93-15

93-15

Proceedings of the Second International Workshop on

MULTISTRATEGY LEARNING (MSL-93)

May 26-29, 1993
Harpers Ferry

Edited by Ryszard S. Michalski and Gheorghe Tecuci

Sponsored by the Office of Naval Research and the Center for Artificial Intelligence, George Mason University

Contents

Knowledge Base Refinement Through a Supervised Validation of Plausible Reasoning

Gheorghe Tecuci
AI Center, CS Department
George Mason University
4400 Univ. Dr, Fairfax, VA 22030, USA
and Romanian Academy, Romania
tecuci@aic.gmu.edu

David Duff
AI Center
George Mason University
4400 Univ. Dr, Fairfax, VA 22030, USA
and The MITRE Corp. AI Technical Center
duff@mitre.org

Abstract

This paper presents an integrated heuristic approach to knowledge base refinement which is viewed as a supervised validation of plausible reasoning. The approach integrates multistrategy learning based on multitype inference, active experimentation, and guided knowledge elicitation. One of the main features of this approach is that once the knowledge base has been refined to deductively entail a new piece of knowledge, it can be easily further refined to deductively entail many other similar pieces of knowledge.

Keywords: multistrategy learning, knowledge acquisition, plausible reasoning

1. Introduction

An expert system consisting of an incomplete and partially incorrect knowledge base (KB), and of a deductive inference engine, suffers from two major limitations:

- it is not able to solve some problems from its domain of expertise (because the KB is incomplete);
- the solutions proposed might be incorrect (because the KB is partially incorrect).

The set of problems which such a system could solve is the *deductive closure* of the knowledge base (*DC*). In the case of a theorem prover, it is the set of facts which could be deductively inferred from the KB.

That is, $DC = \{ I : KB \models I \}$

where "$\models$" means deductive entailment.

Figure 1 shows the relationship between the deductive closure *DC* of the imperfect KB of an expert system and the set of true facts in the application domain (*TC*):

- *DC* and *TC* are "crisp" sets, with cleanly defined borders. That is, the system has an algorithm for testing the membership of a statement in *DC*, and presumably a human expert can perform a similar test on *TC*.
- $DC \cap TC$ represents the set of facts which are deductively entailed by the KB and are true. This shows that there is useful and correct knowledge encoded into the facts and the deductive rules of the KB.
- $DC - TC$ represents the set of facts which are deductively entailed by the KB but are false. This shows that there are errors in the set of facts and deductive rules.
- $TC - DC$ represents the set of facts which are true but are not deductively entailed by the KB. This shows that the set of facts and deductive rules is incomplete.

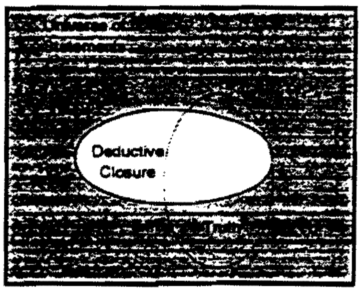

Figure 1: The relationship between *DC* and *TC*.

The goal of KB refinement is to improve the knowledge base so that DC becomes a good approximation of TC. As a result, the KB would become an almost complete and correct one, and the expert system would be able to correctly solve most of the problems from its domain of expertise.

Many of the current KB refinement systems such as ANA–EBL (Cohen, 1991), CLINT (De Raedt and Bruynooghe, 1993), DUCTOR (Cain, 1991), EITHER (Mooney and Ourston, 1993), FORTE (Richards and Mooney, 1993), SEEK (Ginsberg, Weiss, and Politakis, 1988), try to partially generalize the KB so as to cover more of TC, and to partially specialize it, so as cover less of $DC-TC$. In the case of a Prolog–like KB, this is accomplished by generalizing and/or specializing some of the rules, as well as by introducing new facts into the KB, and/or removing other facts.

In this paper, we are also addressing the problem of correcting and extending DC so as to better approximate TC. However, our approach, as opposed to the approaches cited above, brings a new set into play, the plausible closure of a KB, and proposes a different perspective to the KB refinement problem.

2. The plausible closure of the KB

We are assuming that the initial incomplete and partially incorrect KB consists of facts and rules expressed in first–order logic. However, the rules are not restricted to be deductive. They might also be weaker correlations as determinations (Davies and Russell, 1987; Russell, 1989), mutual dependencies (Michalski, 1993), etc. This is so for allowing the introduction of all sorts of relevant knowledge into the KB.

The *plausible closure* of the KB (PC) is defined as the set of problems which a *plausible inference engine* could solve. In the case of a theorem prover, it is the set of facts which could be plausibly inferred from the KB.

That is, $PC = \{ I : KB \models I \}$

where "$\models$" means plausible entailment.

One way to make plausible inferences is to use the rules from the KB not only deductively, but also abductively or analogically.

Let us consider, for instance, the rule

$$\forall_x [P(x) \rightarrow Q(x)].$$

If one knows that $P(a)$ is true, then one may *deductively* infer $Q(a)$:

$$\{P(a), \forall_x [P(x) \rightarrow Q(x)]\} \models Q(a)$$

If one knows that $Q(a)$ is true, then one may *abductively* (Pople, 1973; Josephson, 1991) infer $P(a)$:

$$\{Q(a), \forall_x [P(x) \rightarrow Q(x)]\} \models P(a)$$

If one knows that P(a) is true, and b is similar to a, then one may *analogically* (Carbonell, 1986; Gentner, 1990; Kedar–Cabelli, 1988; Kodratoff, 1990; Winston, 1980) infer $Q(b)$:

$$\left\{ \begin{array}{c} P(a), \\ \forall_x [P(x) \rightarrow Q(x)], \\ (\text{“}b \text{ ‘similar’ to } a\text{”}) \end{array} \right\} \models Q(b)$$

Another way to make plausible inferences is to use weaker correlations between knowledge pieces (e.g. related facts, determinations, dependencies, "A is like B" statements, etc.).

Let us consider, for instance, that the KB contains the following related facts (each set describing an object):

$$P(c) \wedge Q(c) \wedge R(c)$$
$$P(d) \wedge Q(d) \wedge S(d)$$
$$P(e) \wedge Q(e)$$

Then one might empirically generalize (Mitchell, 1978; Michalski, 1983; Quinlan, 1986) these sets of facts to the rule

$$\forall_x [P(x) \rightarrow Q(x)]$$

and might deductively use this rule with the fact $P(a)$ to predict that $Q(a)$ is also true.

Analogical inferences could be made by employing plausible determinations (Russell, 1989; Tecuci, 1993). Let us consider, for instance, the following determination rule stating that U plausibly determines V:

$$U(x,y) \succ V(x,z)$$

Then one may make the following analogical inference:

$$U(s,a) \wedge V(s,b) \wedge U(t,a) \models V(t,b)$$

Another example of plausible inference is the "useful analogical inference", introduced by Greiner (1988). Let us suppose, for instance, that the system is told that $Q(b)$ is true, and is given the analogical hint *"b is like a"*, in order to show that $KB \models Q(b)$. That is, without this analogical hint, the system is not able to show that $KB \models Q(b)$. Based on this analogical hint, the system is looking for a feature of a (e.g. $P(a)$) which, if possessed by b, would allow it to prove $KB \models Q(b)$:

$$KB \models P(a)$$

$$KB \not\models P(b)$$

$$KB \not\models \neg P(b)$$

$$\{P(b), KB\} \models Q(b)$$

As a result of this reasoning $P(b)$ is asserted into the KB.

Several other types of plausible derivations based on implications and dependencies are described in (Collins and Michalski, 1989).

In order to show that a certain fact, I, is plausibly entailed by the KB, a system is not restricted to making only one plausible inference. In general, it could build a plausible justification tree (Tecuci, 1993). A plausible justification tree is like a proof tree, except that the inferences which compose it may be the result of different types of reasoning (not only deductive, but also analogical, abductive, predictive, etc.). An example of such a tree is presented in Figure 5.

One of the main reasons for illustrating the above plausible inferences was to show that, by employing a plausible inference engine, one could significantly extend the set of problems that could be solved by a system.

Figure 2 presents our conjecture about the relationships between the plausible closure of the KB, the deductive closure of the KB, and the set of true facts in the application domain:

- PC is a "soft" set, the boundaries of which are not strictly defined. Indeed, depending of the number and of the strength of the different types of plausible reasoning steps in a justification tree for a fact F, the plausibility of F is higher or lower.
- $PC \supset DC$ because the deductive proof trees are special cases of plausible justification trees.
- $PC \cap TC$ represents the set of facts which are plausibly entailed by the KB and are true.
- $PC \cap TC - DC$ represents the set of true facts which are plausibly entailed by the KB, but are not deductively entailed by the KB. Our hypothesis is that this is a significantly large set.
- $TC - PC$ represents the set of true facts that are not plausibly entailed by the KB. Although this set is not well defined, it expresses the intuition that there are true facts which even a plausible inference engine could not derive from the current KB.

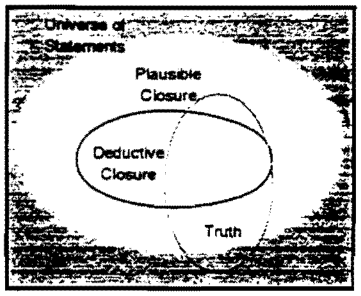

Figure 2: The relationship between DC, PC, and TC.

The deductive and plausible closures are two approximations of truth. In the approach we are proposing, we are considering DC as being an approximate lower bound for TC, and PC as being an approximate upper bound for TC. With this interpretation, the KB refinement problem reduces to one of determining the set TC in the plausible space defined by DC and PC. More precisely, during KB refinement, DC will be extended with a significant portion of $PC \cap TC$, and will also be corrected to remove from it most of $DC - TC$. Consequently, as a result of this process, DC will become a good approximation of TC.

Otherwise stated, *we propose an approach to KB refinement which is viewed as a transfer of knowledge form the plausible closure to the deductive closure.*

In this paper we are proposing a heuristic method which is an effective way of extending DC with a significant portion of

$PC \cap TC$. More precisely, during knowledge refinement, the sets DC and PC are transformed as follows:

- DC is extended by acquiring new facts or rules, or by generalizing some of the rules;
- DC is improved by specializing some of the deductive rules which could be partially incorrect;
- PC is extended and/or improved by acquiring new facts or rules, or improving some of the existing rules.

The next section contains a general presentation of the proposed KB refinement method.

3. General presentation of the KB refinement method

The KB of the system is assumed to be incomplete and partially incorrect. The KB is improved during training sessions with a human expert who provides the system with new input information I. Each such input I is an example of an answer that the final expert system should be able to generate, that is, I should be in the deductive closure of the final expert system. The goal of KB refinement is to improve the KB of the system so that to answer questions as the human expert.

If an input I is already in the plausible closure of the KB, then the system will be able to make a significant transfer of knowledge from the plausible closure to the deductive closure. More precisely, it will extend the deductive closure with new hypotheses, H, so as to include a generalization I_g of I, that is

$$\{H, KB\} \models I_g$$

At the same time, it will extend the plausible closure of the KB, so as to include more of TC, and might also remove some inconsistencies from the deductive closure, reducing the size of the set $DC - TC$.

If I is not in the plausible closure of the KB, then it will be simply asserted into the KB. This has the effect of extending both DC and PC. Indeed, the presence of I in DC may make it possible for the system to show that other facts (e.g. I_1, I_2) are deductively or plausibly entailed by the KB:

$$KB \not\models I_1, \text{ but } \{I, KB\} \models I_1$$

$$KB \not\models I_2, \text{ but } \{I, KB\} \models I_2$$

This also shows that during KB refinement, PC may grow to include facts from $TC - PC$.

The main stages of the KB refinement process are presented in Figure 3. They are:

- multitype inference and generalization;
- experimentation, verification and repair;
- goal-driven knowledge elicitation.

In the first stage, the system analyzes the input in terms of its current knowledge by building a plausible justification tree which demonstrates that the input is a plausible consequence of the system's current knowledge.

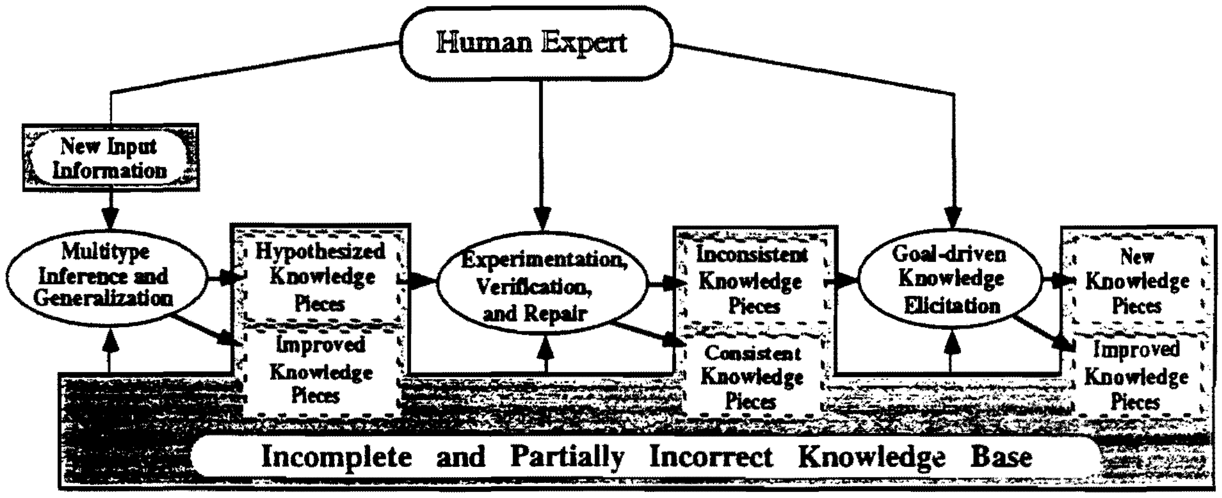
Human Expert
New Input Information
Multitype Inference and Generalization
Hypothesized Knowledge Pieces
Improved Knowledge Pieces
Experimentation, Verification, and Repair
Inconsistent Knowledge Pieces
Consistent Knowledge Pieces
Goal-driven Knowledge Elicitation
New Knowledge Pieces
Improved Knowledge Pieces
Incomplete and Partially Incorrect Knowledge Base

Figure 3: The main stages of the KB refinement process.

As a result of the analysis of the input via multitype inferences, new pieces of knowledge are hypothesized (through analogy, abduction, inductive generalization and prediction, etc.), and existing pieces of knowledge are improved so that the extended knowledge base to entail the input. By asserting these pieces of knowledge into the knowledge base, the system is able to deductively entail the input. The support for these new pieces of knowledge is that they allow building a logical connection (the justification tree) between a knowledge base that represents a part of the real world, and a piece of knowledge (the input) that is known to be true in the real world.

Next, the system will generalize the plausible justification tree, by employing different types of generalizations (not only deductive or empirical, but also based on analogy and, possibly, on other types of inferences). By this, it will generalize the hypothesized knowledge, so that the resulting knowledge base will entail not only the received input I, but also a generalization of it I_g.

The generalized plausible justification tree shows how I_g (a generalization of the input I) is entailed by the KB. However, this tree was obtained by making both plausible inferences and plausible generalizations. Consequently, both the tree and the corresponding knowledge pieces learned are less certain. One may improve them by performing experiments. This is the second stage of the KB refinement process. During this stage, the system will generate instances of I_g and will ask the user if they are true or false. It will further improve the hypothesized knowledge pieces so that the updated KB to deductively entail the instances of I_g which are true and to reject the ones which are false.

However, because the KB is incomplete and possibly partially incorrect, some of the learned knowledge pieces may be inconsistent (i.e. may cover negative examples). In order to remove such inconsistencies, additional knowledge pieces (which represent new terms in the representation language of the system) are elicited from the expert, through several consistency-driven knowledge elicitation techniques. This represents the third stage of the KB refinement process.

The entire knowledge refinement process is characterized by a cooperation between the learning system and the human expert in which the learner performs most of the tasks and the expert helps it in solving the problems that are intrinsically difficult for a learner (e.g., the credit/blame assignment problem, the problem of new terms) and relatively easy for the human expert.

4. Exemplary application domain

We will use the domain of workstation allocation and configuration in order to illustrate this KB refinement method. The expert system to be built has to reason about which machines are suitable for which tasks and to allocate an appropriate machine for each task.

The initial (incomplete and partially incorrect) knowledge base contains information about various printers and workstations distributed throughout the workplace. A sample of this knowledge base is presented in Figure 4. Notice that it contains different types of knowledge: deductive rules, a plausible determination (Russell, 1989; Tecuci, 1993), facts, and hierarchies. Each of these knowledge pieces might be incomplete and/or partially incorrect.

Let us suppose that the system is told that macII02 is suitable for publishing

suitable(macII02, publishing)

and this fact is representative of the type of answers it should be able to provide.

5. Multitype inference and generalization

The system tries to analyze ("understand") the input in terms of its current knowledge by building the plausible justification tree in Figure 5. Such a tree demonstrates that the input is a plausible consequence of the system's current knowledge. The method for building such a tree is presented in (Tecuci, 1993). It employs a backward chaining uniform–cost search.

The tree in Figure 5 is composed of four deductions, an inductive prediction, and a determination–based analogical implication.

```
suitable(X, publishing) :-                          ; X is suitable for publishing if it runs
        runs(X, publishing-sw), communicate(X, Y),  ; publishing software and communicates
        isa(Y, high-quality-printer).               ; with a high quality printer

communicate(X, Y) :-                     ; X and Y communicate if they are on the same network
        on(X, Z), on(Y, Z).

communicate(X, Y) :-                     ; X and Y communicate if they are on connected networks
        on(X, Z), on(Y, V), connect(Z, V).

isa(X, high-quality-printer) :-                                  ; X is a high quality printer if
        isa(X, printer), speed(X, high), resolution(X, high).    ; it has high speed and resolution

runs(X, Y) :- os(X, Z).          ; the type of software which a machine could run is largely determined
                                 ; by its operating system (":-" means plausible determination)

runs(X, Y) :- runs(X, Z), isa(Z, Y).

os(sun01, unix).    on(sun01, fddi).      speed(sun01, high).     processor(sun01, risc).
; sun01's operating system is unix, it is on the fddi network, has high speed and a risc processor

os(hp05, unix).   on(hp05, ethernet).   speed(hp05, high).   processor(hp05, risc).   runs(hp05, frame-maker).
os(macplus07, mac-os).           on(macplus07, appletalk).
os(macII02, mac-os).             on(macII02, appletalk).
os(maclc03, mac-os).             runs(maclc03, page-maker).
on(proprinter01, ethernet).      resolution(proprinter01, high).      processor(proprinter, risc).
on(laserjet01, fddi).            resolution(laserjet01, high).        processor(laserjet01, risc).
on(microlaser03, ethernet).      resolution(microlaser03, high).      processor(microlaser03, risc).
resolution(xerox01, high).       speed(xerox01, high).                processor(xerox01, risc).
connect(appletalk, ethernet).    connect(appletalk, fddi).            connect(fddi, ethernet).
```

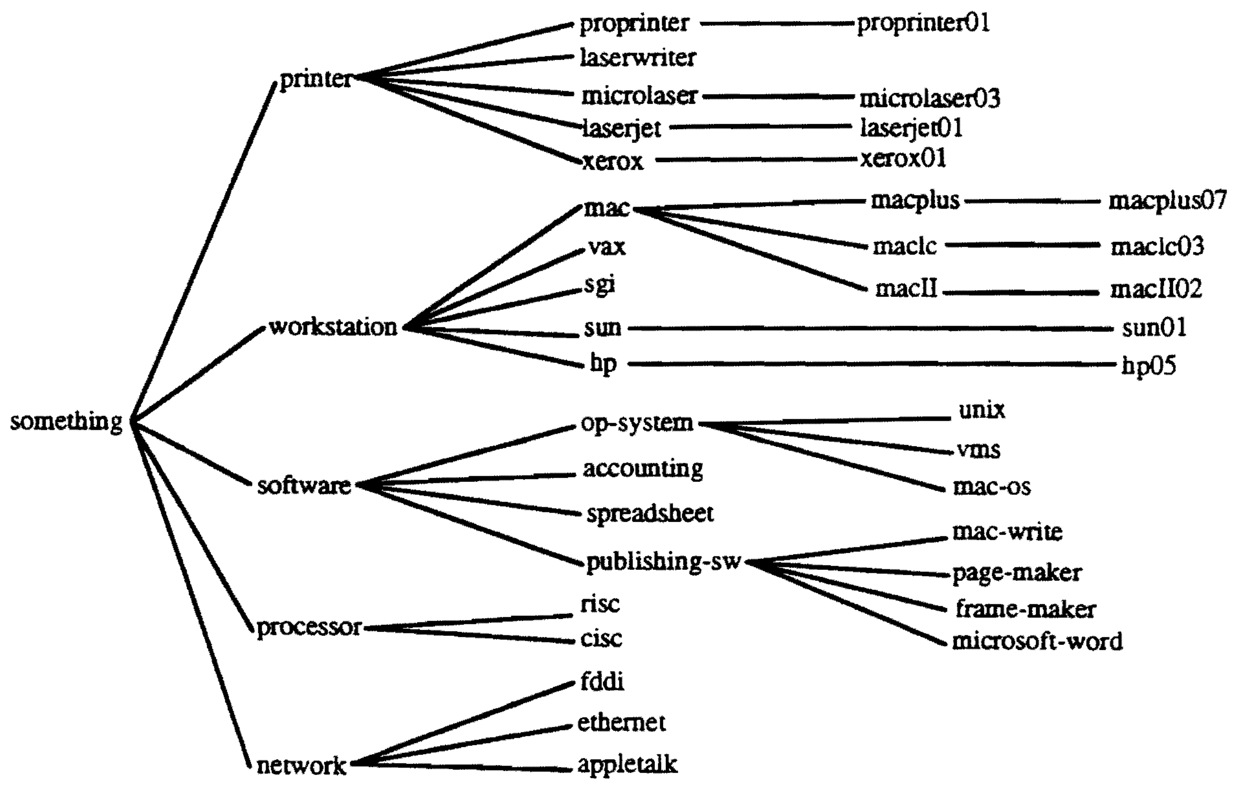

Figure 4: Sample of an incomplete and partially incorrect KB for the domain of workstation allocation and configuration.

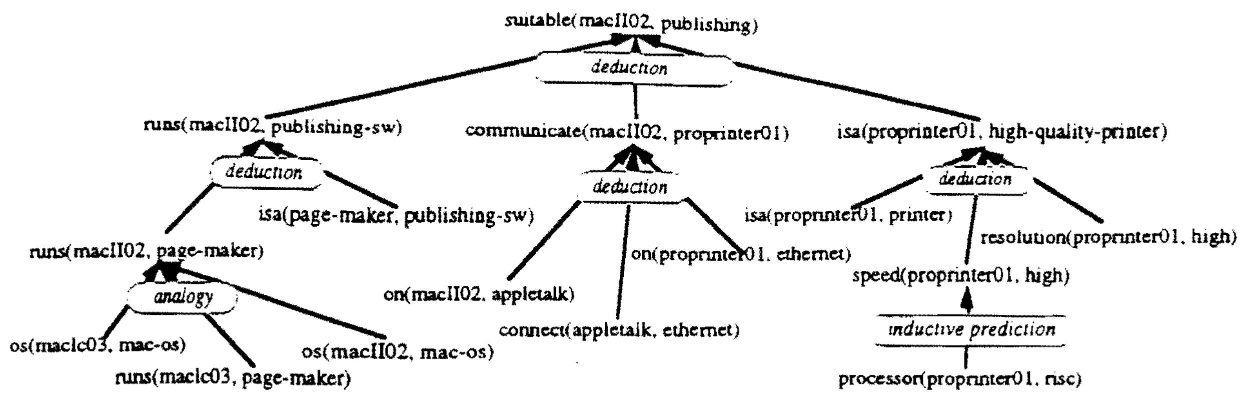
suitable(macII02, publishing)
deduction
runs(macII02, publishing-sw)
communicate(macII02, proprinter01)
isa(proprinter01, high-quality-printer)
deduction
isa(page-maker, publishing-sw)
deduction
deduction
isa(proprinter01, printer)
resolution(proprinter01, high)
runs(macII02, page-maker)
on(proprinter01, ethernet)
speed(proprinter01, high)
analogy
on(macII02, appletalk)
connect(appletalk, ethernet)
inductive prediction
os(maclc03, mac-os)
os(macII02, mac-os)
runs(maclc03, page-maker)
processor(proprinter01, risc)

Figure 5: A plausible justification tree for "suitable(macII02, publishing)".

The inductive prediction

processor(proprinter01,risc) → speed(proprinter01,high)

was made by empirically generalizing the facts:

speed(sun01, high), os(sun01, unix), on(sun01, fddi), processor(sun01, risc).

speed(hp05, high), os(hp05, unix), on(hp05, ethernet), processor(hp05, risc), runs(hp05, frame–maker).

speed(xerox01, high), resolution(xerox01, high), processor(xerox01, risc).

to the rule

speed(x, high) :– processor(x, risc)

and then applying this rule deductively.

An open problem is how to collect the facts to be generalized, and what kind of generalization to look for. One solution which we are investigating is based on CLINT's approach (De Raedt, 1991) of using a hierarchy of languages for the rules to be learned. Each language is characterized by a certain form of the rules to be learned, which suggests the kind of facts to look for.

The analogical inference was made by using the determination rule

runs(X, Y) :– os(X, Z)

as indicated in Figure 6.

While there may be several justification trees for a given input, the attempt is to find the most simple and the most plausible one (Lee, 1993). This tree shows how a true fact *I* derives from other true facts from the KB. Based on the Occam's razor (Blumer, Ehrenfeucht, Haussler, and Warmuth, 1987), and of the general hypothesis used in abduction which states that the best explanation of a true fact is most likely to be true (Peirce, 1965), one could assume that all the inferences from the most simple and plausible justification tree are correct.

With this assumption, the KB is improved by:

- learning a new rule by empirical inductive generalization:

  speed(X, high) :– processor(X, risc)
  with the positive examples
  X=sun01, X=hp05, X=xerox01, X=proprinter01

- discovering positive examples of the determination rule (which is therefore enforced):

  runs(X, Y) :– os(X, Z).
  with the positive examples
  X=maclc03, Y=page-maker, Z=mac–os
  X=macII02, Y=page-maker, Z=mac–os

- discovering positive examples for the deductive rules used in building the plausible justification tree as, for instance:

  suitable(X, publishing) :–
  runs(X, publishing–sw), communicate(X, Y),
  isa(Y, high–quality–printer).
  with the positive example
  X=macII02, Y=proprinter01

Therefore, the user merely verifying a statement allows the system to refine the KB by making several justified hypotheses. As a result of these improvements

$$\text{KB} \models \text{suitable(macII02, publishing)}$$

During KB refinement, the rules are constantly updated so as to remain consistent with the accumulated examples. This is a type of incremental learning with full memory of past examples.

As mentioned before, the input fact "suitable(macII02, publishing)" is representative for the kind of answers the final system should be able to generate. This means that the final system should be able to give other answers of the form "suitable(x, y)". It is therefore desirable to extend *DC* so as to include other such true facts, but also to improve *DC* so as no longer to include false facts of the same form.

While the integration of the fact "suitable(macII02, publishing)" into *DC* was a costly process that involved multitype inferences and the determination of the most plausible justification tree, the integration in (or exclusion from) *DC* of similar facts is a much simpler process which basically replicates most of the reasoning involved in the "understanding" of the input "suitable(macII02, publishing)". This feature is one of the main strengths of the proposed KB refinement method.

The basic idea is the following one. One performs a costly reasoning process to show that $KB \models I$. Then it computes a generalization of that reasoning so as to speed up future problem solving which requires a similar reasoning. Indeed, such a reasoning process could be generated by simply instantiating this generalization to the new problem to be solved.

A simple illustration of this idea is explanation–based learning (Mitchell, Keller, Kedar–Cabelli, 1986; DeJong and Mooney, 1986). In this case, an explanation (proof tree) of a concept example is deductively generalized. Different instances of this deductive generalization demonstrate that other descriptions are examples of the same concept.

In our method, each inference from the plausible justification tree is replaced with a generalization which depends of the type of inference, as shown in (Tecuci, 1993). Thus, the system is performing not only deductive generalizations, but also empirical inductive generalizations, generalizations based on different types of analogies, and possibly, even generalizations based on abduction. To illustrate this, let us consider the analogical implication from Figure 5. The process of making this inference is illustrated in Figure 6.

According to the plausible determination rule "runs(X, Y) :~ os(X, Z)", the software which a machine can run is largely determined by its operating system. It is known that the operating system of "maclc03" is "mac–os", and that it runs "page-maker". Because the operating system of "macII02" is also "mac–os", one may infer by analogy that "macII02" could also run "page-maker".

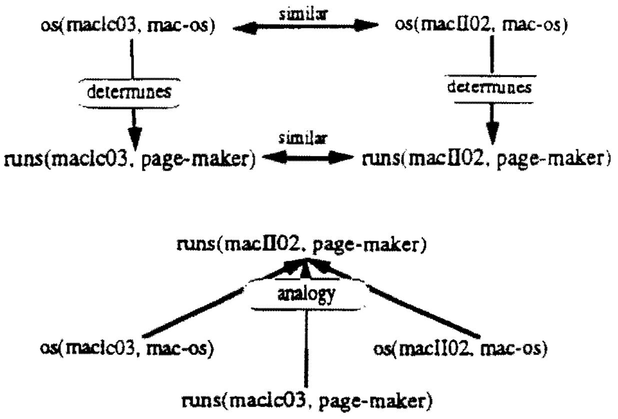

Figure 6: Inferring "runs(macII02, page-maker)" by analogy.

Let us notice now that the same kind of reasoning is valid for any type of operating system, and for any type of software, as illustrated in Figure 7.

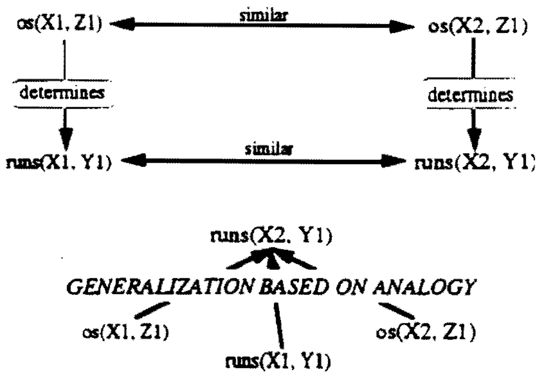

Figure 7: Generalization of the reasoning illustrated in Figure 6.

Now, if one knows, for instance, that "os(hp05, unix)", "runs(hp05, frame-maker)", and "os(sun01, unix)", then one may immediately infer "runs(sun01, frame-maker)", by simply instantiating the general inference from the bottom of Figure 7.

This might not appear to be a significant saving but, in the case of a plausible justification tree, one generalizes several such individual inferences and, even more importantly, their interconnection in a plausible reasoning process. For instance, the

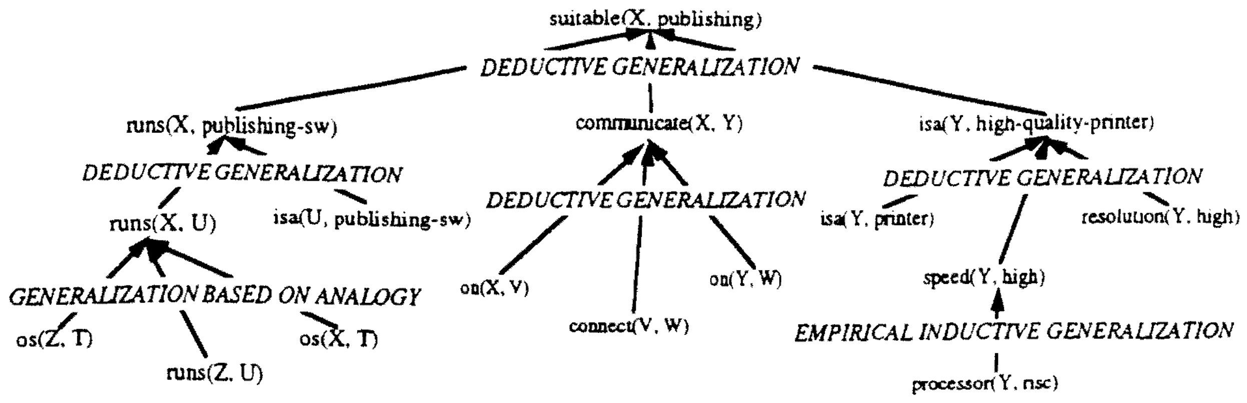

Figure 8: A generalized plausible justification tree.

generalization of the tree in Figure 5 is presented in Figure 8. The generalization method is presented in (Tecuci, 1993).

The important thing about the general tree in Figure 8 is that it covers many of the plausible justification trees for facts of the form "suitable(x, publishing)". If, for instance, sun01 is another computer for which the leaves of the plausible justification tree in Figure 8 are true, then the system will infer that sun01 is also suitable for publishing, by simply instantiating the tree in Figure 8 (see Figure 9).

Let us mention again that while building the plausible justification tree in Figure 5 was a difficult problem which required the employment of different types of reasoning, and of determining the simplest and the most plausible justification tree, the building of the tree in Figure 9 was a very simple process of instantiating the tree in Figure 8. However, once this tree is built, one may draw conclusions that are similar to the ones drawn from the tree in Figure 5:

- if the top of the tree in Figure 9 is known to be true, then one may assume that all the intermediate implications are also valid. This reinforces each rule used in building the tree with a new positive example.
- if the top of the tree is not true, then one has to identify the wrong implication and to correct accordingly the KB.

6. Experimentation, verification and repair

Building the plausible justification tree from Figure 5 and its generalization from Figure 8 was the first stage of the KB refinement process described in Figure 3. The next stage is one of experimentation, verification, and repair.

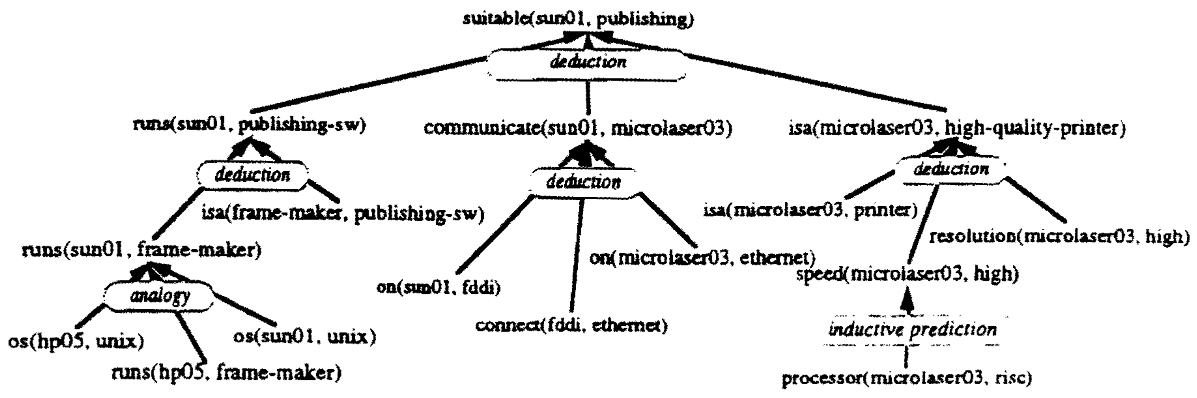
suitable(sun01, publishing)
deduction
runs(sun01, publishing-sw)
communicate(sun01, microlaser03)
isa(microlaser03, high-quality-printer)
deduction
deduction
deduction
isa(frame-maker, publishing-sw)
isa(microlaser03, printer)
resolution(microlaser03, high)
runs(sun01, frame-maker)
on(microlaser03, ethernet)
speed(microlaser03, high)
analogy
on(sun01, fddi)
inductive prediction
os(hp05, unix)
os(sun01, unix)
connect(fddi, ethernet)
runs(hp05, frame-maker)
processor(microlaser03, risc)

Figure 9: An instance of the plausible justification tree in Figure 8, justifying that sun01 is suitable for publishing.

The system will generate plausible justification trees like the one in Figure 9. These trees show how statements of the form "suitable(x, publishing)" plausibly derive from the KB. Each such statement is shown to the user who is asked if it is true or false. Then, the system (with the expert's help) will update the KB such that it will deductively entail the true statements and only them.

The experimentation phase is controlled by a heuristic search in a plausible version space (*PVS*) which limits significantly the number of experiments needed to improve the KB. In the case of our example, the plausible version space is defined by the trees in Figure 5 and Figure 8, and is represented in Figure 10.

```
plausible upper bound
suitable(X, publishing) :-
   os(Z, T), runs(Z, U), os(X, T),
   isa(U, publishing-sw), on(X, V),
   connect(V, W), on(Y, W),
   isa(Y, printer), processor(Y, risc),
   resolution(Y, high).
plausible lower bound
suitable(macII02, publishing) :-
   os(maclc03, mac-os), runs(maclc03, page-maker),
   os(macII02, mac-os),
   isa(page-maker, publishing-sw),
   on(macII02, appletalk), connect(appletalk,ethernet),
   on(proprinter01, ethernet), isa(proprinter01,printer),
   processor(proprinter01, risc),
   resolution(proprinter01, high).
```

Figure 10: The plausible version space (*PVS*)

The plausible upper bound is a rule the left hand side of which is the top of the general tree in Figure 8, and the right hand side of which is the conjunction of the leaves of the same tree. The plausible lower bound is a similar rule corresponding to the tree in Figure 5. This plausible version space synthesizes some of the inferential capabilities of the system with respect to the facts of the form "suitable(x, publishing)". We call these bounds plausible because they are only approximations of the real bounds (Tecuci, 1992). The upper bound rule is supposed to be more general than the exact rule for inferring "suitable(x, publishing)", and the lower bound rule is supposed to be less general than this rule. Let us notice that this version space corresponds to the version space in Figure 2. The plausible upper bound corresponds to the plausible closure, and the plausible lower bound corresponds to the deductive closure. Of course, this space is restricted to facts of the form "suitable(x, publishing)".

The version space in Figure 10 could be represented in the equivalent form in Figure 11. Note that the facts of the form "isa(Q, something)" are always true.

```
suitable(X, publishing) :-
  plausible upper bound
  isa(T, something), isa(U, publishing-sw),
  isa(V, something), isa(W, something),
  isa(X, something), isa(Y, printer),
  isa(Z, something), os(Z, T), runs(Z, U), os(X, T),
  on(X, V), connect(V, W), on(Y, W),
  processor(Y, risc), resolution(Y, high).
  plausible lower bound
  isa(T, mac-os), isa(U, publishing-sw),
  isa(V, appletalk), isa(W, ethernet),
  isa(X, macII02), isa(Y, printer),
  isa(Z, maclc03), os(Z, T), runs(Z, U), os(X, T),
  on(X, V), connect(V, W), on(Y, W),
  processor(Y, risc), resolution(Y, high).
with the positive example
  T=mac-os, U=page-maker, V=appletalk,
  W=ethernet, X=macII02, Y=proprinter01,
  Z=maclc03.
```

Figure 11: Equivalent form of the plausible version space in Figure 10.

The version space in Figure 11 serves both for generating facts of the form "suitable(x, publishing)", and for determining the end of the experimentation phase.

To generate such a fact, the system looks into the KB for an instance of the upper bound which is not an instance of the lower bound. Such an instance is the following one:

```
suitable(sun01, publishing) :-
   os(hp05, unix),
   runs(hp05, frame-maker),
   os(sun01, unix),
   isa(frame-maker, publishing-sw),
   on(sun01, fddi),
   connect(fddi, ethernet),
   on(microlaser03, ethernet),
   isa(microlaser03, printer),
   processor(microlaser03, risc),
   resolution(microlaser03, high).
```

which could be written as:

```
suitable(X, publishing) :-
  isa(T, unix), isa(U, publishing-sw), isa(V, fddi),
  isa(W, ethernet), isa(X, sun01),
  isa(Y, microlaser03), isa(Z, hp05), os(Z, T),
  runs(Z, U), os(X, T), on(X, V), connect(V, W),
  on(Y, W), processor(Y, risc), resolution(Y, high).
```
with the positive example
```
  T=unix, U=frame-maker, V=fddi, W=ethernet,
  X=sun01, Y=microlaser03, Z=hp05.
```

The corresponding instance of the general tree in Figure 8 shows how "suitable(sun01, publishing)" is plausibly entailed by the KB (see Figure 9). The user is asked if "suitable(sun01, publishing)" is true or false, and the KB is updated accordingly. Assuming the user accepted "suitable(sun01, publishing)" as a true fact, the KB and the plausible version space are updated as follows:

- the KB is improved so as to deductively entail "suitable(sun01, publishing)";
- the plausible lower bound of the PVS is conjunctively generalized to "cover" the leaves of the tree in Figure 9.

It has already been shown how the KB is improved (see section 5). The plausible lower bound of the PVS is generalized as shown in Figure 12.

Let us also consider the case of a generated fact which is rejected by the user:

"suitable(macplus07, publishing)".

The corresponding plausible justification tree is shown in Figure 13. This tree was obtained by instantiating the general tree in Figure 8 with facts from the KB. It shows how a false fact is plausibly entailed by the KB.

```
suitable(X, publishing) :-
```
plausible upper bound
```
  isa(T, something), isa(U, publishing-sw),
  isa(V, something), isa(W, something),
  isa(X, something), isa(Y, printer),
  isa(Z, something), os(Z, T), runs(Z, U), os(X, T),
  on(X, V), connect(V, W), on(Y, W),
  processor(Y, risc), resolution(Y, high).
```
plausible lower bound
```
  isa(T, op-system), isa(U, publishing-sw),
  isa(V, network), isa(W, ethernet),
  isa(X, workstation), isa(Y, printer),
  isa(Z, workstation), os(Z, T), runs(Z, U), os(X, T),
  on(X, V), connect(V, W), on(Y, W),
  processor(Y, risc), resolution(Y, high).
```
with the positive example
```
  T=mac-os,U=page-maker,V=appletalk,W=ethernet,
    X=macII02, Y=proprinter01, Z=maclc03.
  T=unix, U=frame-maker, V=fddi, W=ethernet,
    X=sun01, Y=microlaser03, Z=hp05.
```

Figure 12: Updated PVS.

In such a case one has to detect and correct the wrong inference(s), as well as to update the KB, the general justification tree in Figure 8, and the plausible version space such that:

- the tree in Figure 13 is no longer a plausible justification tree;
- the KB does not deductively entail "suitable(macplus07, publishing)";
- the updated general justification tree no longer covers the tree in Figure 13;
- the plausible upper bound of the PVS is specialized so that it no longer covers the leaves of the tree in Figure 13.

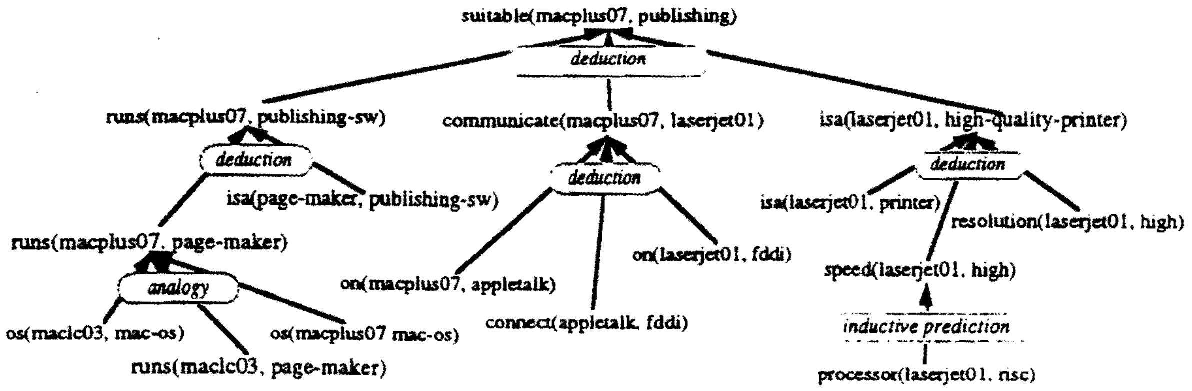
suitable(macplus07, publishing)
deduction
runs(macplus07, publishing-sw)
communicate(macplus07, laserjet01)
isa(laserjet01, high-quality-printer)
deduction
isa(page-maker, publishing-sw)
runs(macplus07, page-maker)
analogy
os(maclc03, mac-os)
os(macplus07 mac-os)
runs(maclc03, page-maker)
deduction
on(macplus07, appletalk)
connect(appletalk, fddi)
on(laserjet01, fddi)
deduction
isa(laserjet01, printer)
resolution(laserjet01, high)
speed(laserjet01, high)
inductive prediction
processor(laserjet01, risc)

Figure 13: Another instance of the plausible justification tree in Figure 8, which shows how a false fact is plausibly entailed by the KB.

Detecting the wrong implication from the plausible justification tree in Figure 13 is an intrinsically difficult problem for an autonomous learning system. One possible solution, which is presented in (Tecuci, 1993), is to blame the implication which is the least plausible, and the correction of which requires the smallest change in the KB. For a human expert, however, it should not be too difficult to identify the wrong implication and even to find the explanation of the failure (Tecuci, 1992). In the case of the tree in Figure 13, the wrong implication could be identified by the user as being the deduction from the top of the tree:

```
suitable(macplus07, publishing) :-
    runs(macplus07, publishing-sw),
    communicate(macplus07, laserjet01),
    isa(laserjet01, high-quality-printer).
```

Although macplus07 runs publishing software and communicates with a high quality printer, it is not suitable for publishing because it does not have a large display.

Consequently, the rule which generated the above implication is specialized as follows (requiring X to have a large display):

suitable(X, publishing) :-
  runs(X, publishing-sw), display(X, large),
  communicate(X, Y), isa(Y, high-quality-printer).
  with the positive examples
    X=macII02, Y=proprinter01.
    X=sun01, Y=microlaser03.
  with the negative example
    X=macplus07, Y=laserjet01.

One should notice that the predicate "display(X, Y)" could be defined by the user, or could be suggested by the system as one which distinguishes the known positive examples of the rule from the discovered negative example.

As a result of updating the above rule, the general plausible justification tree in Figure 8 is updated as shown in Figure 14, and the version space is updated as shown in Figure 15.

It might not always be easy to identify the problem with a wrong inference, and to specialize the corresponding rule so as no longer to cover the negative example. In such a case, the wrong inference is kept as a negative exception of the rule which generated it, as shown in Figure 16.

suitable(X, publishing) :-
  plausible upper bound
  isa(T, something), isa(U, publishing-sw),
  isa(V, something), isa(W, something),
  isa(X, something), isa(Y, printer), isa(Z, something),
  os(Z, T), runs(Z, U), os(X, T), display(X, large),
  on(X, V), connect(V, W), on(Y, W),
  processor(Y, risc), resolution(Y, high).
  plausible lower bound
  isa(T, op-system), isa(U, publishing-sw),
  isa(V, network), isa(W, ethernet),
  isa(X, workstation), isa(Y, printer),
  isa(Z, workstation), os(Z, T), runs(Z, U), os(X, T),
  display(X, large), on(X, V), connect(V, W),
  on(Y, W), processor(Y, risc), resolution(Y, high).
with the positive example
  T=mac-os, U=page-maker, V=appletalk, W=ethernet,
    X=macII02, Y=proprinter01, Z=maclc03.
  T=unix, U=frame-maker, V=fddi, W=ethernet,
    X=sun01, Y=microlaser03, Z=hp05.
  with the negative example
  T=mac-os, U=page-maker, V=appletalk, W=fddi,
    X=macplus07, Y=laserjet01, Z=maclc03.

Figure 15: Updated PVS.

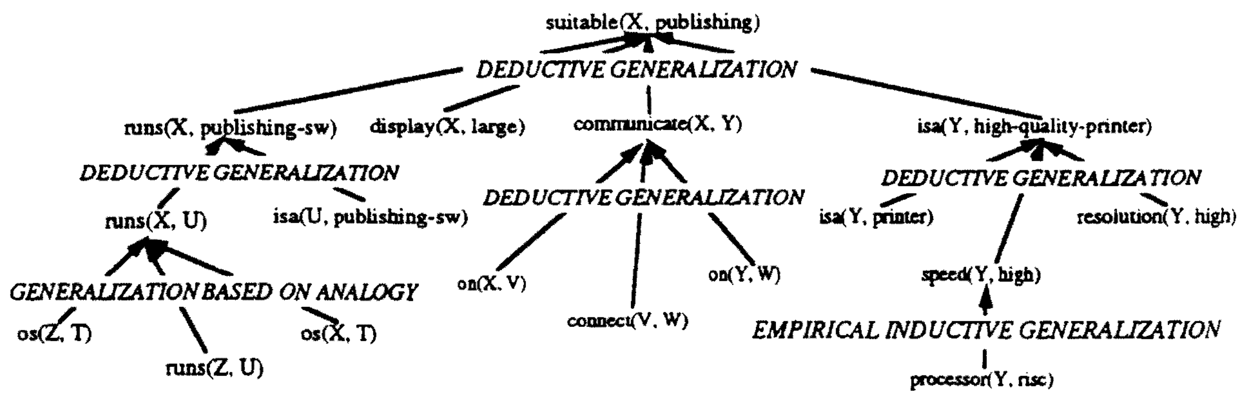

Figure 14: Updated general justification tree.

```
suitable(X, publishing) :-
  runs(X, publishing-sw), communicate(X, Y),
  isa(Y, high-quality-printer).
  with the positive examples
    X=macII02, Y=proprinter01.
    X=sun01, Y=microlaser03.
  with the negative exception
    X=macplus07, Y=laserjet01.
```

Figure 16: A rule with a negative exception.

During experimentation, the lower bound of the plausible version space is generalized so as to cover the generated facts accepted by the user (the positive examples), and the upper and lower bounds are specialized so as to no longer cover the generated facts rejected by the user (the negative examples). This process will end in one of the following situations:

- the bounds of the plausible version space become identical.
- 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 "suitable(x, publishing)" can be generated.

Notice that the plausible version space is only used for controlling the experimentation phase. It is not kept in the KB as a new rule for inferring "suitable(x, publishing)" because it would be a redundant rule.

7. Goal-driven knowledge elicitation

Because the KB is incomplete and partially incorrect, some of the learned knowledge pieces may be inconsistent (i.e. may cover negative examples), as it is illustrated in Figure 16. In order to remove such inconsistencies, additional knowledge pieces (which may represent new terms in the representation language of the system) are elicited from the expert, through several consistency-driven knowledge elicitation methods, as described in (Tecuci and Hieb, 1993). These methods are applied in the third phase of KB refinement, as shown in Figure 3.

Let us consider the case of the inconsistent rule in Figure 16.

One consistency-driven knowledge elicitation method is to look for a new predicate which could characterize all the positive instances of X (although it might not be associated with each of these instances), without characterizing any negative exception of X. A potentially discriminating predicate like "display" is one which characterizes a positive instance of X (either macII02 or sun01), and does not characterize the negative exception of X (macplus07). If such a predicate is found, then the user is asked if it characterizes all the other positive instances of X. The same technique could, of course, be applied to the instances of Y.

It may happen, however, that the system cannot find a property to transfer from one positive example of X to the others. In such a case, it will try to elicit a new property by using a technique similar to the triad method employed in the elicitation of the repertory grids (Boose and Bradshaw, 1988; Shaw and Gaines, 1987).

Another method for removing the negative exception is to look for a relationship between X and Y which could characterize all the positive instances of X and Y, without characterizing the negative exception.

Yet another method is to define a new concept that covers all the positive instances of X (or all the positive instances of Y), without covering the negative exception of X (Y). A method similar to this one is reported by (Wrobel, 1989).

8. Summary and conclusions

Figure 2 suggests two basic approaches to the development of the competence of a deductive knowledge-based system. One approach is to extend the deductive closure of the KB by acquiring new knowledge. The other approach is to replace the deductive inference engine with a plausible inference engine, and thus to enable the system to solve additional problems from the plausible closure. The first approach has the advantage that the system employs "sound" reasoning, but it has the disadvantage of requiring a difficult knowledge acquisition process. The second approach has the advantage of avoiding knowledge acquisition, but the disadvantage that the system needs to rely on plausible reasoning.

The knowledge refinement method presented in this paper is an attempt to combine these two approaches in such a way as to take advantage of their complementarity. This method is summarized in Figure 17. It resulted from the merging and extension of two related research directions:

- the knowledge acquisition methodology of Disciple (Tecuci and Kodratoff, 1990) and NeoDisciple (Tecuci, 1992);
- the MTL-JT framework for multistrategy learning based on plausible justification trees (Tecuci, 1993).

On the one hand, it extends NeoDisciple with respect to the knowledge representation used and the types of inferences and generalizations employed and, on the other hand, it adapts and integrates the MTL-JT framework into an interactive knowledge acquisition scenario.

The method is based on the following general idea. The system performs a complex reasoning process to solve some problem *P*. Then it determines a justified generalization of the reasoning process so as to speed up the process of solving similar problems P_i. When the system encounters such a similar problem, it will be able to find a solution just by instantiating the above generalization.

In the context of the presented method, the problem to solve is to extend the KB so as to entail a new fact *I*. The complex reasoning process involved consists of building a plausible justification tree. This reasoning process is generalized by employing various types of generalization procedures. Then, during the experimentation phase, the system instantiates this generalization and, using it, improves the KB so as to entail similar facts which are true (or to no longer entail similar facts which are false).

One important aspect of the presented method is the notion of plausible justification trees (Tecuci and Michalski, 1991; Tecuci, 1993). Other systems have employed implicit justification trees (DeRaedt and Bruynooghe, 1993), or even explicit justification trees (Tecuci, 1988; Mahadevan, 1989; Widmer, 1989), that integrated only a small number of inferences. In our method, the plausible justification tree is defined as a general framework for integrating a whole range of inference types. Therefore, theoretically, there is no limit with respect to the type or

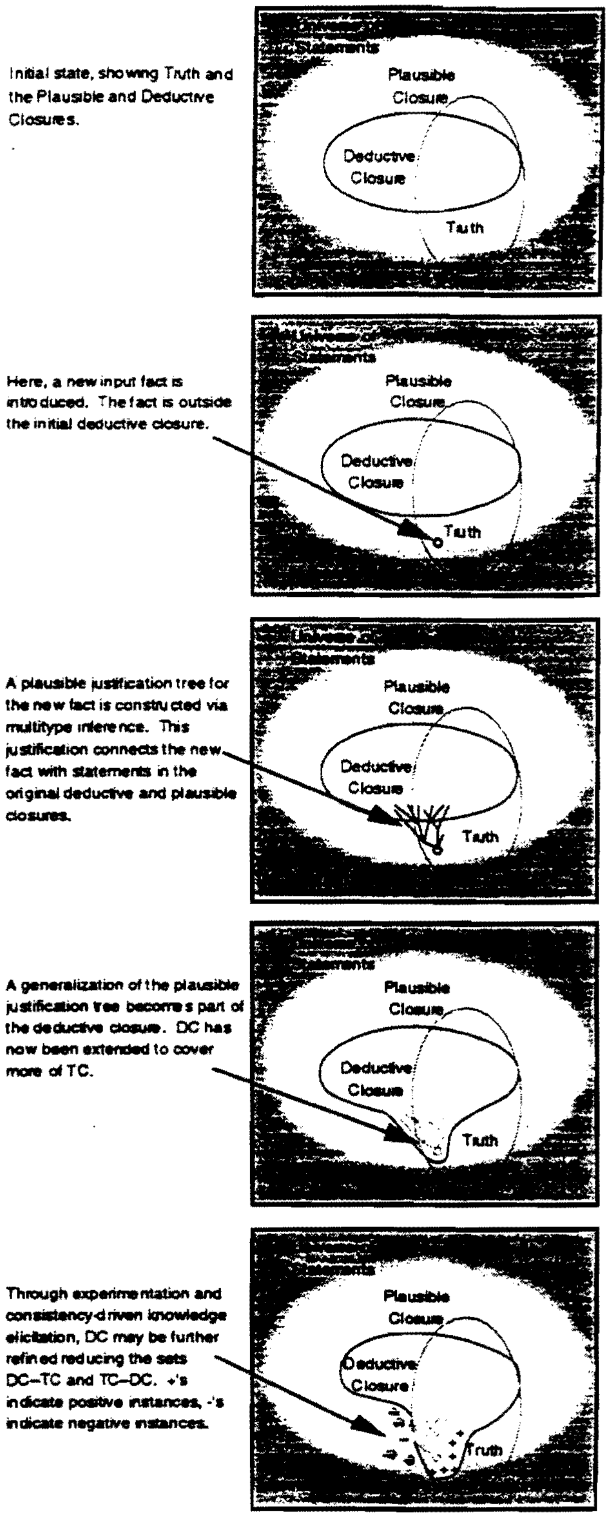
Initial state, showing Truth and the Plausible and Deductive Closures.
Here, a new input fact is introduced. The fact is outside the initial deductive closure.
A plausible justification tree for the new fact is constructed via multitype inference. This justification connects the new fact with statements in the the original deductive and plausible closures.
A generalization of the plausible justification tree becomes part of the deductive closure. DC has now been extended to cover more of TC.
Through experimentation and consistency-driven knowledge elicitation, DC may be further refined reducing the sets DC–TC and TC–DC. +'s indicate positive instances, -'s indicate negative instances.
Universe of Statements
Plausible Closure
Deductive Closure
Truth

Figure 17: Modification of DC and PC during KB refinement.

number of inferences employed in a plausible justification tree.

Another important feature of the KB refinement method is the employment of different types of generalizations. While the current machine learning research distinguishes only between deductive generalizations and inductive generalizations (Michalski, 1993), this method and the MTL-JT framework (Tecuci, 1993) suggest that one may consider other types of generalizations, each associated with a certain type of inference (as, for instance, generalization based on analogy).

There are also several ways in which the method could be improved. For instance, the set of inferences involved in the present version of the method is quite limited (deduction, determination-based analogy, inductive prediction, and abduction). New types of inferences should be included, as well as more complex versions of the current ones.

Also, new types of justified generalizations (each corresponding to a certain inference type) should be defined.

Finally, the goal-driven knowledge elicitation methods briefly mentioned in section 7 should be extended so as not only to add new concepts and relationships into the KB but also to delete those that become unnecessary.

Acknowledgments

The authors are grateful to Luc De Raedt and Michael Hieb for useful comments and criticisms. This research was done in the GMU Center for Artificial Intelligence. Research of the Center is supported in part by the National Science Foundation Grant No.IRI–9020266, in part by the Office of Naval Research grant No.N0014–91–J–1351, and in part by the Advanced Research Projects Agency grant No.N0014–91–J–1854, administered by the Office of Naval Research, and grant No. F49620–92–J–0549, administered by the Air Force Office of Scientific Research.

References

Blumer A., Ehrenfeucht A., Haussler D. and Warmuth M. K., Occam's Razor, *Information Processing Letters,* 24, pp.377–380, 1987.

Boose J.H. and Bradshaw J.M., 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.), AP, 1988.

Cain T., DUCTOR: A Theory Revision System for Propositional Domains, in *Machine Learning: Proc. of the Eighth Int. Workshop,* L.A. Birnbaum and G.C. Collins (Eds.), Chicago, IL, Morgan Kaufmann, 1991.

Carbonell J.G., Derivational Analogy: a Theory of Reconstructive Problem Solving and Expertise Acquisition, in *Machine Learning: An Artificial Intelligence Approach,* (Vol. 2), R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds), San Mateo, CA: Morgan Kaufmann, 1986.

Cohen W., The Generality of Overgenerality, in *Machine Learning: Proc. of the Eighth Int. Workshop,* L.A. Birnbaum and G.C. Collins (Eds.), Chicago, Morgan Kaufmann, 1991.

Collins A., and Michalski R.S., The Logic of Plausible Reasoning: A Core Theory, *Cognitive Science,* Vol. 13, pp. 1–49, 1989.

Davies T.R. and Russell S.J., A Logical Approach to Reasoning by Analogy, *Proc. IJCAI,* Milan, Italy, Morgan Kaufmann, 1987.

DeJong G. and Mooney R., Explanation–Based Learning: An Alternative View, *Machine Learning,* Vol. 1, 1986.

Ginsberg A., Weiss S.M. and Politakis P., Automatic Knowledge Base Refinement for Classification Systems, *Artificial Intelligence* 35, pp. 197–226, 1988.

Gentner D., The Mechanisms of Analogical Reasoning, in *Readings in Machine Learning,* J.W.Shavlik, T.G.Dietterich (Eds), Morgan Kaufmann, 1990.

Greiner R., Learning by Understanding Analogies, in *Analogica,* A. Prieditis (Ed.), Pitman, London, 1988.

Josephson J., Abduction: Conceptual Analysis of a Fundamental Pattern of Inference, Research Report 91–JJ, Laboratory for AI Research, Ohio State University, 1991.

Kedar–Cabelli S., Toward a Computational Model of Purpose–Directed Analogy, in *Analogica,* A. Prieditis (Ed.), Pitman, 1988.

Kodratoff Y., Using Abductive Recovery of Failed Proofs for Problem Solving by Analogy, in *Machine Learning: Proc. of the Seventh Int. Workshop,* B.W. Porter and R.J.

Mooney (Eds.), Morgan Kaufmann, 1990.

Lee O., A Control Strategy for Multistrategy Task-adaptive Learning with Plausible Justification Trees, Research Report, AI Center, George Mason University, 1993.

Mahadevan S., Using Determinations in Explanation-based Learning: A Solution to Incomplete Theory Problem, in *Proc. of the Sixth Int. Workshop on Machine Learning*, Ithaca, NY: Morgan Kaufman, 1989.

Michalski R.S., A 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.), Tioga Publishing Co., 1983.

Michalski R.S., Inferential Learning Theory: Developing Theoretical Foundations for Multistrategy Learning, in *Machine Learning: An Multistrategy Approach*, Vol. 4, R.S. Michalski and G. Tecuci (Eds.), San Mateo, CA, Morgan Kaufmann, 1993.

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

Mitchell T.M., Keller T., and Kedar-Cabelli S., Explanation-Based Generalization: A Unifying View, *Machine Learning*, Vol. 1, pp. 47-80, 1986.

Mooney R.J. and Ourston D., A Multistrategy Approach to Theory Refinement, in *Machine Learning: An Multistrategy Approach*, Vol. 4, R.S. Michalski and G. Tecuci (Eds.), San Mateo, CA, Morgan Kaufmann, 1993.

Pople H.E., On the Mechanization of Abductive Logic, *Proc. IJCAI*, Stanford, CA, William Kaufmann, Inc., pp. 147-152, 1973.

Peirce C. S., Elements of Logic, in *Collected Papers of Charles Sanders Peirce (1839-1914)*, Ch. Hartshorne and P. Weiss (Eds.), The Belknap Press Harvard University Press, Cambridge, MA, 1965.

Quinlan J. R., Induction of Decision Trees, *Machine Learning* 1, pp. 81-106, 1986.

De Raedt L., Interactive Concept Learning, Ph.D. Thesis, Catholic Univ. Leuven, 1991.

De Raedt L. and Bruynooghe M., Interactive Theory Revision, in *Machine Learning: An Multistrategy Approach*, Vol. 4, R.S. Michalski and G. Tecuci (Eds.), San Mateo, CA, Morgan Kaufmann, 1993.

Richards B.L. and Mooney R.J., First-Order Theory Revision, to appear, 1993.

Russell S.J., *The Use of Knowledge in Analogy and Induction*, Pitman, 1989.

Shaw M.L.G. and Gaines B.R., An interactive knowledge elicitation technique using personal construct technology, in *Knowledge Acquisiion for Expert Systems: A Practical Handbook*, A.L. Kidd (Ed.), Plenum Press, 1987.

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

Tecuci G., Automating Knowledge Acquisition as Extending, Updating, and Improving a Knowledge Base, *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 22, No. 6, 1992.

Tecuci G., Plausible Justification Trees: a Framework for the Deep and Dynamic Integration of Learning Strategies, *Machine Learning*, Vol.4&5, 1993.

Tecuci G. and Hieb M., Consistency-driven Knowledge Elicitation: Using a Machine Learning Oriented Knowledge Representation to Integrate Learning and Knowledge Elicitation in NeoDISCIPLE, *Knowledge Acquisition Journal*, to appear 1993.

Tecuci G. and Kodratoff Y., Apprenticeship Learning in Imperfect Theory Domains, in *Machine Learning: An Artificial Intelligence Approach*, Vol. 3, Y. Kodratoff and R.S. Michalski (Eds.), San Mateo, CA, Morgan Kaufmann, 1990.

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

Widmer, G., A Tight Integration of Deductive and Inductive Learning, in *Proc. of the Sixth Int. Workshop on Machine Learning*, Ithaca, NY: Morgan Kaufman; 1989.

Winston P.H., Learning and Reasoning by Analogy, *Communications of the ACM, Vol.* 23, 1980.

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