Automating Knowledge Acquisition as Extending, Updating, and Improving a Knowledge Base

Gheorghe D. Tecuci

Abstract— **A method for the automation of knowledge acquisition that is viewed as a process of incremental extension, updating, and improvement of an incomplete and possibly partially incorrect knowledge base of an expert system is presented. The knowledge base is an approximate representation of objects and inference processes in the expertise domain. Its gradual development is guided by the general goal of improving this representation to consistently integrate new input information received from the human expert (as, for instance, new facts or examples of problem solving episodes). The knowledge acquisition method is presented as part of a methodology for the automation of the entire process of building expert systems, and is implemented in the system NeoDISCIPLE. The method promotes several general ideas for the automation of knowledge acquisition as, for instance, understanding-based knowledge extension, knowledge acquisition through multistrategy learning, consistency-driven concept formation and refinement, closed-loop learning, and synergistic cooperation between a human expert and a learning system.**

I. Introduction

AUTOMATING the process of building expert systems is one of the major goals of artificial intelligence [4]. 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 has 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 consisting of an inference engine for a class of tasks, and supporting 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. Expert system shells can be characterized by the generality of their inference engine. The range of such systems contains very general shells, like OPS [9] and KEE [18], general shells for a certain type of expertise task like, for instance, diagnosis in the case of EMYCIN [48], and even quite specific shells for role-limiting problem solving methods as, for instance, KNACK [19] and SALT [23]. 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. 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 played by required task knowledge and the form in which that knowledge can be represented [24]. Research on the identification of problem solving methods for generic tasks [5]–[7] 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 [3], [15], [38]. 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 [1], [41], [50]. During reformulation, the knowledge base is reorganized and/or compiled to solve problems more efficiently [29], [33].

The above classification of expert system building tools into expert system shells and knowledge acquisition tools also reflects the traditional distinction made between problem solving and learning. However, as new and more powerful learning methods are developed, it becomes more and more clear that learning and problem solving share many common processes. In fact, for learning methods like explanation-based learning, abductive learning, or learning by analogy [20], [39] problem solving is often part of learning. Consequently, building systems that have both learning and problem solving

Manuscript received September 29, 1991; February 21, 1992. This research was done in the Artificial Intelligence Center of George Mason University and was supported in part by the National Science Foundation under grant IRI-9020266, in part by the Office of Naval Research under grant N00014-91-J-1351, and in part by the Defense Advanced Research Projects Agency under grant N00014-91-J-1854, administered by the Office of Naval Research.

The author is with the Artificial Intelligence Center, Department of Computer Science, George Mason University, 4400 University Drive, Fairfax, VA 22030, and with the Research Institute for Informatics, Bucharest, Romania.

IEEE Log Number 9201431.

capabilities appears to be a very promising research direction. 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 illustrates several general ideas for the automation of knowledge acquisition as, for instance, 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. The goal of this research is to elaborate a general framework for the automation of knowledge acquisition.

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 ideas on which NeoDISCIPLE is based. Then, Sections III–VII present the knowledge refinement method of NeoDISCIPLE, illustrating it with an example (building a question-answering system in geography). Section VIII presents two other applications of NeoDISCIPLE and discusses some of its general features. Section X relates NeoDISCIPLE to other approaches, and the last section outlines some directions of the future research.

II. General Ideas Illustrated by Neodisciple

A. Hybrid Knowledge Representation

In NeoDISCIPLE 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 on the application domain. These rules 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 [42], 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 [43].

The advantage of a hybrid knowledge representation over a uniform one is that it allows a more natural representation of the diverse types of knowledge pieces characterizing a given expertise domain.

B. Expert System Building as a Three Phase Process

With NeoDISCIPLE, an expert system is built in three phases. In the first phase the human expert has to define a preliminary knowledge base (KB). There are two main goals of this phase: 1) to allow the human expert to introduce into the KB whatever knowledge pieces s/he may easily express; 2) to provide the system with some background knowledge that would support it in acquiring new knowledge. The result of this phase will be an incomplete and possibly partially incorrect 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.

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 problems to be solved.

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.

Although the boundaries between these phases cannot be very clear, it is useful to identify them because each is characterized by specific goals and techniques.

C. 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.[1] 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 Fig. 1, and is based on the "understanding" of the input[2] [46]. 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

[1] The input fact will be explicitly stored into the KB when it is completely new and cannot be related to the previous knowledge of the system. This should also happen when the cost of inferring the fact would be too high. However, this last case is not considered in the current version of NeoDISCIPLE.

[2] The understanding of the input (i.e., explaining the input to itself) is performed by the problem solver that is part of the learner.

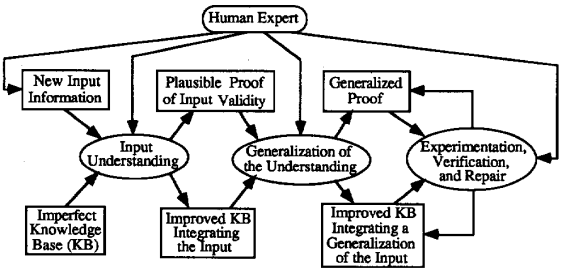


Fig. 1. Incremental development of the knowledge base.

correct them accordingly. As shown in Fig. 1, this process is supervised by the human expert.

The idea of understanding the input information in terms of the current knowledge of the system provides a natural scenario in which additional knowledge is acquired by means of pieces of explanations provided by the expert or generated by the system.

D. Knowledge Acquisition through Multistrategy Learning

Research in machine learning has investigated in detail several single-strategy learning methods [20], [26], [27], [39]. A striking feature of these methods is the complementary nature of their requirements and results. For instance, empirical induction requires many input examples and a small amount of background knowledge. Explanation-based learning requires one input example and a complete background knowledge. Learning by analogy and case-based learning require background knowledge analogous with the input. Learning by abduction requires causal background knowledge related to the input. The result of empirical induction is a hypothetical generalization of several input examples. The result of explanation-based learning is an operational generalization of an input example. The result of learning by analogy and of case-based learning is new knowledge about the input. The result of learning by abduction is new background knowledge. This complementary nature of the single-strategy learning methods naturally suggests that one could obtain a synergistic effect by properly integrating them. In such a multistrategy system different strategies could mutually support each other, and compensate for each other's weaknesses [28].

NeoDISCIPLE employs a multistrategy learning method. 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 generalization (to generalize the generated instances), conceptual clustering (to define new object concepts), and learning by instruction (to acquire new knowledge from the user).

E. Consistency-Driven Concept Formation and Refinement

The semantic network of object concepts provided by the human expert in the first phase of knowledge acquisition is an incomplete terminology for representing and learning new object concepts, facts, rules etc. Because of this incompleteness, the general knowledge pieces learned by NeoDISCIPLE may be inconsistent, covering also some negative examples (called false positive or negative exceptions). In order to uncover these negative examples, new object concepts may need to be introduced into the semantic network, or the definitions of the existing object concepts may need to be refined. For instance, one may uncover 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 [43], [51]. Alternatively, one may refine the definition of some object concept used in the condition of the rule with a new feature shared only by the positive examples of the rule [43]. In this way, the hierarchy of object concepts is iteratively developed with the goal of improving the consistency of the learned rules.

F. Synergistic Cooperation Between the Human Expert and the Learning System

The learning method of NeoDISCIPLE is based on the cooperation between a human expert and a learning system (see Fig. 1) exploiting their complementary abilities. In this cooperation, the human expert helps by solving problems that are intrinsically difficult for a learning system. Difficult problems for learning systems include the credit-blame assignment problem (i.e., assigning credit or blame to the individual decisions that led to some overall result), and the new terms problem (i.e., extending the representation language with new terms when it cannot represent the concept or rule to be learned) [43]. On the other hand, the learning system is responsible for the generation of general concepts or rules that account for specific examples, and for updating the KB so as to consistently integrate the learned knowledge.

G. Closed-Loop Learning

As shown in Fig. 1, the knowledge learned from an input becomes background knowledge that is used in the subsequent learning process, increasing the quality of learning. Therefore, NeoDISCIPLE illustrates a typical case of closed-loop learning.

III. Illustration of the Methodology for Building Expert Systems

A. A Question-Answering System in the Area of Geography

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.

The KB of the system contains explicit object knowledge, which we call basic object knowledge (in the form of a semantic network of object concepts), and implicit object knowledge (in the form of rules for inferring new object features from other object features).

A sample of the KB is presented in Fig. 2. The top part of the figure contains the semantic network of object concepts that describes the types of the geographical objects, together with their features (i.e., 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") that implies that any concept inherits all the features 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. For the same reason, some of the objects from the KB (Jamaica, mango, akee and grape) have not been included in the figure. One may notice that some concepts have several relationships with the same name as, for instance, "(Romania TERRAIN flat)" and "(Romania TERRAIN hill)." This means that the terrain of Romania includes both flat and hill regions. Similarly, "(rice NEEDS-CLIMATE tropical)" and "(rice NEEDS-CLIMATE subtropical)" means that the type of climate needed by "rice" includes "tropical" and "subtropical".

The semantic network may be incomplete in the sense that it may not contain all the relevant object names and features.

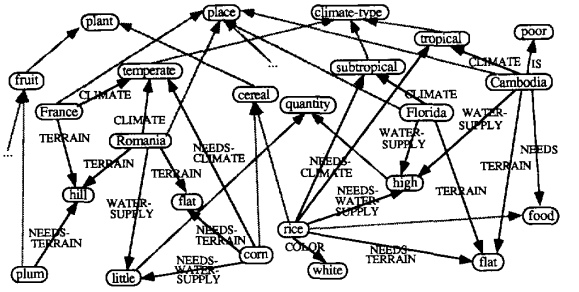
The bottom part of Fig. 2 contains three rules for inferring new properties and relationships of objects from other properties and relationships. The conditions of the inference rules are conjunctive expressions formed with the properties and relationships (as predicates) and the object names (as predicate arguments) from the hierarchical semantic network. One may notice that, while the rule $R1$ has an exact applicability condition, the rules $R2$ and $R3$ have 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 [31], [41] 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 (negative examples that are covered by the lower bound or positive examples that are not covered by the upper bound).

There are two main differences between the plausible version spaces used by NeoDISCIPLE and the standard version spaces defined by Mitchell [31].

First, in the case of Mitchell's version space method, the lower bound and the upper bound 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.

Secondly, in the case of Mitchell's method, both bounds may consist of an arbitrarily large number of conjunctive expressions, that may lead to a combinatorial explosion, due to the use of an exhaustive search method. On the contrary, the learning method of NeoDISCIPLE is based on a heuristic search in which each bound of the version space consists of only one conjunctive expression.

The rules in the NeoDISCIPLE's KB could therefore be incomplete and possibly partially incorrect.



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
plausible 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
plausible lower bound
(x IS-A fruit)&(y IS-A place)& ; the water supply of the place y
(t IS-A high)&(u IS-A tropical)& ; is high, as needed by the fruit x,
(y WATER-SUPPLY t)&(x NEEDS-WATER-SUPPLY t)& ; and the climate of y is tropical
(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<-mango,y<-Cambodia,t<-high,u<-tropical) & (x<-akee,y<-Jamaica,t<-high,u<-tropical)

R3:IF
plausible 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
plausible 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 x
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)

Fig. 2. A sample KB (the grey unlabelled arrows in the semantic network represent IS-A relationships).

One should also notice that the system keeps all the instances from which the rules have been learned. An instance I of a rule R is represented by the substitution σ, which transform R into I (i.e., $I = \sigma R$). These instances are the main source of knowledge for extending the representation language of the system with new concepts or concept features [43], as well as for accordingly updating the rules.

When applying an incompletely learned rule, if the lower bound condition 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.

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 Fig. 3.

B. Definition of the Preliminary Knowledge Base

As mentioned in Section II-B, the expert system is built in three phases. In the first phase the human expert has to define an initial KB, without any interaction with NeoDISCIPLE. S/he may define both an initial semantic network of object concepts, and a set of inference rules.

In the current version of NeoDISCIPLE it is assumed that the descriptions of the object concepts may be incomplete but correct. The inference rules, however, may be both incomplete and possibly partially incorrect.

The minimum knowledge that the human expert is required to provide consists of an incomplete hierarchical semantic network of objects. Providing initial rules is not a requirement because the system may learn such rules, as will be described in Section VI.

C. Knowledge Refinement

After the initial KB has been provided, NeoDISCIPLE extends, updates, and improves it 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.

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

The human expert has told the system that "(rice GROWS-IN Cambodia)," and NeoDISCIPLE has refined the knowledge base represented in Fig. 2 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)."

Third, it has improved the rules $R2$ and $R3$. Indeed, it has discovered two new positive examples of the rule $R2$ and has generalized the plausible lower bound so as to cover them (the generalized literals are underlined). It has also discovered two positive examples and one negative example of the rule $R3$. Consequently, it has specialized the two plausible bounds of $R3$ so as no longer to cover the negative example, and has generalized the plausible lower bound so as to cover the positive examples. One should notice that, as opposed to the standard version space method [31], where the lower bound can only be generalized, in this case the lower bound has been both generalized and specialized. It has been generalized by generalizing "(x IS-A fruit)&(z IS-A hill)" to "(x IS-A plant)&(z IS-A terrain-type)," and it has been specialized by adding the conjunction of literals "(v IS-A soil-type)&(y SOIL v)&(x NEEDS-SOIL v)." The specializations of the plausible bounds of $R3$ show also how the new relationships "SOIL" and "NEEDS-SOIL" extend the representation language of the system. What is not directly observable in Table I 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:

TABLE I
THE KNOWLEDGE REFINEMENT PROBLEM

Given
- *Input:* a new fact provided by the human expert (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 Fig. 2).

Determine
- *Refined knowledge base* entailing not only the received input but also similar ones (as, for instance, (corn GROWS-IN Romania)).

In the case of the input (rice GROWS-IN Cambodia) the KB has been refined by:

Adding new relevant geographical relationships: SOIL and NEEDS-SOIL

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

Improving the applicability conditions of the rules R2 and R3:

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<-mango, y<-Cambodia, t<-high, u<-tropical)
(x<-akee, y<-Jamaica, t<-high, u<-tropical)
(x<-rice, y<-Cambodia, t<-high, u<-tropical)
(x<-corn, y<-Romania, t<-little, u<-temperate)

R3: IF
plausible upper bound
(x IS-A something)&(y IS-A something)&(z IS-A something)&
(y TERRAIN z)&(x NEEDS-TERRAIN z)&
(y 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)
(x<-corn, y<-Romania, z<-flat, v<-normal-soil)
with the negative example
(x<-rice, y<-Florida, z<-flat)

- the KB contains explicit object knowledge (in the form of a hierarchical semantic network), and, possibly, 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 is presented in the next section and illustrated in the Sections V and VI. Section V shows how the system acquired the different knowledge pieces from Table I, that improved the KB in Fig. 2. Section VI illustrates the knowledge refinement method in the situation in which the KB contains only the semantic network of object concepts from the top of Fig. 2.

D. 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 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 Fig. 1 and detailed as follows (these steps will be illustrated in Sections V and VI).

Understand the input

1) Build a plausible proof 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 $F_p, \cdots, F_q$ from the KB that plausible imply I.
2) Introduce into the KB the facts abduced (if any) during the building of the tree T (the abductions made by the system are validated by the expert).
3) Let $R_i, \cdots, R_j$ 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 inference steps from 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 $R_i, \cdots, R_j$. The top of this tree will be the generalization I_g of the input I, and the leaves will be the generalizations $F_{pu}, \cdots, F_{qu}$, of the facts $F_p, \cdots, F_q$.
5) Build the most general deductive generalization T_l, of the tree T, by using the lower bound conditions of the rules $R_i, \cdots, R_j$. The top of this tree will be the generalization Ig of the input I, and the leaves will be the generalizations $F_{pl}, \cdots, F_{ql}$, of the facts $F_p, \cdots, F_q$.
6) Build a plausible version space VP, representing the inferential capabilities of the system with respect to inputs similar to I:

   IF

   plausible upper bound: $F_{pu}\& \cdots \& F_{qu}$

   plausible lower bound: $F_{pl}\& \cdots \& F_{ql}$

   THEN

   I_g

   While

   - the two bounds of the plausible 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 upper bound of VP that is not an instance of the lower bound. Let $(F_{px}\& \cdots \& F_{qx}) = \sigma(F_{pu}\& \cdots \& F_{qu})$, where σ is a substitution, be this instance.
8) Generate a fact similar to the input I, by applying the substitution σ to I_g: $\sigma(I_g)$
9) Generate the instance of the tree T_u, corresponding to the fact $\sigma(Ig)$, by applying σ to T_u : $\sigma(T_u)$.

Verify

10) Ask the expert if $\sigma(I_g)$ is true. If the answer is Yes then go to step. Otherwise go to 13.
11) The tree $\sigma(T_u)$ shows new positive instances of the rules $R_i, \cdots, R_j$. Generalize (if necessary) the plausible lower bounds of these rules, as little as possible, so as to cover the corresponding inference steps 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 $(F_{px}\& \cdots \& F_{qx})$, 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 a false inference step. Let this step be $A_{kx} \rightarrow C_{kx}$ (the blame assignment problem).
14) Let R_k be the rule in the KB the instance of which is $A_{kx} \rightarrow C_{kx}$. If the plausible lower bound of the rule R_k 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 A_{kx} and to remain more general then the plausible lower bound. Update the tree T_u by using the new upper bound of the rule R_k.
16) Specialize the upper bound of the plausible version space VP, as little as possible, so as no longer to cover $(F_{px}\& \cdots \& F_{qx})$ and to remain more general then the lower bound. Then go to 24.
17) Let $A_k \rightarrow C_k$ be the inference step from the tree T, corresponding to the false inference step $A_{kx} \rightarrow C_{kx}$. Ask the user to correct the inference step $A_k \rightarrow C_k$, by adding additional left hand side predicates. If the user indicates that the correct inference is $A_k\& B_k \rightarrow Ck$ then go to step 18. Otherwise, if the user cannot correct the inference step $A_k \rightarrow C_k$, go to 23.
18) Introduce B_k into the KB (the new terms problem: B_k may be a new term).
19) Let B_{ku} be the inductive generalization of B_k obtained by replacing each object in B_k with a variable. Let $B_{kx1}, \cdots, B_{kxn}$ be the instances of Bku 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 Bku. Specialize the upper and the lower bounds of Rk by conjunctively adding B_{ku} and B_{kl}, respectively.
20) Add $B_{kx1}, \cdots, B_{kxn}$ into the KB.
21) Update the tree T_u by using the new upper bound 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 $A_k \rightarrow C_k$ as a negative exception of the rule R_k (the exceptions are used by NeoDISCIPLE in the concept formation and refinement process, as described in [43].

24) Continue the while loop.

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

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. In such a case, the main result of learning is a new inference rule that derives the input. However, if no facts that imply the input could be found, then the input is explicitly stored into the KB. This case will be illustrated in Section VI.

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. This case will be illustrated in Section V.

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 any 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 is correct and complete, the knowledge refinement method behaves as a knowledge reformulation one, as will be shown in Section VII.

An ideal knowledge acquisition scenario is the following one. First the expert provides an initial semantic network (like the one from the top of Fig. 2). Next, the system improves the semantic network and learns rules (like the ones from the bottom of Fig. 2), by applying the techniques illustrated in Section VI. Next, the system improves the semantic network and the rules, by applying the techniques illustrated in Section V. Finally, it optimizes the knowledge base, as indicated in Section VII.

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 proof tree that 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 Fig. 2 and the input is "(rice GROWS-IN Cambodia)." In this case, the system builds the plausible proof tree from Fig. 3. We call this tree "plausible" because it was built by using the upper bound conditions of the rules $R2$ and $R3$. In order to simplify the figure, we have not included in the tree the leaves "(Cambodia IS-A something)," "(high IS-A something)," "(rice IS-A something)," "(tropical IS-A something)," and "(flat IS-A something)," which are obviously true facts.

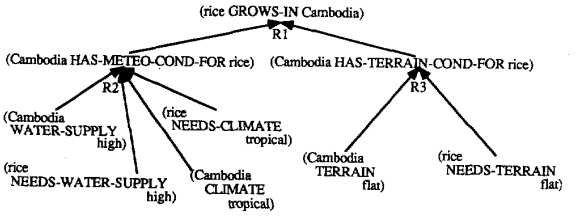


Fig. 3. A plausible proof of "(rice GROWS-IN Cambodia)."

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 [42]. For instance, it generalizes "fruit" to "plant" in the plausible lower bound of $R2$. This generalization is made by climbing the generalization hierarchy defined by the IS-A relationship, in the semantic network from Fig. 2. NeoDISCIPLE also keeps the instance of $R2$ from the tree in Fig. 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 high)&(u IS-A tropical)&
(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<-mango, y<-Cambodia, t<-high, u<-tropical)
(x<-akee, y<-Jamaica, t<-high, u<-tropical)
(x<-rice,y<-Cambodia,t<-high,u<-tropical)

Similarly, NeoDISCIPLE generalizes the lower bound of the version space of $R3$, so as to cover the instance of $R3$ from Fig. 3 (i.e., it generalizes "fruit" to "plant" and "hill" to "terrain-type"):

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)

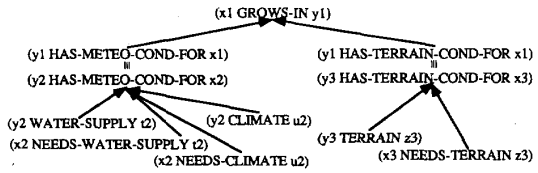


Fig. 4. The explanation structure of the input, corresponding to T_u.

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 $R3$, then 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 tree T in Fig. 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 [34]. 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 Fig. 4. In order to simplify the figure, we have removed the seven leaves of the form "(x IS-A something)." Although these literals do not impose any constraints on the corresponding variables, they are part of the tree.

Next NeoDISCIPLE determines the most general unification of the connection patterns:

($y1$ HAS-METEO-COND-FOR $x1$)
| | |
($y2$ HAS-METEO-COND-FOR $x2$) ($y2 = y1$, $x2 = x1$)

($y1$ HAS-TERRAIN-COND-FOR $x1$)
| | |
($y3$ HAS-TERRAIN-COND-FOR $x3$) ($y3 = y1$, $x3 = x1$)

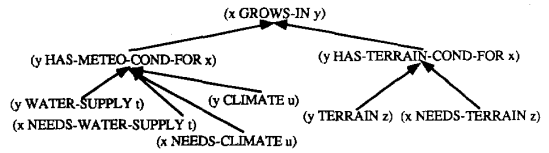


Fig. 5. The generalization T_u of the plausible proof tree in Fig. 3.

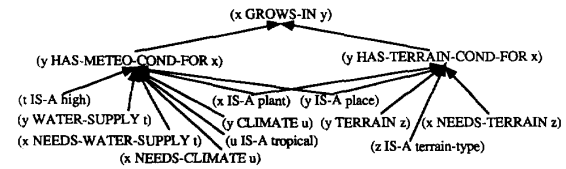


Fig. 6. The generalization T_l of the plausible proof tree in Fig. 3.

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 high) & (u IS-A tropical) &
(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)

Fig. 7. The plausible version space VP synthesizing the inferential capabilities of the system with respect to the facts of the form "(x GROWS-IN y)."[3]

Therefore: $x1 = x2 = x3 = x$, $y1 = y2 = y3 = y$.

By applying this unification to the explanation structure in Fig. 4 one builds the most general justified generalization of the tree in Fig. 3 (see Fig. 5).

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

The general proof tree T_u in Fig. 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 the leaves of the tree T_l (see Fig. 6). To synthesize these inferential capabilities, NeoDISCIPLE builds the plausible version space VP in Fig. 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:

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

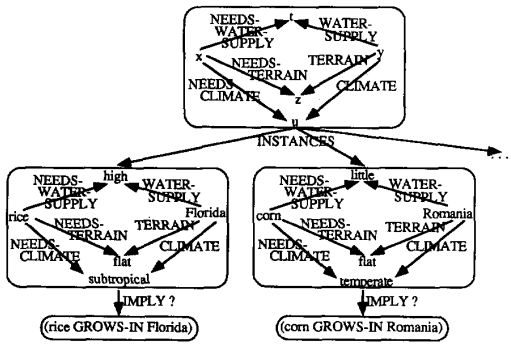


Fig. 8. Generation of facts similar with the input one.

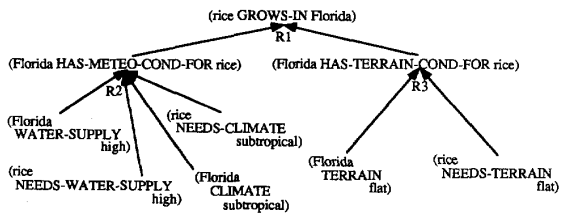


Fig. 9. Instance of the tree T_u corresponding to the generated fact "(rice GROWS-IN Florida)."

C. Experimentation[4]

The version space VP in Fig. 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 Fig. 8), asking if it is true or false. Then the system updates the KB such that the true facts are inferred, and the false facts are not.

The version space in Fig. 7 serves both for generating 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 accepted by the user (the positive examples), and the upper bound is specialized so as to no longer cover the generated facts rejected by the user (the negative examples). The learning process will end in one of the following situations: 1) the bounds of the version space in Fig. 7 become identical; 2) 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 fact of the form "(x GROWS-IN y)" can be generated.

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 Fig. 9).

D. Verification

The generated fact is shown to the user who is asked to confirm or to reject it:

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

[4]Experimentation in NeoDISCIPLE is a form of learning by analogy, as shown in [42].

Could you tell me why (system)
Not(Florida HAS-TERRAIN-COND-FOR rice)
in spite of the fact that
(Florida TERRAIN flat) & (rice NEEDS-TERRAIN flat) ?

The explanation is: (human expert)
(Florida SOIL normal-soil) & (rice NEEDS-SOIL fertile-soil)

Could you provide the corresponding piece of explanation for (system)
(Cambodia HAS-TERRAIN-COND-FOR rice) ?

The explanation is: (human expert)
(Cambodia SOIL fertile-soil) & (rice NEEDS-SOIL fertile-soil)

Fig. 10. Sample dialog between the expert and the system.

Because the KB plausible entails this false fact, it should be repaired, as shown in the following section.

E. Repair

The proof tree from Fig. 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 rule $R3$) is wrong:

(Florida TERRAIN flat)&(rice NEEDS-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 VP in Fig. 7, as little as possible, so as no longer to cover the leaves of the tree in Fig. 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 of VP. 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 this 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 VP in Fig. 7. To this purpose, the system asks the user to give an explanation of the above failure, as indicated in the sample dialog from Fig. 10.

The aforementioned sample dialog between the system and the human expert illustrates how, by asking for explanations, the system may extract many useful pieces of knowledge from the expert.

The explanations given by the expert introduce two new relevant geographical relationships, "SOIL" and "NEEDS-SOIL," as well as four new facts expressed with these new relationships, that are explicitly stored into the semantic network of object concepts.

The second explanation also shows that the inference step

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

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

(Cambodia TERRAIN flat)&(rice NEEDS-TERRAIN flat)&
(Cambodia SOIL fertile-soil)&(rice NEEDS-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 rule $R3$, that produced this inference, so as no longer to generate it.

First, the system inductively generalizes the additional condition

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

by replacing each object with a variable, thus obtaining

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

and conjunctively adds this expression to the upper bound of $R3$.

Then the system determines all the instances of the previous general expression, corresponding to the known positive examples 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 previous expressions, together with

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

correspond to the known positive examples of the rule $R3$. Therefore, NeoDISCIPLE determines a least general conjunctive generalization of them [42] 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)[5]

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

[5]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."

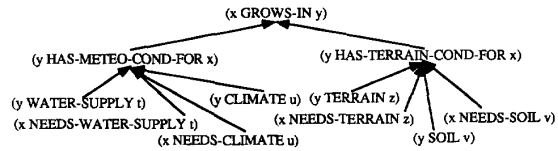


Fig. 11. The updated tree T_u.

Next, the tree T_u (which was built by using the upper bound condition of the rule $R3$) is updated to correspond to the new upper bound of $R3$. That is, "(v IS-A something)," "(y SOIL v)" and "(x NEEDS-SOIL v)" are added as leaf preconditions for inferring "(y HAS-TERRAIN-COND-FOR x)." The updated tree T_u is shown in Fig. 11. As mentioned previously, to simplify the figure, the literals of the form "(x IS-A something)" are not shown.

The same type of modifications that have been made to the version space of $R3$ are also made to the version space VP in Fig. 7. Consequently, the version space VP becomes:
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 high)&(u IS-A tropical)&
(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)

The first Experiment-Verify-Repair sequence is now completed and the system starts a new sequence.

F. Experimentation

In the previous Experiment-Verify-Repair sequence, the bounds of the version space VP have been specialized by conjunctively adding literals built with the newly defined relationships "SOIL" and "NEEDS-SOIL."

Because these relationships have been defined for very few objects, it is very likely that the system will no longer find instances of the plausible upper bound of VP, into the KB. Consequently, the knowledge acquisition process would end. To avoid such a situation, the system will try to elicit additional knowledge about the new relationships "SOIL" and "NEEDS-SOIL" for any objects that satisfy the other conditions of the plausible upper bound of VP, without satisfying the corresponding conditions of the plausible lower bound of VP. As shown in Fig. 8, such objects are "corn" and "Romania." Therefore, the system initiates the following dialog:

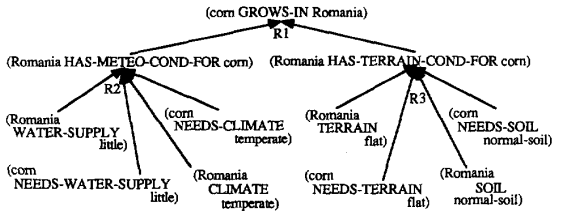


Fig. 12. Instance of the updated tree T_u corresponding to the fact "(corn GROWS-IN Romania)."

Could you provide a value for v such that
(Romania SOIL v)&(corn NEEDS-SOIL v)
[No / value] ? normal-soil

Because the user was able to provide a value for v, which satisfy the plausible upper bound condition of VP, the system generates the new fact "(corn GROWS-IN Romania)" and the corresponding instance of the updated tree T_u from Fig. 11 (see Fig. 12).

G. Verification

The generated fact (which is plausibly entailed by the current knowledge base) is shown to the user, which has to characterize it as true or false:

"Does (corn GROWS-IN Romania) ?" Yes.

Because the user confirmed this fact, NeoDISCIPLE treats the tree in Fig. 12 in the same way it treated the tree from Fig. 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, to cover these examples. In this case, however, only the lower bound of $R2$ needs to be generalized ("high" is generalized to "quantity" and "tropical" is generalized to "climate-type"). Also, the newly discovered positive examples are associated with the rules $R2$ and $R3$. Consequently, the new rules $R2$ and $R3$ are those from Table I.

The system also generalizes the plausible lower bound of the version space VP, to cover the leaves of the tree from Fig. 12, and therefore to "deductively" infer "(corn GROWS-IN Romania)." The new version space is shown in Fig. 13, with the generalized literals underlined.

With the KB from Fig. 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 Fig. 13, which is not an instance of the lower bound). One should notice, however, that the system has made all the knowledge refinement operations indicated in Table I.

It should also be noticed that the version space in Fig. 13 does not contain any new knowledge because whatever facts it can infer could also be inferred by applying the rules $R1$, $R2$, and $R3$. Therefore, the version space VP is not kept into the KB.

VI. Knowledge Refinement in the Case of Poor Knowledge

A. Understanding the Input

Let us now suppose that the current KB contains only the

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)

Fig. 13. Updated version space for the inference of "(x GROWS-IN y)."

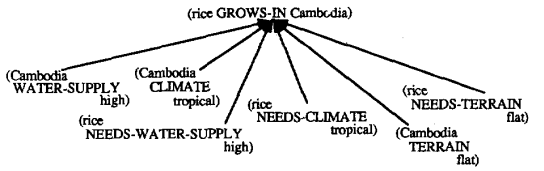


Fig. 14. Plausible proof of "(rice GROWS-IN Cambodia)."

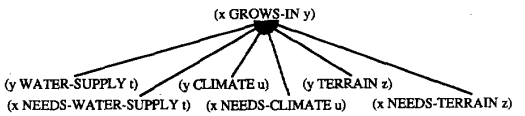


Fig. 15. Inductive generalization of the plausible proof in Fig. 14.

semantic network from the top of Fig. 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 proposes them as partial explanations of the input. The user has to select the true pieces of explanations and may indicate additional ones. One 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 [42]):

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 tropical)&(Cambodia CLIMATE tropical) ? 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 plausible proof in Fig. 14.

B. Generalizing the Understanding

NeoDISCIPLE inductively generalizes the plausible proof tree in Fig. 14 by simply turning all the constants into variables:

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 tropical) &
(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<-tropical)

Fig. 16. A plausible version space for a new inference rule.

The plausible proof tree from Fig. 14 and its inductive generalization from Fig. 15 define the version space VP from Fig. 16.

C. Experimentation

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

Also, the plausible version space from Fig. 16 is no longer a redundant knowledge piece. On the contrary, it represents an initial version space for a new inference rule to be learned by NeoDISCIPLE. Therefore, in the case of poor knowledge about an input I, the main result of learning is a rule for inferring I.

In this way 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 Fig. 2 has been augmented with the relationships from the top of Fig. 17, and the incompletely learned rules $R2$ and $R3$ from Fig. 2 have evolved to the rules $R2$ and $R3$ from Fig. 17.

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 [12], [32]. Indeed, NeoDISCIPLE builds a tree similar to the one from Fig. 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 Fig. 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, shown at the bottom of the page.

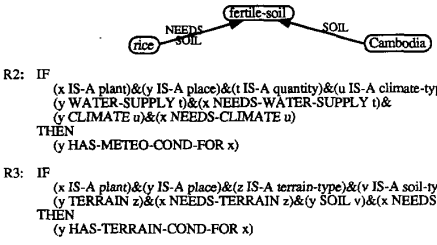


Fig. 17. Knowledge to be added to the one in Fig. 2 in order to transform it into complete knowledge with respect to the input "(rice GROWS-IN Cambodia)."

One should notice that, in such a case, the system does not learn any 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 Fig. 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 also has a negative effect on the efficiency of the rule interpreter that may need to test more rules in order to solve a problem. Therefore, the decision on whether to keep the learned rule should be based on its utility, that takes into account both its positive and negative effects. Initial results on the utility problem [29] suggest that the best performance is obtained when the system learns a small number of such rules that are sufficient for solving most of the problems. The utility problem in NeoDISCIPLE is a topic for future research.

VIII. Experiments

A version of NeoDISCIPLE is implemented in Common Lisp and runs on the 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

---

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 x
THEN ; then
(x GROWS-IN y) ; x grows in y

NeoDISCIPLE (developing a knowledge base for planing the manufacturing of a loudspeaker) is described in [42].

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 a robot. First the system was 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 ; and to take the cup from it
FROM cabinet

the system learned the rule

IF ; If
(x IS-A movable-obj) & ; x is a movable object
(y IS-A container) &
(x IS-IN y) & (y IS closed) ; in 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 examples that were covered by the learned rule [43].

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→H_2O+NaCl," the system learned that, in general, by combining a base with an acid, one obtains water and salt:

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

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

C. Discussion

There are several general questions that one may ask with respect to the KA method illustrated by NeoDISCIPLE. In the following we present some of these questions and give initial answers.

How general is the KA method ? Which are the suitable application domains ?

In NeoDISCIPLE, the basic source of knowledge for learning is the hierarchical semantic network that provides the generalization language. Therefore, the first requirement for a suitable application domain is to allow the definition of a rich semantic network.

A second requirement is to base the problem solving method on problem solving operators that can be learned by generalizing specific examples. The generality of these two requirements is a strong indication that the proposed KA method is potentially applicable in a wide range of domains.

It is also necessary to characterize the difficulty of the tasks that the expert is required to perform during the knowledge acquisition process. How much knowledge should the expert initially introduce into the KB? Should s/he also define rules?

As shown in Section VI, the minimum knowledge to be initially introduced into the KB is the semantic network of object concepts. This knowledge allows the system to learn rules from the new input information provided by the expert. Therefore, providing rules is not a requirement. On the other hand, there is no theoretical upper limit with respect to the knowledge provided by the expert, which may consist of both object descriptions and rules. The practical limitation comes from the fact that the provided semantic network is supposed to be incomplete but correct. During learning, the definition

of the object concepts may be refined and new concepts may even be defined [15]. However, no deletions are performed. It is a future research direction to develop the knowledge acquisition method to start with a semantic network that is not only incomplete, but also partially incorrect, and to gradually improve it.

How many questions are asked by the system during learning from an input? How difficult it is to answer them?

Based on the experiments performed, we could state that the number of questions asked is usually small (for instance, during a learning session, the system needed to generate less than 10 examples similar to the input). Also, the questions are, in general, very easy to answer. Many of them ask for an "yes" or "no" answer (e.g., asking if a generated example is positive or negative, if some expression is or is not an explanation of some failure; if an object has or does not have a certain feature; if an abduced fact is true or not; if an inference step is true or not). More difficult questions are those asking the expert to provide an explanation of some failure (when the system was not able to propose any) or to indicate the name of a concept covering specified instances [15]. However, even these questions are not very difficult to answer.

What is the relationship between the knowledge the user initially enters into the KB and the behavior of the system ?

NeoDISCIPLE has the capacity of adapting its learning behavior to the knowledge it has about the input, which could be characterized as poor, incomplete or complete. In the case of poor knowledge about the input, the system performs mainly inductive operations and relies heavily on the user to provide explanations of the input, without giving much assistance. In the case of incomplete knowledge, the system performs both deductive and inductive operations. Its questions are much more focussed and easier to answer. Finally, in the case of complete knowledge, the system does not need any help from the user. Consequently, there is a trade-off between how much knowledge the user initially enters into the KB and how much assistance s/he is required to give in the next stages of knowledge acquisition. Our hypothesis is that the optimal situation is obtained by requiring the expert to initially introduce into the KB as much knowledge s/he can easily express.

The proposed knowledge acquisition methodology is characterized by three phases: definition of the initial knowledge base, refinement of the knowledge base, and reformulation of the knowledge base. When do the transitions between these phases occur?

There are, in fact, no clear cut transitions between these stages. Knowledge base refinement should start immediately after the expert has provided the initial KB. However, if the initial knowledge provided by the expert is not enough for the understanding of a new input from the expert, this input is explicitly stored into the KB (as if provided in the initial KB). Also, knowledge reformulation should start when all the rules in the KB have been completely learned. Nothing, however, guarantees that the expert will not enter a piece of knowledge that is totally or partially new, causing the system to explicitly store it into the KB (as if provided in the first phase) or to learn a new rule (as if provided in the second phase).

We should finally stress that the above answers are based on experiments involving small knowledge bases. It is therefore necessary to test how the system would scale up, and to answer these questions in the context of much more complex applications.

IX. Related Research

NeoDISCIPLE is an extension and a generalization of DISCIPLE [41], [42]. 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., [2], [10], [13], [14], [16], [17], [22], [30], [35], [37], [40], [49], [50]. Our method is however different from other multistrategy learning methods in terms of the integrated strategies, the way they are integrated, and the interaction with the expert. More precisely, NeoDISCIPLE uses explanation-based learning (to build 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 generalization (to generalize the generated instances), conceptual clustering (to define new object concepts), and learning by instruction (to acquire new knowledge from the user). This multistrategy learning method exploits the complementarity of the requirements and results of the integrated single-strategy learning methods, and allows NeoDISCIPLE to adapt its behavior to the relationship between the input information and its knowledge.

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 [31] 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. Also, NeoDISCIPLE employs a heuristic search in the plausible version space, each

of the bounds consisting of a single conjunctive expression. Thus, NeoDISCIPLE avoids the problem of the exponential growth of these bounds.

The learning method of NeoDISCIPLE was mostly influenced by the explanation-based learning strategy [12], [32]. A significant merit of this learning strategy is the identification of the importance of the explanations in learning. NeoDISCIPLE extends this idea by using the explanations as the main source of expert knowledge. 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 with EBL, NeoDISCIPLE still requires a single input example to learn, because it is able to generate additional examples. This example generation capability is an important feature of NeoDISCIPLE that was originally introduced by DISCIPLE.

NeoDISCIPLE also extends constructive induction [25]. In constructive induction, the representation language of the learning system is extended with new terms that 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 [36], [51], NeoDISCIPLE introduces new terms in order to reduce the number of the exceptions of the learned rules. However, the methods employed by NeoDISCIPLE [43] are more flexible and adaptive to the current knowledge of the system, and also involve the expert in this process. As opposed to BLIP, which 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 [50] and EITHER [35]. ODYSSEUS automatically refines a KB by abducing facts that explain the actions of a human expert. NeoDISCIPLE may also use abduction to complete an explanation of the input, but it also attempts to generalize the explanation, so as to explain similar inputs. Thus, it learns more from an input than ODYSSEUS. However, it does this with the help of the expert.

Both systems deal with uncertain rules. ODYSSEUS, however, represents uncertainty through numeric certainty factors, while NeoDISCIPLE represents it symbolically [21], through plausible version spaces.

EITHER is an autonomous multistrategy learning system that uses deduction, abduction, and induction to improve a KB, so as to correctly classify a given set of positive and negative examples of some concept. NeoDISCIPLE is an interactive system that tries to improve a KB so as to produce the same answers as a human expert, which is a more general problem. However, EITHER globally improves the KB (for instance, by considering all the possible explanations of a given example), while NeoDISCIPLE performs a local improvement of the KB (it improves only the object concepts and the inference rules that are involved in an explanation of the input, and does not attempt to find all the possible explanations). The price paid by EITHER for its more ambitious goal is a simpler representation language (an extended propositional logic), algorithms that are very expensive computationally, and the simplifying assumptions that the representation language is complete and the only problem is with incorrect rules. 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.

NeoDISCIPLE illustrates a general methodology for the automation of knowledge acquisition, which is applicable to problem solving methods employing operators that may be learned by generalizing specific examples. On the other hand, the problem solving engine of NeoDISCIPLE is a general-purpose one and has limited capabilities. Therefore, NeoDISCIPLE is complementary to KA systems based on role-limiting methods as, for instance, SALT [23], which provides a specialized problem solving method and helps the expert to express his/her knowledge in the format required by the method. However, SALT could hardly be regarded as more than a powerful knowledge elicitation system, because it has no learning capabilities. This complementarity suggests however to develop customized versions of NeoDISCIPLE, in which its general knowledge acquisition capabilities would be associated with a specialized problem solving method. An indication that this is a promising research direction is the system KNACK [19] that employs a role-limited method enhanced with several generalization capabilities.

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). Several future research directions have already been mentioned in the previous two sections, in connection with some of the weaknesses of our approach. Others are presented in the following.

The proposed KA methodology divides the process of building an expert system into three phases: 1) providing a preliminary KB; 2) incrementally extending and improving the KB; and 3) 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 assist an expert user during all the three phases. For the automation of the first phase, one may incorporate techniques for systematic elicitation of expert knowledge into NeoDISCIPLE as, for instance, the repertory grid technique used by ETS [38] and

ACQUINAS [3]. One could also use an approach similar to that of the BLIP and MOBAL systems [36], [51]. 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.

Also, the problem solving engine of NeoDISCIPLE has quite limited capabilities because the main focus of this research was not problem solving but learning. Consequently, a necessary research direction is to develop the problem solving capabilities of the system and, possibly, to develop customized versions of NeoDISCIPLE for specialized problem solving methods, as mentioned in Section IX.

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 [45]–[47]. 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 [1], determinations [11], dependencies [8], 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 [29].

ACKNOWLEDGMENT

The author is grateful to Mike Hieb, Yves Kodratoff, Ryszard Michalski, Jianping Zhang, and the anonymous reviewers for useful comments and criticisms.

REFERENCES

[1] E. R. Bareiss, B. W. Porter, and K. S. Murray, "Supporting start-to-finish development of knowledge bases," in *Machine Learning*, vol. 4, pp. 259–283, 1989.

[2] F. Bergadano and A. Giordana, "Guiding induction with domain theories," in *Machine Learning: An Artificial Intelligence Approach*, vol. 3., Y. Kodratoff and R. S. Michalski, Eds. San Mateo, CA: Morgan Kaufman, 1990, pp. 474–492.

[3] J. H. Boose and J. M. Bradshaw, "Expertise transfer and complex problems: Using AQUINAS as a Knowledge-acquisition workbench for knowledge-based systems," in *Knowledge Acquisition Tools for Expert Systems*, J. Boose and B. Gaines, Eds. New York: Academic, 1988, pp. 39–64.

[4] B. Buchanan and D. Wilkins, Eds., *Readings in Knowledge Acquisition and Learning: Automating the Construction and the Improvement of Programs*. San Mateo, CA: Morgan Kaufman, 1992.

[5] B. Chandrasekaran, "Toward a taxonomy of problem solving types," *AI Mag.*, vol. 4, pp. 9–17, 1983.

[6] ——, "Generic tasks in knowledge-based reasoning: high-level building blocks for expert systems design," *IEEE Expert*, vol. 1, pp. 23–29, 1986.

[7] W. Clancey, Classification Problem Solving, in *Proc. Third National Conf. Artificial Intell. (AAAI-84)*, Austin, TX, 1984, pp. 49–55.

[8] A. Collins and R. S. Michalski, "The logic of plausible reasoning: A core theory," *Cognitive Sci.*, vol. 13, pp. 1–49, 1989.

[9] T. Cooper and N. Wogrin, *Rule-based Programming with OPS5*. San Mateo, CA: Morgan Kaufman, 1988.

[10] A. P. Danyluk, "The use of explanations for similarity-based learning," in *Proc. Int. Joint Conf. Artificial Intell. (IJCAI-87)*, Milan, Italy, 1987, pp. 274–276.

[11] T. R. Davies and S. J. Russell, "A logical approach to reasoning by analogy," in *Proc. Int. Joint Conf. Artificial Intell. (IJCAI-87)*, Milan, Italy, 1987, pp. 264–270.

[12] G. DeJong and R. Mooney, *Explanation-Based Learning: An Alternative View, in Machine Learning*, vol. 1, pp. 145–176, 1986.

[13] L. De Raedt and M. Bruynooghe, "CLINT: A multistrategy interactive concept learner and theory revision system," in *Proc. First Int. Workshop on Multistrategy Learning*, R. S. Michalski and G. Tecuci, Eds., Harpers Ferry, WV, Nov. 1991, pp. 175–190.

[14] N. Flann and T. Dietterich, "A study of explanation-based methods for inductive learning," *Machine Learning*, vol. 4, pp. 187–266, 1989.

[15] J. G. Gammack, "Different techniques and different aspects on declarative knowledge," in *Knowledge Acquisition for Expert Systems: A Practical Handbook*, A. L. Kidd, Ed. New York: Plenum, 1987, pp. 137–163.

[16] J. Genest, S. Matwin, and B. Plante, "Explanation-based learning with incomplete theories: A three-step approach," in B. W. Porter and R. J. Mooney, Eds., in *Machine Learning: Proc. Eighth Int. Workshop*. San Mateo, CA: Morgan Kaufman, 1990.

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

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

[19] G. Klinker, "KNACK: Sample-driven knowledge acquisition for reporting systems," in S. Marcus, Ed., *Automating Knowledge Acquisition for Expert Systems*. Boston: Kluwer, 1988, pp. 125–174.

[20] Y. Kodratoff and R. S. Michalski, Eds., *Machine Learning: An Artificial Intelligence Approach*. San Mateo, CA: Morgan Kaufman, vol. 3, 1990.

[21] Y. Kodratoff, C. Rouveirol, G. Tecuci, and B. Duval, "Symbolic approaches to uncertainty," in *Intelligent Systems: State of the Art and Future Directions*, Z. W. Ras and M. Zemankova, Eds. New York: Ellis Horwood, 1990, pp. 259–291.

[22] M. Lebowitz, "Integrated learning: controlling explanation," *Cognitive Sci.*, vol. 10, pp. 219–240, 1986.

[23] S. Marcus, "SALT: A knowledge-acquisition tool for propose-and-revise systems," in S. Marcus, Ed., *Automating Knowledge Acquisition for Expert Systems*. Boston: Kluwer, 1988, pp. 81–123.

[24] J. McDermott, "Preliminary steps toward a taxonomy of problem solving methods," in S. Marcus, Ed., *Automating Knowledge Acquisition for Expert Systems*. Boston: Kluwer, 1988, pp. 225–266.

[25] R. S. Michalski, "Theory and methodology of inductive learning," in *Machine Learning: An Artificial Intelligence Approach*, vol. 1, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Palo Alto, CA: Tioga, 1983, pp. 83–134.

[26] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach*, vol. I, Palo Alto, CA: Tioga, 1983.

[27] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach*, vol. II. San Mateo, CA: Morgan Kaufman, 1986.

[28] R. S. Michalski and G. Tecuci, Eds., in *Proc. First Int. Workshop on Multistrategy Learning*, Harpers Ferry, WV, Nov. 1991.

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

[30] S. Minton and J. G. Carbonell, "Strategies for learning search control rules: An explanation-based approach," in *Proc. Int. Joint Conf. Artificial Intell. (IJCAI-87)*, Milan, Italy, 1987, pp. 228–235.

[31] T. M. Mitchell, "Version spaces: An approach to concept learning," Doctoral dissertation, Stanford Univ., 1978.

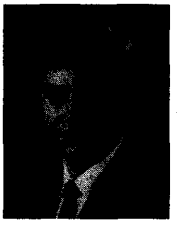
[32] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli, "Explanation-based generalization: A unifying view," *Machine Learning*, vol. 1, pp. 47–80, 1986.

[33] T. M. Mitchell, S. Mahadevan, and L. I. Steinberg, "LEAP: A learning apprentice system for VLSI design," in *Proc. Int. Joint Conf. Artificial Intell. (IJCAI-85)*, Los Angeles. San Mateo, CA: Morgan Kaufman, 1985, pp. 573–580.

[34] R. Mooney and S. Bennet, "A domain independent explanation based generalizer," in *Proc. Fourth Nat. Conf. Artificial Intell. (AAAI-86)*, Philadelphia, 1986, pp. 551–555.

[35] R. Mooney and D. Ourston, "A multistrategy approach to theory refinement," in *Proc. First Int. Workshop on Multistrategy Learning*,

R. S. Michalski and Tecuci G., Eds., Harpers Ferry, WV, 1991, pp. 115–130.
[36] K. Morik, Sloppy Modeling, in K. Morik, Ed., *Knowledge Representation and Organization in Machine Learning*. Berlin: Springer Verlag, 1989, pp. 107–134.
[37] M. J. Pazzani, "Learning causal relationships: An integration of empirical and explanation-based learning methods," Ph.D. dissertation, Univ. Calif., Los Angeles, 1988.
[38] M. L. G. Shaw and B. R. Gaines, "An interactive knowledge elicitation technique using personal construct technology," in A. L. Kidd, Ed., *Knowledge Acquisition for Expert Systems: A Practical Handbook*. New York: Plenum, 1987, pp. 109–136.
[39] J. W. Shavlik and T. Dietterich, Eds., *Readings in Machine Learning*. San Mateo, CA: Morgan Kaufman, 1990.
[40] J. W. Shavlik and G. G. Towell, "An approach to combining explanation-based and neural learning algorithms," in *Readings in Machine Learning*, J. W. Shavlik and T. Dietterich, Eds. San Mateo, CA: Morgan Kaufman, pp. 828–839, 1990.
[41] G. Tecuci, "DISCIPLE: A theory, methodology, and system for learning expert knowledge," Ph.D. dissertation, Univ. Paris-Sud, 1988.
[42] G. Tecuci and Y. Kodratoff, "Apprenticeship learning in imperfect theory domains," in *Machine Learning: An Artificial Intelligence Approach*, Y. Kodratoff and R. S. Michalski, Eds., vol. 3. San Mateo, CA: Morgan Kaufman, 1990, pp. 514–551.
[43] G. Tecuci, "A multistrategy learning approach to domain modeling and knowledge acquisition," in *Machine Learning—EWSL-91*, Y. Kodratoff, Ed. Berlin: Springer-Verlag, 1991.
[44] ——, "Steps toward automating knowledge acquisition for expert systems," in *Proc. AAAI-91 Workshop on Knowledge Acquisition: From Science to Technology to Tools*, Anaheim, CA, 1991, pp. 115–126.
[45] G. Tecuci and R. S. Michalski, "A method for multistrategy task-adaptive learning based on plausible justifications," in *Machine Learning: Proc. Eighth Int. Workshop, Chicago*, L. Birnbaum and G. Collins, Eds. San Mateo, CA: Morgan Kaufman, 1991, pp. 549–553.
[46] G. Tecuci and R. S. Michalski, "Input 'understanding' as a basis for multistrategy task-adaptive learning," in *Proc. Int. Symp. Methodologies for Intelligent Syst.*, Charlotte, NC, Oct. 1991, Lecture Notes in Artificial Intelligence, Springer-Verlag, pp. 419–428, 1991.
[47] G. Tecuci, "Plausible justification trees: A framework for deep and dynamic integration of learning strategies," *Machine Learning J.*, Special Issue on Multistrategy Learning, 1993.
[48] W. van Melle, A. C. Scott, J. S. Bennett, and M. Peairs, "The EMYCIN manual," rep. no. HPP-81–16, Comput. Sci. Dept., Stanford Univ., 1981.
[49] B. L. Whitehall, "Knowledge-based learning: Integration of deductive and inductive leaning for knowledge base completion," Ph.D. dissertation, Rep. no. UIUCDCS-R-90–1637, Dept. Comput. Sci., Univ. Ill., Champaign-Urbana, 1990.
[50] D. C. Wilkins, "Knowledge base refinement as improving an incorrect and incomplete domain theory," in *Machine Learning: An Artificial Intelligence Approach*, Y. Kodratoff and R. S. Michalski, Eds., vol. 3. San Mateo, CA: Morgan Kaufman, 1990, pp. 493–513.
[51] S. Wrobel, "Demand-driven concept formation," in *Knowledge Representation and Organization in Machine Learning*, K. Morik, Ed. Berlin: Springer Verlag, 1989, pp. 289–319.



Gheorghe D. Tecuci was born in Bucharest, Romania, in 1954. He received the M.S. degree in computer science from the Polytechnic Institute of Bucharest in 1979. He received two Ph.D. degrees in computer science, one from the University of Paris South, Paris, France in July 1988, and the other from the Polytechnic Institute of Bucharest, Romania in December 1988.

In 1979 he joined the Research Institute for Informatics in Bucharest as researcher and research project director. During the summers of 1986–1990 he worked at the Laboratory of Research in Computer Science, University of Paris South, as a Research Director of a joint research program on multistrategy learning, sponsored by the Romanian Academy and the French National Research Center (CNRS). In Spring 1990 he was a Visiting Scientist at the Center for Artificial Intelligence of the George Mason University and in Fall 1990 he joined the Faculty of the Computer Science Department as an Assistant Professor, and became Associate Professor in 1992. He has published more than 50 papers in machine learning, knowledge acquisition, expert systems, advanced robotics, compiler generation, and graph theory. He co-edited the book *Machine Learning: A Multistrategy Approach*, Vol. IV. His current research focuses on multistrategy learning, knowledge acquisition, and expert systems with learning capabilities. In 1987 he received the prize Traian Vuia of the Romanian Academy, and in 1991 was elected Corresponding Member of the Romanian Academy. He was Program Chairman and Co-organizer of the First International Workshop on Multistrategy Learning, in 1991, and served as chair program committee member, organizer, or referee to several conferences in artificial intelligence.