specialize the upper bound and the lower bound of $R_k$ by conjunctively adding $B_{ku}$ and $B_{kl}$, respectively.

**20.** Add $B_{kx1}$,...,$B_{kxn}$ into the KB.

**21.** Update the trees $T_u$ and $T_l$, by using the new bounds of the rule $R_k$.

**22.** Specialize the upper bound and the lower bound of the version space VP by conjunctively adding $B_{ku}$ and $B_{kl}$, respectively. Then go to 24.

**23.** Keep $A_k$-->$C_k$ as a negative exception of the rule $R_k$*.

**24.** Continue the while loop.

The details of this method depend of the current inferential capabilities of the system with respect to the input. We distinguish between three types of such capabilities:

• *Incomplete knowledge about the input*

The system has inference rules allowing it to build a plausible proof of the input. Usually, in such a case, the main result of learning is the improvement of the rules from the KB.

• *Poor knowledge about the input*

The system has no inference rules to build a plausible proof of the input. In such a case, NeoDISCIPLE uses heuristics to propose facts from the KB that directly imply the input. The user has to validate these hypotheses and may indicate additional facts (that are also introduced into the KB). In such a case, the main result of learning is a new inference rule that derives the input.

• *Complete knowledge about the input*

The system has inference rules allowing it to build a complete deductive proof of the input. In such a case, the system does not learn new knowledge. However, it may learn a rule that has the potential of improving the problem solving performance of the system. Therefore, when the KB of the system becomes correct enough and complete enough, the knowledge refinement method becomes a knowledge reformulation one, as will be shown in section VII.

The knowledge refinement problem illustrated in Table 1 corresponds to the case of incomplete knowledge about the input. In the following section we show how NeoDISCIPLE learned these

* The exceptions are used by NeoDISCIPLE in the concept formation and refinement process, as indicated in section 2D and described in (Tecuci, 1991a).

different kinds of knowledge from Table 1. Then we illustrate the learning method in the case of poor and complete knowledge about the input.

## V. KNOWLEDGE REFINEMENT IN THE CASE OF INCOMPLETE KNOWLEDGE

### *A. Understanding the input*

Whenever the system receives a new input, it will try to understand it by building a plausible justification tree which shows that the input is a plausible consequence of the knowledge in the KB. Let us suppose that the current KB is the one from Figure 2 and the input is "(rice GROWS-IN Cambodia)". In this case, the system builds the plausible proof tree from Figure 3. We call this tree "plausible" because it was built by using the upper bound conditions of the rules R2 and R3.

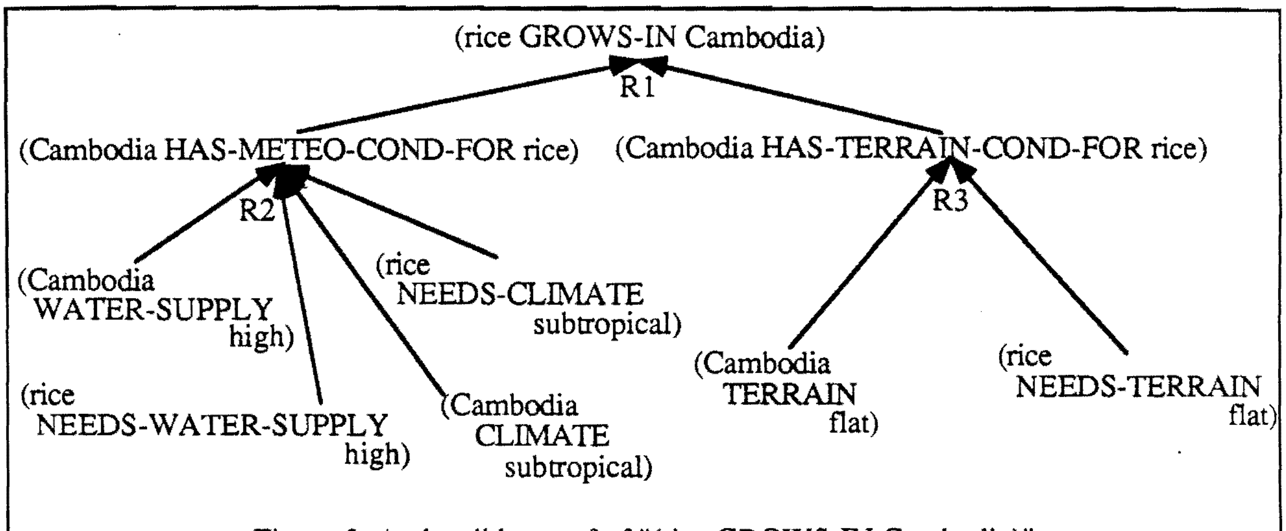

(rice GROWS-IN Cambodia)
R1
(Cambodia HAS-METEO-COND-FOR rice)
(Cambodia HAS-TERRAIN-COND-FOR rice)
R2
R3
(Cambodia WATER-SUPPLY high)
(rice NEEDS-CLIMATE subtropical)
(rice NEEDS-WATER-SUPPLY high)
(Cambodia CLIMATE subtropical)
(Cambodia TERRAIN flat)
(rice NEEDS-TERRAIN flat)

Figure 3: A plausible proof of "(rice GROWS-IN Cambodia)"

Because both the leaves and the top of the tree are true facts, NeoDISCIPLE makes the hypothesis that all the inference steps from this tree are correct. That is, it makes the hypothesis that the instances of R2 and R3 (obtained by applying the upper bound conditions of these rules) are positive instances of these rules. Therefore, the system generalizes the plausible lower bounds of these rules, as little as possible, so as to cover these instances and to remain less general than the corresponding plausible upper bounds (Tecuci and Kodratoff, 1990). For instance, it generalizes "fruit" to "plant", "little" to "quantity", and "temperate" to "climate-type", in the plausible lower bound of R2. These generalizations are made by climbing the generalization hierarchy defined by the IS-A relationship, in the semantic network from Figure 2. NeoDISCIPLE also keeps the instance of R2 from the tree in Figure 3, as a known positive example of R2. Therefore, the version space of rule R2 becomes:

```
R2: IF
      plausible upper bound
      (x IS-A something)&(y IS-A something)&
      (t IS-A something)&(u IS-A something)&
      (y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&
      (y CLIMATE u)&(x NEEDS-CLIMATE u)

      plausible lower-bound
      (x IS-A plant)&(y IS-A place)&
      (t IS-A quantity)&(u IS-A climate-type)&
      (y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&
      (y CLIMATE u)&(x NEEDS-CLIMATE u)
    THEN
      (y HAS-METEO-COND-FOR x)
   with the positive examples
      (x<-plum,y<-Romania,t<-little,u<-temperate)
      (x<-grape,y<-France,t<-little,u<-temperate)
      (x<-rice,y<-Cambodia,t<-high,u<-subtropical)
```

Similarly, NeoDISCIPLE generalizes the lower bound of the version space of R3, so as to cover the instance of R3 from Figure 3:

```
R3:  IF
         plausible upper bound
         (x IS-A something) & (y IS-A something) & (z IS-A something) &
         (y TERRAIN z) & (x NEEDS-TERRAIN z)

         plausible lower-bound
         (x IS-A plant) & (y IS-A place) & (z IS-A terrain-type) &
         (y TERRAIN z) & (x NEEDS-TERRAIN z)
       THEN
         (y HAS-TERRAIN-COND-FOR x)
       with the positive examples
         (x<-plum, y<-Romania, z<-hill)
         (x<-grape, y<-France, z<-hill)
         (x<-rice, y<-Cambodia, z<-flat)
```

One should notice that the improved KB (with the improved rules R2 and R3) deductively implies the input fact "(rice GROWS-IN Cambodia)".

*B. Generalizing the understanding*

The same rules that have been used to prove the validity of the input "(rice GROWS-IN Cambodia)" may be used by the system to prove the validity of other facts of the form "(a GROWS-IN b)". If "(a GROWS-IN b)" is proven by using R1 and the lower bound conditions of R2 and R3, then this fact is considered to be true. However, if "(a GROWS-IN b)" is proven by using at least one of the upper bound conditions of R2 and R3, this fact is considered only plausible. NeoDISCIPLE uses the opportunity offered by the current learning situation to improve the KB until all the implied facts of the form "(a GROWS-IN b)" are derived from exact rules, or from lower bound conditions of incompletely learned rules.

First, NeoDISCIPLE builds two generalizations, $T_u$ and $T_l$, of the proof tree T in Figure 3. $T_u$ is the most general generalization of T, based on the rule R1 and the upper bound conditions of the rules R2 and R3. $T_l$ is the most general generalization of T, based on the rule R1 and the lower bound conditions of the rules R2 and R3.

The generalization technique is similar to that of (Mooney and Bennet, 1986). NeoDISCIPLE first replaces each inference step with the rule that generated it (by using the upper bound of the condition in the case of $T_u$, and the lower bound of the condition in the case of $T_l$). In this way it builds two explanation structures. The one corresponding to $T_u$ is shown in Figure 4.

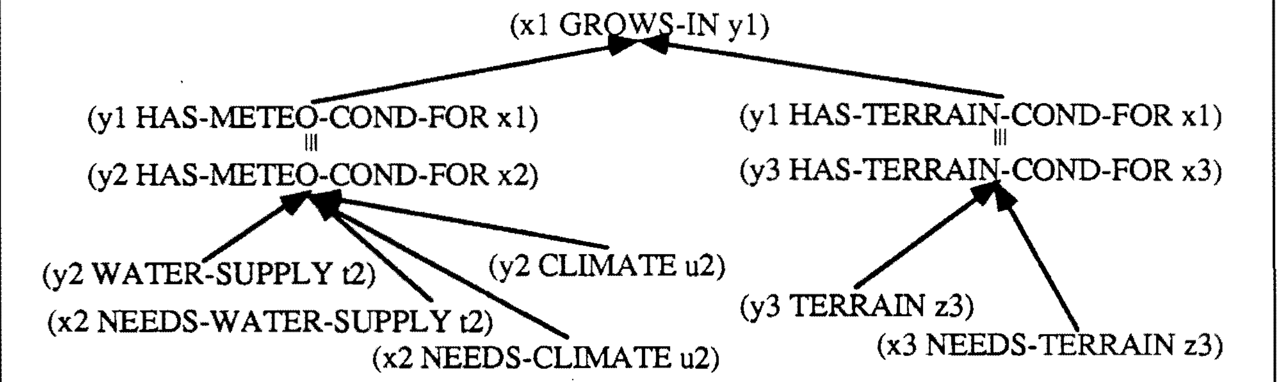

(x1 GROWS-IN y1)
(y1 HAS-METEO-COND-FOR x1)
(y2 HAS-METEO-COND-FOR x2)
(y1 HAS-TERRAIN-COND-FOR x1)
(y3 HAS-TERRAIN-COND-FOR x3)
(y2 WATER-SUPPLY t2)
(y2 CLIMATE u2)
(x2 NEEDS-WATER-SUPPLY t2)
(x2 NEEDS-CLIMATE u2)
(y3 TERRAIN z3)
(x3 NEEDS-TERRAIN z3)

Figure 4: The explanation structure of the input, corresponding to $T_u$

Next NeoDISCIPLE unifies the connection patterns and builds the most general justified generalization of the plausible proof tree in Figure 3 (see Figure 5).

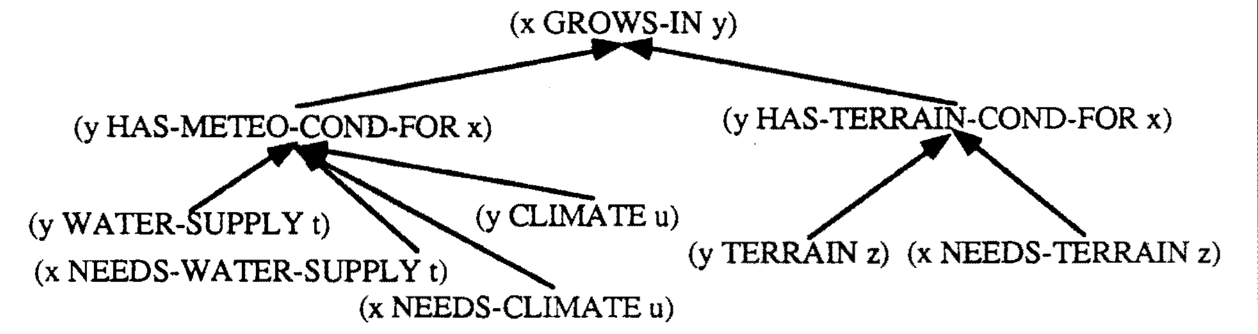


Figure 5: The generalization $T_u$ of the plausible proof tree in Figure 3

Similarly, the system builds the generalized tree $T_l$, which is shown in Figure 6. It should be noticed that the system has built this tree by using the lower bounds of the improved rules R2 and R3.

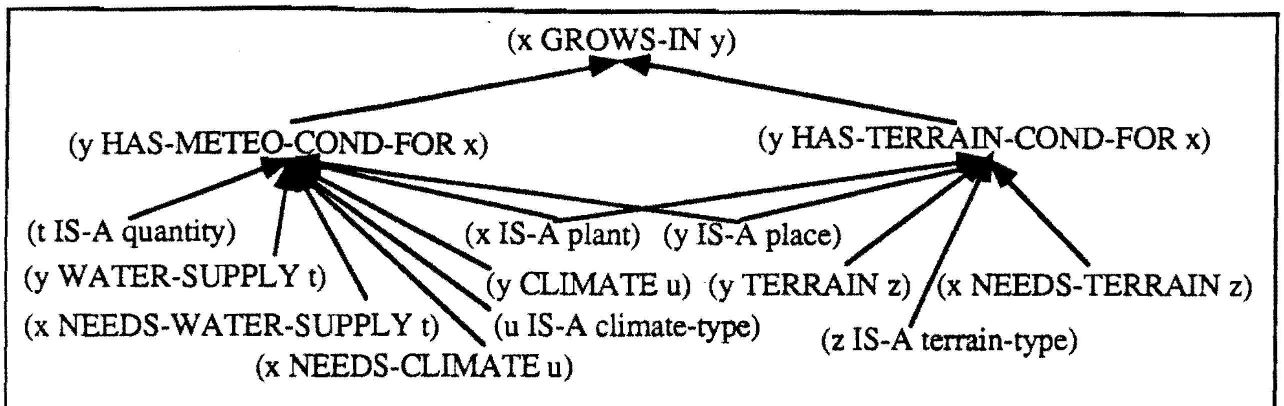


Figure 6: The generalization $T_l$ of the plausible proof tree in Figure 3

The general proof tree $T_u$ in Figure 5 shows that the system will consider plausible all the facts of the form "(x GROWS-IN y)", for all x and y such as the leaves of the tree are facts explicitly represented into the KB. Among these, the system will consider to be true, without looking for any other support, those facts for which x and y are such as the leaves of the tree $T_l$ (see Figure 6) are facts explicitly represented into the KB. To synthesize these inferential capabilities, NeoDISCIPLE builds the plausible version space in Figure 7. The plausible upper bound of this version space corresponds to the leaves of the tree $T_u$, the plausible lower bound corresponds to the leaves of the tree $T_l$, and the conclusion corresponds to the top of these trees:

```
IF
    plausible upper bound
    (x IS-A something) & (y IS-A something) & (z IS-A something) &
    (t IS-A something) & (u IS-A something) &
    (y WATER-SUPPLY t) & (x NEEDS-WATER-SUPPLY t) &
    (y CLIMATE u) & (x NEEDS-CLIMATE u) &
    (y TERRAIN z) & (x NEEDS-TERRAIN z)
    plausible lower bound
    (x IS-A plant) & (y IS-A place) & (z IS-A terrain-type) &
    (t IS-A quantity) & (u IS-A climate-type) &
    (y WATER-SUPPLY t) & (x NEEDS-WATER-SUPPLY t) &
    (y CLIMATE u) & (x NEEDS-CLIMATE u) &
    (y TERRAIN z) & (x NEEDS-TERRAIN z)
THEN
    (x GROWS-IN y)
```

Figure 7: A plausible version space synthesizing the inferential capabilities of the system with respect to the facts of the form "(x GROWS-IN y)"*.

* It should be noticed that the system may be able to prove "(x GROWS-IN y)" by building other justification trees. Considering all the plausible proof tree, however, would not be computational feasible, and would require an unacceptable long interaction with the human expert.

## *C. Experimentation*[*]

The version space in Figure 7 synthesizes the inferential capabilities of the system with respect to the facts of the form "(x GROWS-IN y)". To improve these capabilities, the system looks into the KB for instances of the upper bound that are not instances of the lower bound. For each such instance it shows the user the corresponding inferred fact (see Figure 8), asking if it is true or false:

"Does (rice GROWS-IN Florida) ?" No

Receiving an answer, the system updates the KB such that the true facts are inferred, and the false facts are not.

The version space in Figure 7 serves both for generating the facts of the form "(x GROWS-IN y)", and for determining the end of the learning process. To justify this last point, let us anticipate that, during learning, the lower bound of the version space is generalized so as to cover the generated facts that are accepted by the user (the positive examples), and the upper bound is specialized so as no longer to cover the generated facts that are rejected by the user (the negative examples). The learning process will stop in one of the following situations: a) the bounds of the version space in Figure 7 become identical; b) the bounds are not identical, but the KB no longer contains any instance of the upper bound of the version space that is not an instance of the lower bound. Therefore, no new facts of the form "(x GROWS-IN y)" can be generated.

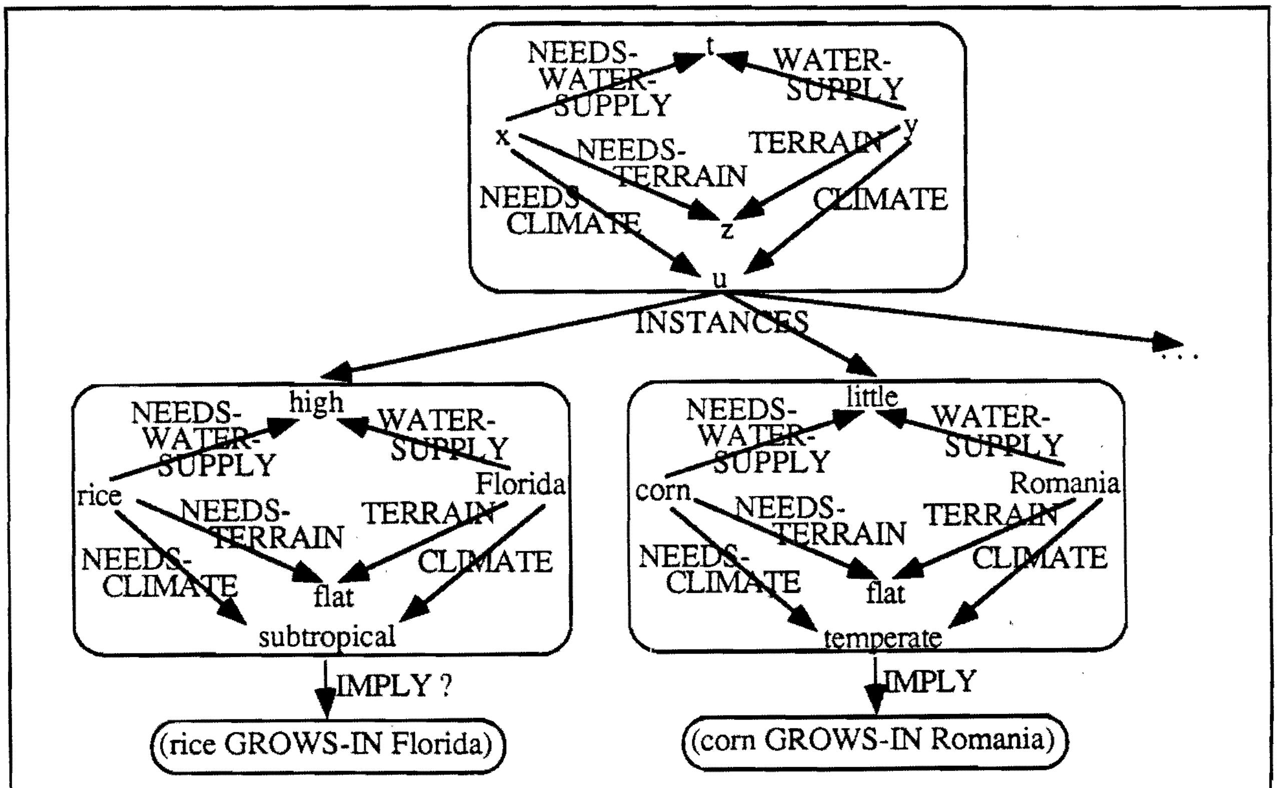
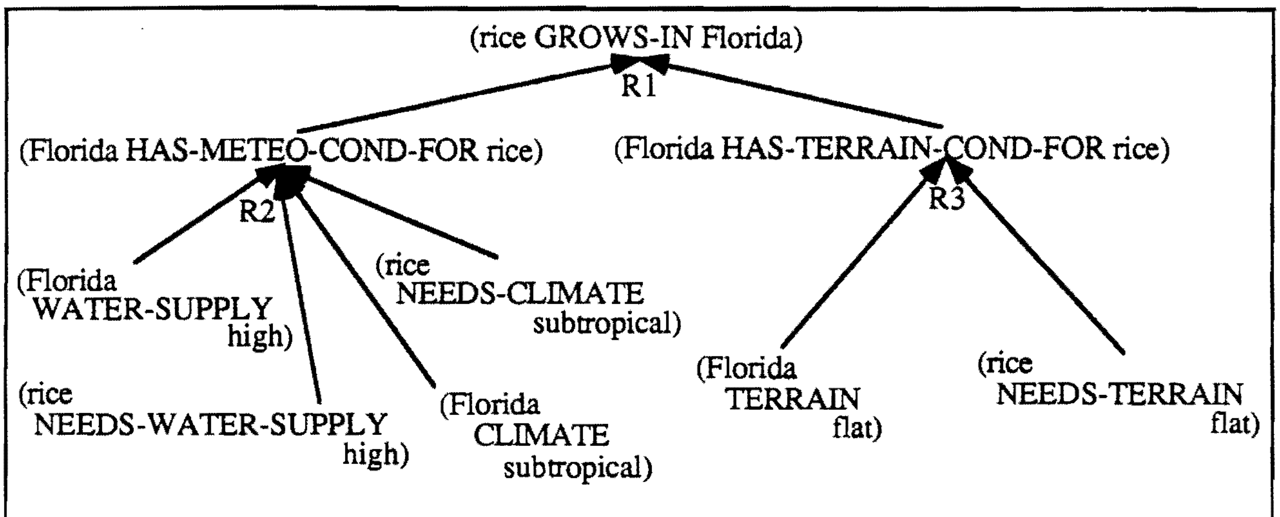


Figure 8: Generation of facts similar with the input one.

[*] Experimentation in NeoDISCIPLE is a form of learning by analogy, as shown in (Tecuci and Kodratoff, 1990).

Let us consider that "(rice GROWS-IN Florida)" is the first generated fact. For each such fact, the system determines the corresponding instance of the tree $T_u$ (see Figure 9).


(rice GROWS-IN Florida)
R1
(Florida HAS-METEO-COND-FOR rice)
(Florida HAS-TERRAIN-COND-FOR rice)
R2
R3
(Florida WATER-SUPPLY high)
(rice NEEDS-CLIMATE subtropical)
(rice NEEDS-WATER-SUPPLY high)
(Florida CLIMATE subtropical)
(Florida TERRAIN flat)
(rice NEEDS-TERRAIN flat)
Figure 9: Instance of the tree $T_u$ corresponding to the generated fact "(rice GROWS-IN Florida)"

## *D. Verification*

If the user states that "(rice GROWS-IN Florida)" is a true fact, then NeoDISCIPLE treats the tree in Figure 9 in the same way it treated the tree from Figure 3. That is, it considers the instances of the rules R2 and R3 from this tree as being new positive examples of these rules, and generalizes their plausible lower bounds, as little as possible, so as to cover these examples. Moreover, it will generalize the plausible lower bound of the version space in Figure 7, so as to cover the leaves of the tree from Figure 9, and therefore to "deductively" infer "(rice GROWS-IN Florida)". However, the user states that the fact "(rice GROWS-IN Florida)" is false, and therefore the KB should be repaired (because it entails a false fact).

## *E. Repair*

The proof tree from Figure 9 is wrong, because the leaf literals are true and the top literal is not. It follows that some of the inferences made are incorrect. To detect them, the system and the user follow the proof tree from bottom up. If the user states that the consequent of a certain inference step is not true, then the corresponding inference may be the faulty one. In this case, the user states that the fact "(Florida HAS-TERRAIN-COND-FOR rice)" is not true. This means that the following inference step (which is an instance of the plausible upper bound of the rule R3) is wrong:

(rice NEEDS-TERRAIN flat) & (Florida TERRAIN flat)
--> (Florida HAS-TERRAIN-COND-FOR rice)

In such a case, the system will attempt to specialize the plausible upper bound of the rule R3, as little as possible, so as no longer to cover the wrong inference and to remain more general than the plausible lower bound. Also, it will attempt to specialize the plausible upper bound of the version space in Figure 7, as little as possible, so as no longer to cover the leaves of the tree in Figure 9 and to remain more general then the lower bound.

However, none of the above specializations is possible. Indeed, the plausible lower bound of the rule R3 covers the wrong inference and the same is true for the lower bound in Figure 7. This shows that the current representation language of the system is incomplete because it does not contain any expression (any lower bound) covering the positive examples of the rule R3 and rejecting the negative example. Therefore, the solution in this case is to extend the representation language with new predicates, and then to update the rule R3 and the version space in Figure 7. To this purpose, the system asks the user to give an explanation of the above failure:

*Can you tell me why* "Not(Florida HAS-TERRAIN-COND-FOR rice)", (system)
*in spite of the fact that*
"(rice NEEDS-TERRAIN flat) & (Florida TERRAIN flat)" ?
*The explanation is:* (human expert)
"(rice NEEDS-SOIL fertile-soil) & (Florida SOIL normal-soil)"

*Can you provide the corresponding piece of explanation for* (system)
"(Cambodia HAS-TERRAIN-COND-FOR rice)" ?
"(rice NEEDS-SOIL fertile-soil) & (Cambodia SOIL fertile-soil)" (human expert)

The above sample dialog between the system and the human expert illustrates how, by asking easy to answer questions, the system may extract many useful pieces of knowledge from the expert.

First of all, the system has learned two new relationships, "SOIL" and "NEEDS-SOIL", as well as new facts (expressed with these relationships), that extend the representation language of the system.

Secondly, it shows that the inference step

(rice NEEDS-TERRAIN flat) & (Cambodia TERRAIN flat)
--> (Cambodia HAS-TERRAIN-COND-FOR rice)

from the plausible proof tree in Figure 3, is incomplete, and indicates how it should be corrected:

(rice NEEDS-TERRAIN flat) & (Cambodia TERRAIN flat) &
(rice NEEDS-SOIL fertile-soil) & (Cambodia SOIL fertile-soil)
--> (Cambodia HAS-TERRAIN-COND-FOR rice)

Having discovered the wrong inference and the means of correcting it, NeoDISCIPLE will next correct the inference rule R3, that produced this inference, so as no longer to generate it.

First, the system inductively generalizes the additional condition

(rice NEEDS-SOIL fertile-soil) & (Cambodia SOIL fertile-soil)

by replacing each object with a variable, thus obtaining

(y SOIL v) & (x NEEDS-SOIL v)

and conjuntively adds this expression to the upper bound of R3.

Then the system determines all the instances of the above general expression, corresponding to the known positive instances of R3:

(Romania SOIL v) & (plum NEEDS-SOIL v)
(France SOIL v) & (grape NEEDS-SOIL v)

Each such instance is shown to the human expert, that is asked to provide the corresponding value for the variable v:

(Romania SOIL v) & (plum NEEDS-SOIL v) & (v IS-A normal-soil)
(France SOIL v) & (grape NEEDS-SOIL v) & (v IS-A normal-soil)

The above expressions, together with

(Cambodia SOIL v) & (rice NEEDS-SOIL v) & (v IS-A fertile-soil)

correspond to the all known positive examples of the rule R3. Therefore, NeoDISCIPLE determines a least general conjunctive generalization of them (Tecuci and Kodratoff, 1990) and conjunctively adds it to the lower bound of R3. Consequently, the version space of R3 becomes:

```
R3: IF
        plausible upper bound
        (x IS-A something)&(y IS-A something)&(z IS-A something)&
        (y TERRAIN z)&(x NEEDS-TERRAIN z)&
        (v IS-A something)&(y SOIL v)&(x NEEDS-SOIL v)

        plausible lower bound
        (x IS-A plant)&(y IS-A place)&(z IS-A terrain-type)&
        (y TERRAIN z)&(x NEEDS-TERRAIN z)&
        (v IS-A soil-type)&(y SOIL v)&(x NEEDS-SOIL v)
    THEN
        (y HAS-TERRAIN-COND-FOR x)
    with the positive examples
        (x<-plum, y<-Romania, z<-hill, v<-normal-soil)
        (x<-grape, y<-France, z<-hill, v<-normal-soil)
        (x<-rice, y<-Cambodia, z<-flat, v<-fertile-soil)
    with the negative example
        (x<-rice, y<-Florida, z<-flat)*
```

The system introduces into the KB all the factual knowledge learned: (Romania SOIL normal), (plum NEEDS-SOIL normal), (grape NEEDS-SOIL normal), and (France SOIL normal).

The same type of modifications that have been made to the version space of R3 are also made to the version space in Figure 7. The new version space is shown in Figure 10.

With the KB from Figure 2 the knowledge acquisition process stops here because there is no other fact of the form "(x GROWS-IN y)" to be generated (i.e. there is no instance of the upper bound of the version space in Figure 10, which is not an instance of the lower bound). For example, the fact "(corn GROWS-IN Romania)" is not shown to the user because it is also inferred by the lower bound of version space, and is therefore considered to be true.

* The value for v in the negative example is not defined. The predicate (Florida SOIL v) would require the value "normal-soil", and the predicate (rice NEEDS-SOIL v) would require the value "fertile-soil".

It may also be noticed that the version space in Figure 10 does not contain any new knowledge because whatever facts it can infer could also be inferred by applying the rules R1, R2, and R3.

```
IF
    plausible upper bound
    (x IS-A something) & (y IS-A something) & (z IS-A something) &
    (t IS-A something) & (u IS-A something) &
    (y WATER-SUPPLY t) & (x NEEDS-WATER-SUPPLY t) &
    (y CLIMATE u) & (x NEEDS-CLIMATE u) &
    (y TERRAIN z) & (x NEEDS-TERRAIN z)
    (v IS-A something)&(y SOIL v)&(x NEEDS-SOIL v)

    plausible lower bound
    (x IS-A plant) & (y IS-A place) & (z IS-A terrain-type) &
    (t IS-A quantity) & (u IS-A climate-type) &
    (y WATER-SUPPLY t) & (x NEEDS-WATER-SUPPLY t) &
    (y CLIMATE u) & (x NEEDS-CLIMATE u) &
    (y TERRAIN z) & (x NEEDS-TERRAIN z)
    (v IS-A soil-type)&(y SOIL v)&(x NEEDS-SOIL v)
THEN
    (x GROWS-IN y)
```

Figure 10: Updated version space for the inference of "(x GROWS-IN y)".

## VI KNOWLEDGE REFINEMENT IN THE CASE OF POOR KNOWLEDGE

### *A. Understanding the input*

Let us now suppose that the current KB contains only the semantic network from the top of Figure 2. This represents an example of poor knowledge about the input "(rice GROWS-IN Cambodia)" because the system does not have rules to build a plausible proof of the input. However, it makes the hypothesis that the input fact is a *direct* consequence of other facts that are explicitly represented into the semantic network. It therefore uses heuristics to select such facts, and to propose them as partial explanations of the input, to be validated by the user, who may himself indicate other facts. One used heuristic is to propose as plausible explanations of input validity the relationships between the objects from the input (rice and Cambodia), as shown in the following sample dialog (see also (Tecuci and Kodratoff, 1990)):

*Are the following relationships explanations for* "(rice GROWS-IN Cambodia)":
(rice NEEDS-TERRAIN flat) & (Cambodia TERRAIN flat) ? *Yes*
(rice IS-A food) & (Cambodia NEEDS food) ? *No*
(rice NEEDS-WATER-SUPPLY high) & (Cambodia WATER-SUPPLY high) ? *Yes*
(rice NEEDS-CLIMATE subtropical) & (Cambodia CLIMATE subtropical) ? *Yes*

The pieces of explanations marked by a user's yes represent the facts from the KB that imply the input and therefore define the following plausible proof:

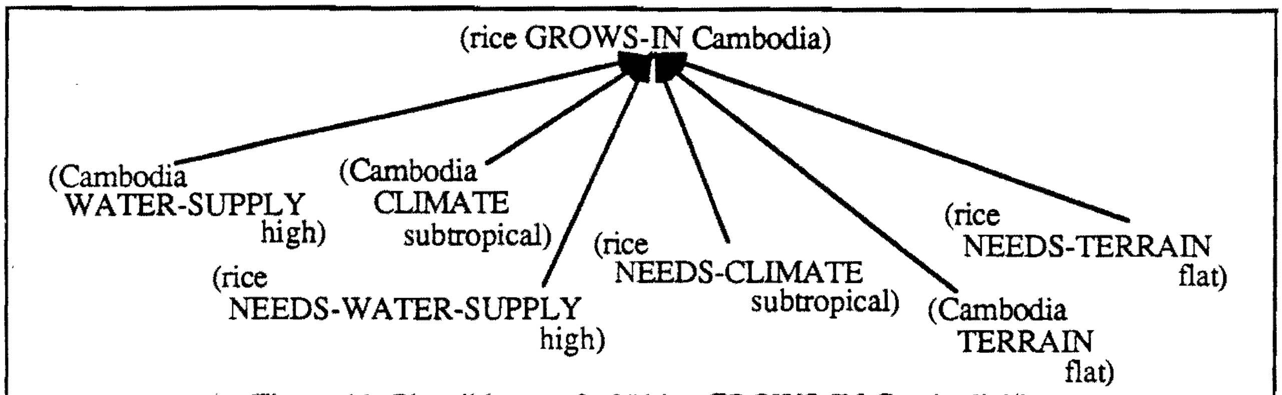

(rice GROWS-IN Cambodia)
(Cambodia WATER-SUPPLY high)
(Cambodia CLIMATE subtropical)
(rice NEEDS-WATER-SUPPLY high)
(rice NEEDS-CLIMATE subtropical)
(rice NEEDS-TERRAIN flat)
(Cambodia TERRAIN flat)

Figure 11: Plausible proof of "(rice GROWS-IN Cambodia)"

*B. Generalizing the understanding*

NeoDISCIPLE inductively generalizes the plausible proof tree in Figure 11 by simply turning all the constants into variables:

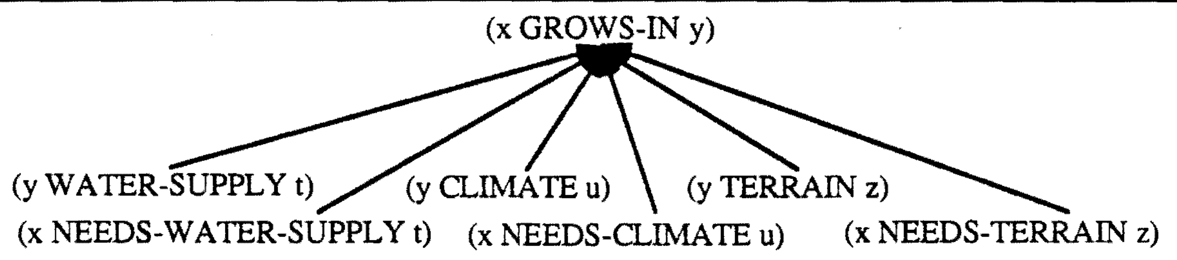


Figure 12: Inductive generalization of the plausible proof in Figure 11.

The plausible proof tree in Figure 11 and its inductive generalization from Figure 12 define an initial version space for a new inference rule:

```
IF
    plausible upper bound
    (x IS-A something) & (y IS-A something) & (z IS-A something) &
    (t IS-A something) & (u IS-A something) &
    (y WATER-SUPPLY t) & (x NEEDS-WATER-SUPPLY t) &
    (y CLIMATE u) & (x NEEDS-CLIMATE u) &
    (y TERRAIN z) & (x NEEDS-TERRAIN z)
    plausible lower bound
    (x IS-A rice) & (y IS-A Cambodia) & (z IS-A flat) &
    (t IS-A high) & (u IS-A subtropical) &
    (y WATER-SUPPLY t) & (x NEEDS-WATER-SUPPLY t) &
    (y CLIMATE u) & (x NEEDS-CLIMATE u) &
    (y TERRAIN z) & (x NEEDS-TERRAIN z)
THEN
    (x GROWS-IN y)
with the positive example
    (x<-rice, y<-Cambodia, z<-flat, t<-high, u<-subtropical)
```

Figure 13: A plausible version space for a new inference rule.

*C. Experimentation*

The knowledge acquisition process continues as in the case of incomplete knowledge with the only difference that the plausible proof trees considered contain a single inference step, corresponding to the above rule.

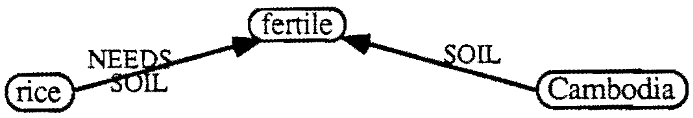
The result of this process is the learning of a rule from the above version space and the acquisition of new object knowledge.

Let us notice that in this case, the rule to be learned serves also as the version space that guides the learning process.

It is in this way that NeoDISCIPLE learns new rules that increase its inferential capabilities.

## VII. KNOWLEDGE REFORMULATION

When the KB of the system is complete and correct, the knowledge refinement method becomes a knowledge reformulation one. Let us suppose, for instance, that the semantic network in Figure 2 has been augmented with the relationships from the top of Figure 14, and the incompletely learned rules R2 and R3 have evolved to the rules R2 and R3 from Figure 14.



```
R2:  IF
        (x IS-A plant)&(y IS-A place)&(t IS-A quantity)&(u IS-A climate-type)&
        (y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&
        (y CLIMATE u)&(x NEEDS-CLIMATE u)
     THEN
        (y HAS-METEO-COND-FOR x)

R3:  IF
        (x IS-A plant)&(y IS-A place)&(z IS-A terrain-type)&(v IS-A soil-type)
        (y TERRAIN z)&(x NEEDS-TERRAIN z)&(y SOIL v)&(x NEEDS-SOIL v)
     THEN
        (y HAS-TERRAIN-COND-FOR x)
```

Figure 14. Knowledge to be added to the one in Figure 2 in order to transform it into complete knowledge with respect to the input "(rice GROWS-IN Cambodia)".

The resulting KB is "complete" with respect to the input fact "(rice GROWS-IN Cambodia)" because it allows the system to build a deductive proof of it. In such a case, the learning method of NeoDISCIPLE reduces to pure explanation-based learning (Mitchell et al., 1986; DeJong and Mooney, 1986). Indeed, NeoDISCIPLE builds a tree similar to the one from Figure 3, except that each inference step is a deduction, and the tree is a logical proof. Then, by using the general form of the rules R1, R2, and R3, it builds a generalized proof tree, similar to the one from Figure 5.

Because this generalized tree is a logical proof, its leaves deductively imply the top. Therefore, the system may generate a rule, the condition of which are the leaves of the tree, and the conclusion of which is the top of the tree:

```
IF                                                          ; If
   (x IS-A plant)&(y IS-A place)&(z IS-A terrain-type)&     ; the water supply of the place y
   (t IS-A quantity)&(u IS-A climate-type)&(v IS-A soil-type)& ; is that needed by the plant x,
   (y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)&             ; and the climate of y is that
   (y CLIMATE u)&(x NEEDS-CLIMATE u)&                       ; needed by x, and the terrain of
   (y TERRAIN z)&(x NEEDS-TERRAIN z)&                       ; y is that needed by x, and
   (y SOIL v)&(x NEEDS-SOIL v)                              ; the soil of y is that needed by y
THEN                                                        ; then
   (x GROWS-IN y)                                           ; x grows in y
```

One should notice that, in such a case, the system learns no new knowledge. It simply concentrates the knowledge contained in the rules R1, R2, and R3, into a new rule. This new rule allows the system to immediately infer facts of the form "(x GROWS-IN y)", without needing to build a proof tree like the one in Figure 3. Thus, the learned rule has a positive effect on the efficiency of the system. However, the addition of a new rule that does not contain any new knowledge has also a negative effect on the efficiency of the rule interpreter that may have more rules to test for solving a problem. Therefore, the decision on wether to keep or not the learned rule should be based on its utility, that takes into account both its positive effects and its negative ones. Initial results on the utility problem (Minton, 1990) suggest that the best performance is obtained when the system learns a small number of such rules that are sufficient for solving most problems. The utility problem in NeoDISCIPLE is a topic for future research.

## VIII. EXPERIMENTAL EVIDENCE

A version of NeoDISCIPLE is implemented in Common Lisp and runs on Macintosh. In order to test its feasibility and generality, we have used it to build small knowledge bases for several types of expertise domains. Two of them are briefly described in the following. Another application of NeoDISCIPLE (developing a knowledge base for planing the manufacturing of a loudspeaker) is described in (Tecuci and Kodratoff, 1990).

### *A. Action planning for robot domestic tasks*

The framework for this domain consists of a hierarchical planner that decomposes "complex" robot commands into simple actions executable by the robot.

First the system has been provided with a preliminary KB. This KB contained incomplete descriptions of concepts representing some of the objects from the robot world. Then the system learned to solve problems by analyzing examples of problem solving episodes.

For instance, from the following problem solving episode

```
    The problem
            TAKE clean-cup1                   ; to take clean-cup1
    has the solution
            OPEN cabinet                      ; the robot has to open the cabinet
            TAKE clean-cup1 FROM cabinet      ; and to take the cup from it
```

the system has learned the rule

```
    IF                                                        ; If
            (x IS-A movable-obj) & (y IS-A container) &       ; x is a movable object
            (x IS-IN y) & (y IS closed)                       ; from a closed container y
    THEN                                                      ; then
    the problem
            TAKE x                                            ; to take x
    has the solution
            OPEN y                                            ; the robot has to open y
            TAKE x FROM y                                     ; and to take x from it
```

The concept "movable-obj" represents the set of objects that could be moved by the robot and has been defined by the system in order to eliminate the negative exceptions that were covered by the learned rule (Tecuci, 1991a).

*B. Qualitative prediction in Chemistry*

NeoDISCIPLE was also used to developed a preliminary model of inorganic Chemistry consisting of elementary knowledge about some basis, acids and salts, but no knowledge about chemical reactions.

Starting from the reaction "$NaOH + HCl \rightarrow H_2O + NaCl$", the system learned that, in general, by combining a base with an acid, one get water and salt:

```
IF
   (b IS-A base)                                   ; by combining a base b, composed of
   (b COMPOSED-OF x1) (b COMPOSED-OF x2)           ; a hydroxide x1 and a metal x2,
   (a IS-A acid)                                   ; with an acid a, composed of
   (a COMPOSED-OF x3) (a COMPOSED-OF x4)           ; a hydrogen x3 and a metalloid x4,
   (w IS-A H2O)                                    ; one gets water w,
   (w COMPOSED-OF x1) (COMPOSED-OF x3)             ; composed of x1 and x3,
   (s IS-A salt)                                   ; and salt s,
   (s COMPOSED-OF x2) (s COMPOSED-OF x4)           ; composed of x2 and x4.
   (s (COMPOSED-OF ANION-OF) a)                    ; one component of the salt s is
   (s (COMPOSED-OF CATION-OF) b))                  ; an anion of the acid a, and
   (x1 IS-A OH)                                    ; the other component of the salt s is
   (x2 IS-A METAL)                                 ; a cation of the base b
   (x3 IS-A H)
   (x4 IS-A METALLOID)
THEN
   b + a --> w + s
```

Such a rule is used to predict the results of other chemical reactions. For instance, it will predict that combining KOH with $H_2SO_4$ will result in $H_2O$ and $K_2SO_4$.

## *C. Conclusions of the experiments*

From these experiments, we have concluded that it is not very difficult to build a small knowledge base with NeoDISCIPLE.

The dialogue with the user is based on "intelligent", specific, and easy to answer questions. Also, NeoDISCIPLE minimizes the number of questions asked by carefully generating only those facts or problem solving episodes that are most likely to advance the learning process. In the experiments performed, NeoDISCIPLE needed to generate less than 10 examples (as those from Figure 8) during a learning session. However, the knowledge base may sometimes allow the generation of thousands of such examples. If, in a certain critical application, all these examples need to be tested, then this may require a lot of time from the human expert. Therefore, new methods have to be devised to test the examples independently of the expert (for instance, by comparing them with a database of cases), or to select for testing only the examples that have the highest likelihood of contradicting the knowledge in the KB.

A basic source of knowledge for learning is the hierarchical semantic network which provides the generalization language. Therefore, an application domain for which one cannot define a "rich enough" semantic network is not suitable for NeoDISCIPLE.

The integrated learning method of NeoDISCIPLE outperforms any of the constituent single-strategy methods in that it is able to learn in situations in which they were insufficient. However, NeoDISCIPLE still suffers from the basic limitation of the learning systems: if the bias built into the system is incorrect, the system will fail to learn properly. In the current version of NeoDISCIPLE, the rules may be partially incorrect. However, the initial semantic network (which contributes significantly to the system's learning bias) is supposed to be incomplete but correct. During learning, the definition of the object concepts may be refined and even new concepts may be defined (Tecuci, 1991c). While this may improve the initial bias, it will not modify it drastically. A better way to surmount this limitation is to perform not only additions to the semantic network, but also deletions. We therefore plan to develop the learning method of NeoDISCIPLE so as to start with a semantic network (which is not only incomplete, but also partially incorrect), and to gradually improve it.

## IX. RELATED RESEARCH

NeoDISCIPLE is an extension and a generalization of DISCIPLE (Tecuci, 1988; Tecuci and Kodratoff, 1990). While many of the learning techniques of DISCIPLE and NeoDISCIPLE are similar, the learning problems considered are different. DISCIPLE accepts as input an example of a problem solving episode, represented by a problem P and its solution S, and learns a general problem solving rule, allowing the system to solve problems similar to P, by proposing solutions similar to S. NeoDISCIPLE may accept as input not only an example of a problem solving episode, but also an example of a concept, or even a ground fact (as illustrated in this paper). The main goal of the system is no longer to learn a rule covering the input, but to extend, update and improve the KB so as to consistently integrate the information contained in the input. In particular, if the knowledge of the system with respect to the input is poor, this goal is achieved by learning a rule. Therefore, NeoDISCIPLE includes DISCIPLE.

NeoDISCIPLE is also related to the learning systems that integrate different learning strategies (e.g., Bergadano and Giordana, 1990; Danyluk, 1987; De Raedt and Bruynooghe, 1991; Flann and Dietterich, 1989; Genest et al., 1990; Hirsh, 1989; Lebowitz, 1986; Minton and Carbonell, 1987; Mooney and Ourston, 1989; Pazzani, 1988; Shavlik and Towell, 1989; Tecuci and Kodratoff, 1990; Whitehall, 1990; Wilkins, 1990). While most of these systems are concerned with the integration of explanation-based learning and empirical inductive learning, NeoDISCIPLE extends the range of the integrated learning strategies by also including learning by abduction, learning by experimentation, constructive induction, and learning by instruction.

Although the learning strategies employed by NeoDISCIPLE are well-known, their integration was done by making several significant developments to some of them. For instance, the empirical inductive method of NeoDISCIPLE is an extension of the version-space method (Mitchell, 1978) in that the version space is no longer strict, but plausible. In the case of Mitchell's method the lower bound and the upper bound of the version space are exact boundaries of the version space. Consequently, during learning, the lower bound can only be generalized and the upper bound can only be specialized. On the contrary, in the case of NeoDISCIPLE, these bounds are only plausible (i.e. approximations of the exact bounds). Therefore, during learning, each of them can be both generalized and specialized.

The learning method of NeoDISCIPLE was mostly influenced by the explanation-based learning strategy (DeJong and Mooney, 1986; Mitchell, Keller and Kedar-Cabelli, 1986). A significant merit of this learning strategy is the identification of the importance of the understanding (i.e. explaining to itself) in learning. While EBL defines the notion of explanation only in the context of a complete domain knowledge, NeoDISCIPLE extends this notion to the cases in which the knowledge of the system with respect to the input is incomplete or even poor. As EBL, NeoDISCIPLE still requires a single input example to learn because it is able to generate itself new examples. This example generation capability is an important feature of NeoDISCIPLE that was originally introduced by DISCIPLE.

NeoDISCIPLE also extends constructive induction (Michalski, 1983). In constructive induction, the representation language of the learning system is extended with new terms which are a function of the known terms. The main goal is to find terms that simplify the descriptions of the learned concepts. Constructive induction introduces new terms based of the intentional definitions of the concepts. It is thus a kind of knowledge reformulation that does not extend the representational capabilities of the system. NeoDISCIPLE may introduce new terms by also using the extensional definitions of the concepts (concepts defined by the set of the instances covered). As in BLIP (Wrobel, 1989), NeoDISCIPLE introduces new terms in order to reduce the number of the exceptions of the learned rules. However, the methods employed by NeoDISCIPLE (Tecuci, 1991a) are more flexible and adaptive to the current knowledge of the system. As opposed to BLIP that always introduces a single new concept that eliminate all the exceptions of a rule, NeoDISCIPLE may introduce several concepts, as well as several concept features.

NeoDISCIPLE may also be compared with systems that have the same goal of improving a knowledge base as, for instance, ODYSSEUS (Wilkins, 1990) and EITHER (Mooney and Ourston, 1991).

The main idea of ODYSSEUS is to complete an explanation of an action of an expert, by abducing a fact that is also introduced into the KB. NeoDISCIPLE also uses abduction, when it has incomplete knowledge about the input. It may however abduce not only facts, but also inference steps (Tecuci and Kodratoff, 1990). While ODYSSEUS is concerned only with completing the explanation of the current input, NeoDISCIPLE also attempts to generalize it, so as to explain similar facts. Thus, NeoDISCIPLE learns more from an input than ODYSSEUS. Although both ODYSSEUS and NeoDISCIPLE deal with uncertain rules, ODYSSEUS represents uncertainty through numeric certainty factors, and NeoDISCIPLE represents uncertainty symbolically, through plausible version spaces (Kodratoff et al., 1990).

EITHER is an integrated learning system that uses deduction, abduction, and empirical induction to improve a KB, so as to correctly classify a given set of positive and negative examples of some concepts. While EITHER is an autonomous learner, NeoDISCIPLE is an interactive one that tries to improve a KB so as to produce the same answers as a human expert. Moreover, EITHER attempts to improve globally the KB (for instance, by considering all the possible explanations of a given example). On the contrary; NeoDISCIPLE performs a local improvement of the KB (it improves only the object concepts and the inference rules that are involved in the considered explanation of the input, and does not attempt to find all the possible explanations). The price paid by EITHER for its more ambitious goals is a much simpler representation language (an extended propositional logic), algorithms that are very expensive computationally (practically not applicable for more complex KBs), and the simplifying assumption that the representation language is complete and the only problem is with the rules that are not correct. In NeoDISCIPLE we have taken the position that the problem of KB improvement is too complex to be automatically solved. Therefore the expert should be involved.

## X. FUTURE RESEARCH DIRECTIONS

We have presented an approach to the automation of building expert systems in which knowledge acquisition is viewed as a process of extending, updating and improving an incomplete and partially incorrect knowledge base. The main claim of our approach is that the system will start with a poor KB (i.e. with weak inferential capabilities), provided by the user and, through further interactions with the user, will evolves it to an incomplete KB (i.e. with incomplete inferential capabilities), and then to a complete KB (i.e. with complete inferential capabilities).

The presented methodology divides the process of building an expert system into three phases: a) providing a preliminary KB; b) incrementally extending and improving the KB; and c) reorganizing the KB.

The present version of NeoDISCIPLE addresses mainly the second phase. Therefore, a promising research direction is to evolve NeoDISCIPLE into a system that will greatly assist an expert user during all the three phases.

For the automation of the first phase, one may incorporate into NeoDISCIPLE techniques for systematic elicitation of expert knowledge as, for instance, the repertory greed technique used by ETS and ACQUINAS (Boose and Bradshaw, 1988; Shaw and Gaines, 1987).

Also the problem solving engine of NeoDISCIPLE has quite limited capabilities because the main focus of this research was not problem solving but learning. However, the learning method of NeoDISCIPLE is fairly general. Consequently, an interesting direction of development is to identify expert system shells the knowledge base of which could be learned by NeoDISCIPLE, and to couple each of them with a customized version of NeoDISCIPLE. Such an expert system shell is, for instance KEE (IntelliCorp, 1988).

The definition of the preliminary KB could also be automated by using an approach similar to that of the BLIP and MOBAL systems (Morik, 1989; Wrobel, 1989). These systems are able to build such an initial KB from user provided facts, generalization hierarchies, and general knowledge about the structure of the inference rules.

As mentioned, the main focus of our research was the incremental extension and improvement of the KB. Although the presented learning methods are quite powerful and general, there are many ways in which they can be improved. For instance, they could be enhanced by integrating new learning strategies, as explored in (Tecuci and Michalski, 1991a,b; Tecuci, 1991c). This is also related to the extension of the representation language of the system that currently allows only the representation of objects and of strict or plausible (i.e. incompletely learned) deductive rules. If, in the phase of defining a preliminary knowledge base, the human expert is to be allowed to introduce into the KB whatever knowledge he is able to express easily, then NeoDISCIPLE should also allow him to define cases (Bareiss, Porter and Murray, 1989), determinations (Davies and Russel, 1987), dependencies (Collins and Michalski, 1989), and other forms of knowledge.

Consequently, the learning methods of NeoDISCIPLE should be enhanced to deal with such new forms of knowledge, both in terms of using and improving them.

Concerning the reorganization of the KB, the explanation-based learning method described briefly in section VI provides only a potential for performance improvement. This method should be developed by providing a solution to the utility problem (Minton, 1990).

## ACKNOWLEDGMENTS

The author is grateful to Yves Kodratoff, Ryszard Michalski, Jianping Zhang, and Mike Hieb for useful comments and criticisms. This research was done in the Artificial Intelligence Center of George Mason University and was supported in part by a grant from the National Science Foundation, in part by the Office of Naval Research under grant No. N00014-91-J-1351, and in part by the Defense Advanced Research Projects Agency under grant No. N00014-91-J-1854, administered by the Office of Naval Research.

## REFERENCES

Bareiss, E.R., Porter B.W., and Murray K.S., Supporting Start-to-Finish Development of Knowledge Bases, in *Machine Learning, 4*, 259-283, 1989.

Bergadano F., and Giordana A., Guiding Induction with Domain Theories, in Kodratoff Y., and Michalski R.S. (eds), *Machine Learning: An Artificial Intelligence Approach,* vol. III, Morgan Kaufmann, 1990.

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.

Chandrasekaran, B., Towards a Taxonomy of Problem Solving Types, *AI Magazine*, 4(1):9-17, 1983.

Chandrasekaran, B., Generic Tasks in Knowledge-based Reasoning: High-level Building Blocks for Expert Systems Design, *IEEE Expert,* 1 (3):23-29, 1986.

Clancey, W., Classification Problem Solving, in *Proceedings of the Third National Conference on Artificial Intelligence,* Austin, Texas, 1984.

Collins, A., and Michalski, R.S., The Logic of Plausible Reasoning: A Core Theory, *Cognitive Science*, Vol. 13, No. 1, pp.1-49, 1989.

Cooper T. and Wogrin N., *Rule-based Programming with OPS5,* Morgan Kaufmann, 1988.

Danyluk, A.P., The Use of Explanations for Similarity-Based Learning, *Proceedings of IJCAI-87*, pp. 274-276, Milan, Italy, 1987.

Davies T.R. and Russell S.J., A Logical Approach to Reasoning by Analogy, *Proceedings of IJCAI-87*, pp. 264-270, Milan, Italy, 1987.

DeJong G., and Mooney R., Explanation-Based Learning: An Alternative View, in *Machine Learning,* vol.1, no. 2, pp. 145-176, 1986.

De Raedt L. and Bruynooghe M., CLINT: A Multistrategy Interactive Concept Learner and Theory Revision System, in Michalski R.S. and Tecuci G. (eds), *Multistrategy Learning: Proceedings of the First International Workshop,* Harpers Ferry, November, 1991.

Flann, N., and Dietterich, T., A Study of Explanation-based Methods for Inductive Learning, *Machine Learning* 4, 1989.

Gammack, J.G., Different Techniques and Different Aspects on Declarative Knowledge, in A.L. Kidd (ed), Knowledge Acquisition for Expert Systems: A Practical Handbook, Plenum Press, 1987.

Genest, J., Matwin, S., and Plante, B., Explanation-based Learning with Incomplete Theories: A Three-step Approach, in B.W.Porter and R.J.Mooney (eds), *Machine Learning: Proc. of the Eighth International Workshop,* Texas, Morgan Kaufmann, 1990.

Hirsh, H., Incremental Version-space Merging: A General Framework for Concept Learning, *Doctoral dissertation,* Stanford University, 1989.

IntelliCorp, KEE, User's Guide, Publication number K3.1-IRM-1, 1988.

Klinker G., KNACK: Sample-Driven Knowledge Acquisition for Reporting Systems, in S. Marcus (ed), *Automating Knowledge Acquisition for Expert Systems*, Kluwer Publishers, 1988.

Kodratoff, Y., and Michalski, R.S. (eds), *Machine Learning: An Artificial Intelligence Approach,* Morgan Kaufmann, vol.III, 1990.

Kodratoff Y., Rouveirol C., Tecuci G. and Duval B., Symbolic Approaches to Uncertainty, in Ras Z.W. and Zemankova M. (eds), Intelligent Systems: State of the Art and Future Directions, Ellis Horwood, 1990.

Lebowitz, M., Integrated Learning: Controlling Explanation, *Cognitive Science,* Vol. 10, No. 2, pp. 219-240, 1986.

Marcus S.(ed), *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers, 1988.

McDermott J., Preliminary Steps Toward a Taxonomy of Problem Solving Methods, in S. Marcus (ed), *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Pub., 1988.

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

Michalski R. S., Toward a Unified Theory of Learning: Multistrategy Task-adaptive Learning, Submitted for publication in *Machine Learning Journal*, 1990.

Michalski R.S. and Tecuci G. (eds), *Multistrategy Learning: Proceedings of the First International Workshop,* Harpers Ferry, West Virginia, November 1991.

Minton, S., Quantitative Results Concerning the Utility of Explanation-Based Learning, in *Artificial Intelligence,* 42, pp. 363-392, 1990.

Minton, S., Carbonell, J.G., Strategies for learning search control rules: an explanation-based approach, *Proc. IJCAI-87*, Milan, Italy, 1987.

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

Mitchell T.M., Keller R.M., and Kedar-Cabelli S.T., Explanation-Based Generalization: A Unifying View, *Machine Learning,* vol.1, no.1, pp. 47-80, 1986.

Mitchell T.M., Mahadevan S. and Steinberg L.I., LEAP: A Learning Apprentice System for VLSI Design, in *Proc. IJCAI-85,* Los Angeles, Morgan Kaufmann, 1985.

Mooney R., Bennet S., A Domain Independent Explanation Based Generalizer, in *Proceedings AAAI-86,* Philadelphia, 1986, pp.551-555.

Mooney R. and Ourston D., A Multistrategy Approach to Theory Refinement, in Michalski R.S. and Tecuci G. (eds), *Multistrategy Learning: Proceedings of the First International Workshop,* Harpers Ferry, November, 1991.

Morik K., Sloppy Modeling, in Morik K. (ed), *Knowledge Representation and Organization in Machine Learning*, Springer Verlag, Berlin 1989.

Pazzani M.J., Integrating Explanation-based and Empirical Learning Methods in OCCAM, in Sleeman D.(ed), *Proc. of the Third European Working Session on Learning,* Glasgow, 1988.

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.

Shavlik J.W., and Dietterich T., *Readings in Machine Learning,* Morgan Kaufmann, 1990.

Shavlik J.W., & Towell G.G., An Approach to Combining Explanation-based and Neural Learning Algorithms, *Connection Science*, Vol.1, No.3, 1989.

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, 1991a.

Tecuci G., Steps Toward Automating Knowledge Acquisition for Expert Systems, in *Proceedings of the AAAI-91 Workshop on Knowledge Acquisition: From Science to Technology to Tools,* Anaheim, California, 1991b.

Tecuci G., Learning as Understanding the External World, in R.S.Michalski and G. Tecuci (eds), *Proceedings of the First International Workshop on Multistrategy Learning,* Harpers Ferry, West Virginia, November 1991c.

Tecuci G., and Michalski R.S., A Method for Multistrategy Task-adaptive Learning Based on Plausible Justifications, in Birnbaum L., and Collins G. (eds), *Machine Learning: Proceedings of the Eighth International Workshop,* Chicago, June 1991, Morgan Kaufmann, 1991a.

Tecuci G., and Michalski R.S., Input "Understanding" as a Basis for Multistrategy Task-adaptive Learning, in *Proceedings of the International Symposium on Methodologies for Intelligent Systems, Charlotte,* North-Carolina, October 1991, Lecture Notes in Artificial Intelligence, Springer-Verlag, 1991b.

van Melle, W., Scott, A.C., Bennett, J.S., and Peairs, M., The EMYCIN Manual, *Report no. HPP-81-16,* Computer Science Department, Stanford University, 1981.

Whitehall, B.L., Knowledge-based Learning: Integration of Deductive and Inductive Leaning for Knowledge Base Completion, *PhD Thesis,* Report No. UIUCDCS-R-90-1637, Department of Computer Science, University of Illinois at Champaign-Urbana, 1990.

Wilkins, D.C., Knowledge Base Refinement as Improving an Incorrect and Incomplete Domain Theory, in Kodratoff Y., and Michalski R.S. (eds), *Machine Learning: An Artificial Intelligence Approach,* vol. III, Morgan Kaufmann, 1990.

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