Knowledge Acquisition (1994) **6**, 137–162

A framework for knowledge base refinement through multistrategy learning and knowledge acquisition

GHEORGHE TECUCI AND DAVID DUFF†

Department of Computer Science, George Mason University, Fairfax, VA 22030, USA and Romanian Academy Center for Artificial Intelligence, Bucharest, Romania, and †The MITRE Corporation, AI Technical Center, McLean, VA 22102, USA and Department of Computer Science, George Mason University, Fairfax, VA 22030, USA.

This paper presents a general approach to knowledge base refinement which integrates multistrategy learning, active experimentation and guided knowledge elicitation. Three main features characterize this approach. First, knowledge base refinement is based on a multistrategy learning method that dynamically integrates the elementary inferences (such as deduction, analogy, abduction, generalization, specialization, abstraction and concretion) that are employed by the single-strategy learning methods. Second, much of the knowledge needed by the system to refine its knowledge base is generated by the system itself. Therefore, most of the time, the human expert will need only to confirm or reject system-generated hypotheses. Third, the knowledge base refinement process is efficient due to the ability of the multistrategy learner to reuse its reasoning process. The paper illustrates a cooperation between a learning system and a 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 and relatively easy for an expert.

1. Introduction

Manual knowledge acquisition from experts and automated learning from data represent two extremes on a continuum of approaches to building a knowledge base. Moreover, these approaches are complementary in the sense that many of the tasks that are difficult for the human are relatively easy for a learning system and vice versa. Therefore, an automated knowledge acquisition methodology based on close cooperation between a human expert and a learning system appears to be a more natural approach to the problem of building knowledge-based systems (Tecuci, 1992). This explains the current growing interest in developing methods for integrating knowledge acquisition and machine learning (Birnbaum & Collins, 1991; Buchanan & Wilkins, 1993; Tecuci, Kedar & Kodratoff, 1993).

One could distinguish three main stages in the process of building a knowledge base for a general inference engine (Bareiss, Porter & Murray, 1989; Tecuci, 1992), as shown in Figure 1:

- systematic elicitation of expert knowledge
- knowledge base refinement
- knowledge base reformulation.

During systematic elicitation, one develops a preliminary conceptual model of the final system to be built, generally consisting of the inference engine and an

1042-8143/94/020137 + 25$08.00/0

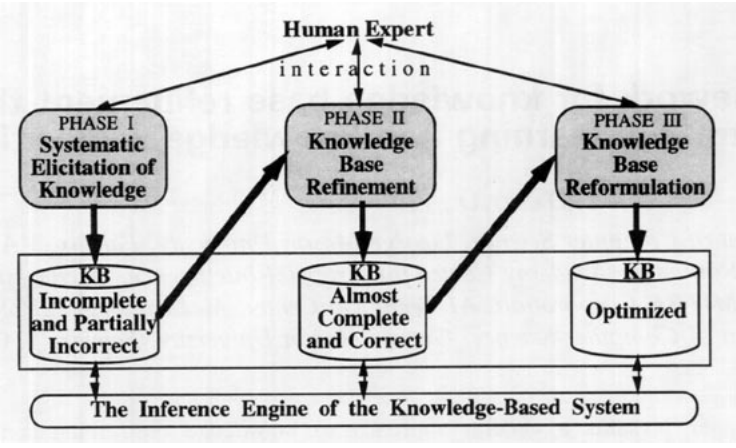


FIGURE 1. Expert system building as a three phase process.

incomplete and partially incorrect knowledge base (KB). During this stage, the expert should define the basic terminology of the KB, its conceptual structure, and any other domain knowledge which he/she can express. The most appropriate methods for this stage are those developed in the mainstream research on knowledge acquisition, such as interviewing (Gammack, 1987; LaFrance, 1987), protocol analysis (Ericsson & Simon, 1984), entity–attribute grids (Shaw & Gaines, 1987; Boose & Bradshaw, 1988) or domain modeling (Wielinga, Schreiber & Breuker, 1992; Morik, Wrobel, Kietz & Emde, 1993).

In the knowledge base refinement phase, the KB is extended and debugged. In general, the human expert uses the system to solve typical problems for which the solutions are already known in order to identify the need for additional knowledge (e.g. when the system cannot solve a problem), or errors in the KB (e.g. when the system proposes an incorrect solution to some problem). Then he/she, in cooperation with the knowledge base refinement module, extends or corrects the KB, accordingly. The result of the knowledge refinement phase should be a KB that is complete and correct enough for providing correct solutions to most of the problems to be solved by the system. A significant amount of work on knowledge refinement has been done in machine learning under the name of theory revision. Most often, the problem addressed is to refine the KB of a classification system so as to classify correctly a given set of positive and negative examples. Some representative theory revision systems are ANA-EBL (Cohen, 1991), EITHER (Mooney & Ourston, 1994) and PTR (Feldman, Koppel & Segre, 1994).

There has already been a significant amount of work on the integration of knowledge acquisition and machine learning methods for KB refinement. Examples of such integrated approaches are learning apprentice systems (Mitchell, Mahadevan & Steinberg, 1985; Porter, Bareiss & Holte, 1990; Tecuci & Kodratoff, 1990; Wilkins, 1990; De Raedt & Bruynooghe, 1994), interactive learners (Sammut & Banerji, 1986; Muggleton & Buntine, 1988), semi-automated knowledge refinement systems (Ginsberg, Weiss & Politakis, 1988; Baudin, Kedar & Pell 1994; Gil & Paris, 1994) and goal-driven knowledge elicitation systems (Wrobel, 1989; Tecuci & Hieb, 1994).

During the knowledge reformulation phase, the KB is reorganized for improving the efficiency in problem solving. In this phase, the machine learning methods like explanation-based learning (Mitchell, Keller & Kedar-Cabelli, 1986; DeJong & Mooney, 1986), knowledge compilation or macro operator learning (Fikes, Hart & Nilsson, 1972; Minton, 1990) will play a major role.

The boundaries between these three phases cannot be very clear. However, it is useful to distinguish them because, as we have already mentioned, each is characterized by specific goals and techniques.

In this paper we present a general approach to knowledge base refinement through multistrategy learning and knowledge acquisition. We first present some of the main ideas of this approach and then we detail it and illustrate its application to the refinement of the KB of a question-answering system in the domain of workstation allocation and configuration.

2. A framework for KB refinement

2.1. THE DEDUCTIVE CLOSURE OF THE KB

Let us consider the preliminary expert system which is the result of the systematic elicitation of knowledge (see Figure 1). It generally consists of a deductive inference engine and an incomplete and partially incorrect KB. 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, the KB contains facts and rules, and DC represents the set of statements which can be deductively derived from the KB:

$$DC = \{I : KB \vDash I\}$$

where "$\vDash$" means deductive entailment.

In order to simplify the presentation of our approach to KB refinement, we will further consider the case of a theorem-prover-based question answering system. We stress, however, that our approach is generally applicable to other types of knowledge-based systems.

Figure 2 shows the relationship between the deductive closure DC of an imperfect KB and the set of true statements in the application domain (TS):

- DC and TS are both considered to be "crisp" sets, with cleanly defined borders. That is, the system has an algorithm for testing the membership of a statement

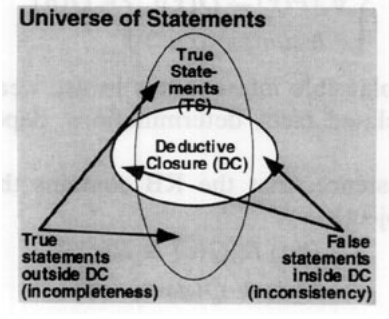


FIGURE 2. The relationship between DC and TS.

in DC, and presumably a human expert can perform a similar test on TS. We are therefore not considering the application domains in which an expert can characterize the truthfulness of a statement only to a certain degree.

- $DC \cap TS$ represents the set of statements 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 from the KB.
- $DC - TS$ represents the set of statements which are deductively entailed by the KB but are false. This shows that there are errors in the set of facts and deductive rules from the KB.
- $TS - DC$ represents the set of statements which are true but are not deductively entailed by the KB. This shows that the set of facts and deductive rules from the KB is incomplete.

2.2. THE PLAUSIBLE CLOSURE OF THE KB

Just as the deductive closure of the KB is the set of statements or problems that can be derived or solved using a deductive inference engine, the *plausible closure* (PC) is the set of statements or problems that can be derived or solved using a *plausible inference engine*:

$$PC = \{I : KB \models I\}$$

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 $Q(a)$:

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

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

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

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

$$P(c) \;\&\; Q(c) \;\&\; R(c)$$
$$P(d) \;\&\; Q(d) \;\&\; S(d)$$
$$P(e) \;\&\; Q(e)$$

By noticing that each time $P(x)$ is true, $Q(x)$ is also true, one may empirically induce the rule

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

and deductively apply it with the fact $P(a)$ to predict that $Q(a)$ is also true (Mitchell, 1978; Michalski, 1983; Quinlan, 1986). We call this type of inference inductive prediction.

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:

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

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 $KB \models Q(b)$, that is:

$$KB \vdash P(a)$$

$$KB \nvdash P(b)$$

$$KB \nvdash \neg P(b)$$

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

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 statement, I, is plausibly entailed by the KB, a system is not restricted to making only one plausible inference. In general, it can 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. (see Figure 3).

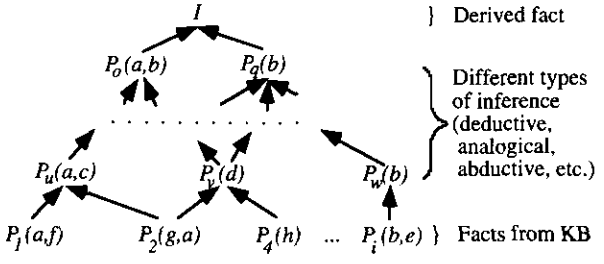


FIGURE 3. A plausible justification tree for the fact I.

For example, the inference, $P_1(a, f)$ & $P_2(g, a) \rightarrow P_u(a, c)$, might have been obtained by deduction, $P_2(g, a)$ & $P_4(h) \rightarrow P_v(d)$, might have been obtained by analogy, $P_i(b, e) \rightarrow P_w(b)$, might have been obtained by abduction, and, $P_o(a, b)$ & $P_q(b) \rightarrow I$, might have been obtained by inductive prediction.

Thus, the tree in Figure 3 shows that I is a plausible consequence of facts that are explicitly represented in the KB.

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

- *PC* is a "soft" set, the boundaries of which are not strictly defined because the statements inferred through plausible inference may have different degrees of certainty
- $DC \subset PC$ because a deductive entailment is a special case of a plausible entailment
- $PC \cap TS$ represents the set of true statements which are plausibly entailed by the KB
- $PC \cap TS - DC$ represents the set of true statements which are plausibly entailed by the KB, but are not deductively entailed by it: our hypothesis is that this set is quite large
- $TS - PC$ represents the set of true statements that are not plausibly entailed by the KB: although this set is not well defined, it expresses the intuition that there are true statements which even a plausible inference engine could not derive from the current KB.

2.3. KB REFINEMENT AS HEURISTIC SEARCH IN A PLAUSIBLE VERSION SPACE

As can be seen from Figure 1, the result of Knowledge Base Refinement should be an almost complete and correct KB. The expert system consisting of this KB and a deductive inference engine should be able to correctly solve most of the problems from its domain of expertise. This goal is achieved if the KB is extended and corrected so that its deductive closure *DC* becomes a good approximation of *TS*.

Figure 4 shows the relationship between *DC*, *PC* and *TS*, corresponding to the

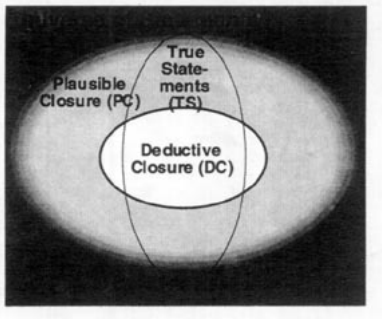


FIGURE 4. The relationship between *DC*, *PC* and *TS*.

initial KB to be refined. The deductive closure *DC* is an approximate lower bound for *TS* in the sense that most of *DC* is included in *TS*. Also, the plausible closure *PC* is an approximate upper bound for *TS* in the sense that most of *TS* is included in *PC*. The set *TS* is not known to the system (otherwise its knowledge would be correct and complete). However, any set *X* which includes most of *DC* and most of which is included in *PC* is a hypothesis for being the set *TS*. We can therefore consider *PC* and *DC* as defining a plausible version space (Tecuci, 1988; Tecuci, 1992) which includes the sets that are candidates for being the set *TS*. The concept of plausible version space is an extension of the concept of version space introduced by Mitchell (1978).

With this interpretation, the KB refinement problem reduces to one of searching the set *TS* in the plausible version space defined by *DC* and *PC*. Because the goal is to refine the KB in such a way that *DC* becomes a good approximation of *TS*, the deductive closure *DC* is also used as the current hypothesis for *TS*, and *PC* is used as a guide for extending *DC* so as to include more of *TS*. More precisely, during KB refinement, *DC* will be extended with a significant portion of $PC \cap TS$, and will also be corrected to remove from it most of $DC - TS$.

3. General presentation of the KB refinement problem and method

3.1. THE KB REFINEMENT PROBLEM

We are assuming that the KB to be refined is incomplete and partially incorrect, and consists of facts and rules expressed in first-order logic. However, the rules are not restricted to being deductive. They might also be weaker correlations such as determinations or mutual dependencies. This allows the introduction of all kinds of relevant knowledge into the KB, during the phase of systematic elicitation of knowledge (see Figure 1), as well as for performing plausible reasoning, during the KB refinement phase.

The KB is extended and corrected during training sessions with a human expert who provides the system with representative examples of problems and their correct solutions. Each such example *I* will initiate a knowledge base refinement session in which the system will extend and correct its knowledge base so as to correctly solve the problems from the class represented by the training example *I*.

In the case of a theorem proving-based question answering system, the input *I* is an example of a typical answer which the system should be able to give.

Referring back to Figure 4, one can see that the input *I* could be in one of three categories with respect to the current deductive and plausible closures of the KB (*DC* and *PC*):

- *I* is in *PC* (but not in *DC*)
- *I* is in *DC*
- *I* is not in *PC* or *DC*.

If the input *I* is in *PC* (but not in *DC*), then the system will be able to make a significant transfer of knowledge from the plausible closure to the deductive

closure, in the sense that statements that were only plausibly entailed by the KB become deductively entailed. More precisely, the system:

- will extend the KB so as to deductively entail new statements which are similar to I (all these similar statements being instances of a generalization I_g of I), reducing the size of the set $TS - DC$ (i.e. reducing the incompleteness of the KB)
- will correct the KB so as no longer to deductively entail false statements which are similar to I, reducing the size of the set $DC - TS$ (i.e. reducing the inconsistency of the KB)
- will extend the plausible closure of the KB, so as to include more of TS.

If I is in DC, then the system will reinforce some of the rules in DC.

If I is not in PC (or DC), 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 the KB may make it possible for the system to demonstrate that other statements (e.g. I_1, I_2) are deductively or plausibly entailed by the KB:

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

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

3.2. THE KB REFINEMENT METHOD

KB refinement is initiated by the expert providing the system with a true statement, I, which is an example of a statement which the system should be able to generate. That is, I should be in the deductive closure of the system after the knowledge refinement process. We will assume that I is in the current plausible closure.

The main stages of the KB refinement process are presented in Figure 5. They

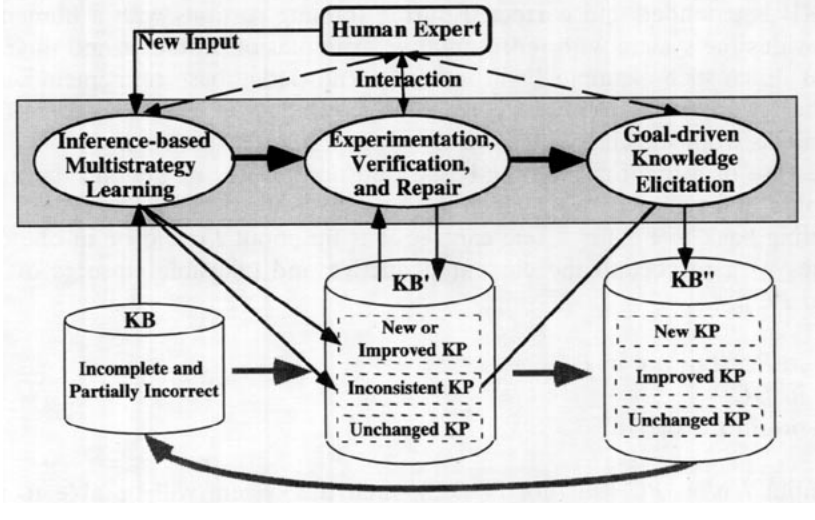


FIGURE 5. The main stages of the KB refinement process.

are:

- inference-based multistrategy learning
- experimentation, verification and repair
- goal-driven knowledge elicitation.

In the first stage, the system analyses the input in terms of its current knowledge by building "the most plausible and simple" justification tree (Tecuci, 1993) which demonstrates that the input is a plausible consequence of the system's current knowledge (see Figure 3). A concrete example of such a tree is presented in Figure 7. Because this "most plausible and simple" justification tree connects true facts from the KB with the true input *I*, the component implications are most likely to also be true. By making the hypothesis that these implications are indeed true, the system makes several extensions to the KB.

During the second stage of experimentation, verification and repair, the system reuses the reasoning process from the previous stage and refines the KB so as to deductively entail new true statements which are similar to *I*, and to no longer deductively entail false statements which are similar to *I*.

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

The knowledge refinement process will end when the system has been trained with examples of the main classes of statements it should be able to generate during its normal use. This should allow the system to derive most of the statements through deductive reasoning. However, it will also be able to derive unanticipated statements through plausible reasoning, and to learn from this experience, in a similar way it learned from the expert. Therefore, a system developed following this knowledge refinement method will have the capability of continuously improving itself during its normal use.

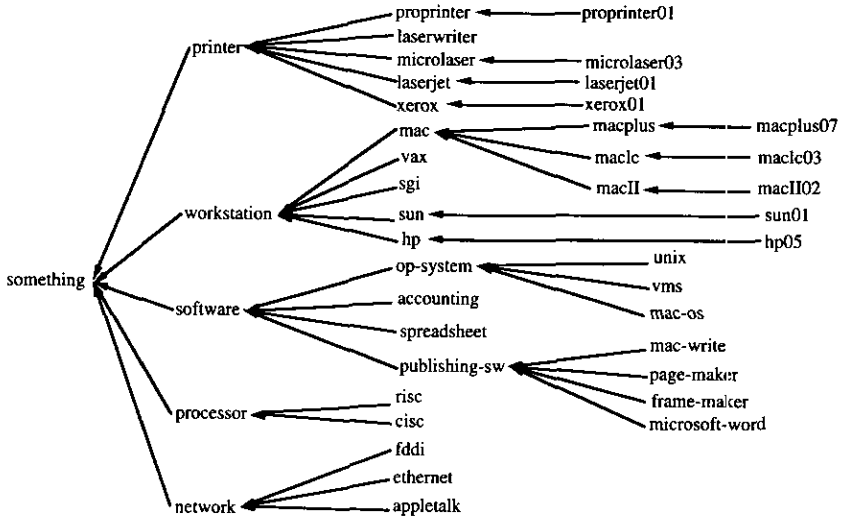
The entire KB refinement process is characterized by a cooperation between the learning system and the human expert in which the learner performs most of the tasks and the expert helps it in solving the problems that are intrinsically difficult for a learner and relatively easy for the expert.

4. Exemplary application domain

We will use the domain of workstation allocation and configuration in order to present 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 KB contains information about various printers and workstations distributed throughout the workplace.

In this example we will use the clausal form of logic. Therefore, instead of writing "A & B→C", which is read "if A and B are true then C is true" we will write the equivalent expression "C:−A, B.", which is read "C is true if A and B are true".

```
os(sun01, unix).  on(sun01, fddi).  speed(sun01, high).  processor(sun01, risc).  display(sun01, large)
; sun01's operating system is unix, it is on the fddi network, has high speed, risc processor and large display
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(proprinter01, 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).
```




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

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

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

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

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

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

FIGURE 6. Part of the incomplete and partially incorrect KB for the domain of workstation allocation and configuration (the arrows represent isa relationships).

A sample of the KB of the considered application domain is presented in Figure 6. It contains different types of knowledge: facts, a hierarchy of object concepts, deductive rules and a plausible determination rule.

The individual facts express properties of the objects from the application domain, or relationships between these objects. For instance, "display(sun01, large)" states that the display of "sun01" is large, and "os(sun01, unix)" states that the operating system of "sun01" is "unix".

The hierarchy of concepts represents the generalization (or "isa") relationships

between different object concepts. For instance, "microlaser03" is a "microlaser" which, in turn, is a "printer". These relationships can also be represented by "isa(microlaser03, microlaser)" and "isa(microlaser, printer)".

The top of the generalization hierarchy is "something", which represents the most general object concept from the application domain. It has the property that "isa(x, something)" is true for any x.

An example of a deductive rule is the following one:

$$\text{suitable(X, publishing)} :- \text{runs(X, publishing} - \text{sw)}, \text{communicate(X, Y)}, \text{isa(Y, high-quality-printer)}.$$

It states that X is suitable for desk top publishing if it runs publishing software and communicates with a high quality printer.

The KB in Figure 6 contains also the following plausible determination rule which states that the type of software run by a machine is plausibly determined by its operating system (":~" means plausible determination):

$$\text{runs(X, Y)} :\sim \text{os(X, Z)}$$

We are assuming that the facts from the KB are correct, but all the other knowledge pieces might be incomplete and/or partially incorrect.

After defining an initial knowledge base like the one from Figure 6, the human expert may start training the system by providing typical examples of answers which the system should be able to give by itself. For instance, the expert may tell the system that "macII02" is suitable for publishing:

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

This statement is representative of the type of answers which the system should be able to provide. This means that the final system should be able to give other answers of the form "suitable(X, Y)", where X is a workstation and Y is a task.

Starting from this input provided by the expert, the system will refine its knowledge base to that the deductive closure DC will include other true statements of the form "suitable(X, Y)" and, in the same time, will no longer include false statements of the same form.

The next sections will illustrate this knowledge refinement method, following the stages indicated in Figure 5.

5. Inference-based multistrategy learning

5.1. INPUT UNDERSTANDING

Let us suppose that the expert wants to teach the system which workstations are suitable for desk top publishing. To this purpose, it will give the system an example of such a workstation:

"suitable(macII02, publishing)"

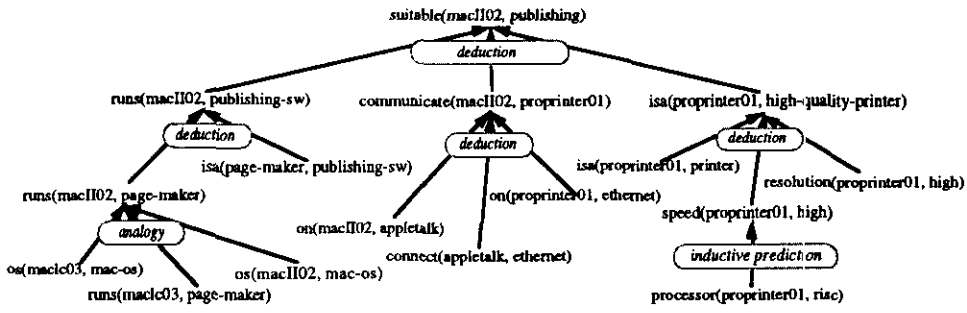


FIGURE 7. A plausible justification tree for "suitable(macII02, publishing)".

First, the system tries to "understand" the input in terms of its current knowledge by building the plausible justification tree in Figure 7. This tree demonstrates that the input is a plausible consequence of the current knowledge of the system. The method for building such a tree has previously been presented by Tecuci (1993). It employs a backward chaining uniform-cost search.

The tree in Figure 7 is composed of four deductive implications, a determination-based analogical implication and an inductive prediction.

The analogical implication was made by using the plausible determination rule

$$\text{runs}(X, Y) \text{:}\sim \text{os}(X, Z)$$

as indicated in Figure 8. According to this rule, 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".

The inductive prediction from Figure 7 was made by empirically generalizing the three sets of facts

(1) speed(sun01, high), os(sun01, unix), on(sun01, fddi), processor(sun01, risc), display(sun01, large)
(2) speed(hp05, high) os(hp05, unix), on(hp05, ethernet), processor(hp05, risc), runs(hp05, frame-maker)
(3) speed(xerox01, high), resolution(xerox01, high), processor(xerox01, risc)

to the rule

speed(x, high):—processor(x, risc). ;the speed of x is high if its processor is risc and then applying this rule deductively.

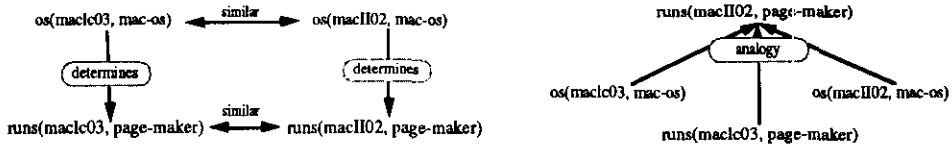


FIGURE 8. Inferring "runs(macII02, page-maker)" by analogy.

An algorithm for collecting the sets of facts to be generalized is described in (Lee and Tecuci, 1993). It starts with the fact "speed(proprinter01, high)" and recursively looks for other facts that share with it the predicate name or one of its arguments (e.g. "speed(sun01, high)", "os(sun01, unix)").

6.2. IMPROVEMENT OF THE KB

While there may be several justification trees for a given input, the attempt is to find the most simple and the most plausible one (Tecuci, 1993). This tree shows how a true statement I derives from other true statements from the KB. Based on Occam's razor (Blumer, Ehrenfeucht, Haussler & Warmuth, 1987), and on the general hypothesis used in abduction which states that the best explanation of a true statement 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 extended 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.

- learning a new fact by analogy:

  runs(macII02, page-maker).

- discovering positive examples of the determination rule, which is therefore reinforced:

  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 of the four 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 indicating a true statement allows the system to refine the KB by making several justified hypotheses.

As a result of these improvements, the KB entails deductively "suitable(macII02, publishing)", while before this statement was only plausibly entailed.

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.

5.3. REUSING THE REASONING PROCESS

As mentioned before, the input statement "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 statements, but also to improve *DC* so as no longer to include false statements of the same form.

Our method is based on the following general explanation-based approach to speed-up learning (Mitchell, Keller, Kedar-Cabelli, 1986; DeJong & Mooney, 1986). 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 this generalization.

The problem solved was the refinement of the KB so as to deductively entail the statement "suitable(macII02, publishing)". This was achieved through a complex multitype inference process of building the plausible justification tree in Figure 7. This reasoning process is generalized by employing various types of generalization procedures (see Section 5.4), and then (during the experimentation phase) it is instantiated to various plausible justification trees which show how statements similar to "suitable(macII02, publishing)" are entailed by the KB. Each such plausible justification tree is then used to extend or correct the KB.

5.4. MULTITYPE GENERALIZATION

The plausible justification tree in Figure 7 is generalized by generalizing each implication and by globally unifying all these generalizations. The generalization of an implication depends of the type of inference made, as shown in (Tecuci, 1993), the system employing different types of generalizations (e.g. deductive generalizations, empirical inductive generalizations, generalizations based on different types of analogies, and possibly, even generalizations based on abduction). To illustrate this, let us consider again the analogical implication from Figure 8. One could notice that the same kind of reasoning is valid for any machines X1 and X2, as long as they have the same type of operating system Z1, and X1 runs the software Y1, as illustrated in Figure 9.

Therefore, if one knows, for instance, that "os(hp05, unix)", "os(sun01, unix)" and "runs(hp05, frame-maker)", then one may immediately infer "runs(sun01, frame-maker)", by simply instantiating the general inference from the right side of

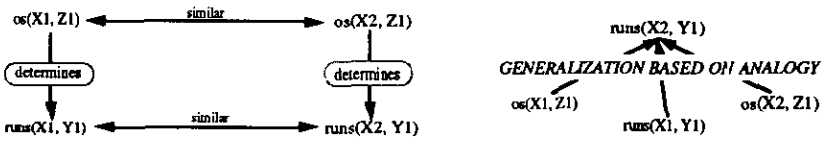


FIGURE 9. Generalization of the reasoning illustrated in Figure 8.

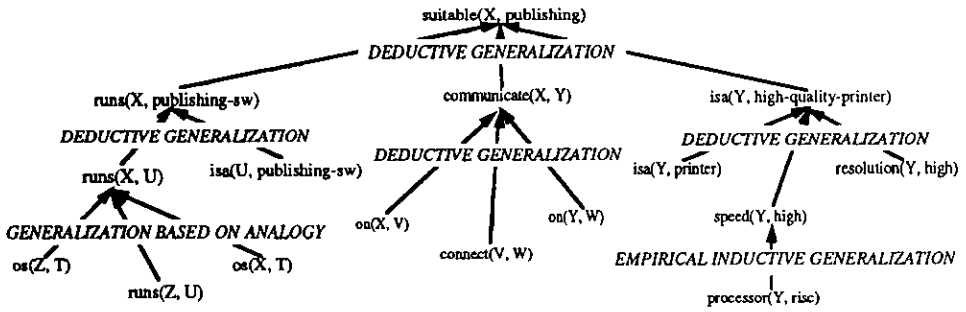


FIGURE 10. A generalized plausible justification tree.

Figure 9. This might not appear to be a significant saving but, in the case of a plausible justification tree, one generalizes several such individual implications and, even more importantly, their interconnection in a plausible justification tree. For instance, the generalization of the tree in Figure 7 is presented in Figure 10. The generalization method has previously been presented in detail by Tecuci (1993).

The important feature of the general tree in Figure 10 is that it covers many of the plausible justification trees for statements of the form "suitable(x, publishing)". If, for instance, "sun01" is another computer for which the leaves of the plausible justification tree in Figure 10 are true, then the system will infer that "sun01" is also suitable for publishing, by simply instantiating the tree in Figure 10. The corresponding instance is the tree in Figure 11.

There are several things to notice when comparing the tree in Figure 7 with the tree in Figure 11:

- although the structure of these trees and the corresponding predicates are identical (both being instances of the general tree in Figure 10), the arguments of the predicates are different and therefore the meaning of the trees is different
- while the tree in Figure 7 was generated through a complex reasoning process, the generation of the tree in Figure 11 was a simple matching and instantiation process
- based on each of these trees the KB is improved in a similar way so as to

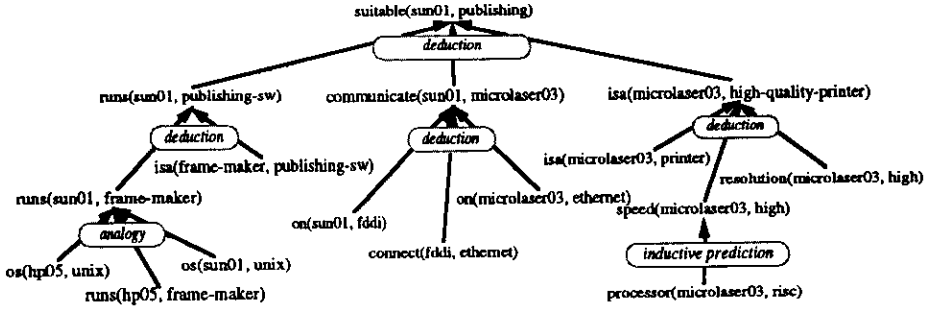


FIGURE 11. An instance of the plausible justification tree in Figure 10, justifying that sun01 is suitable for publishing.

deductively entail the statement from the top of the tree (assuming that "suitable(sun01, publishing)" is also true).

6. Experimentation, verification and repair

Building the plausible justification tree from Figure 7 and its generalization from Figure 10 was the first stage of the KB refinement process described in Figure 5. The next stage is one of experimentation, verification, and repair. In this stage, the system will generate plausible justification trees like the one in Figure 11, by matching the leaves of the tree in Figure 10, with the facts from the KB. 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.

6.1. CONTROL OF THE EXPERIMENTATION

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 (Tecuci, 1992). In the case of our example, the plausible version space is defined by the trees in Figure 7 and Figure 10, and is represented in Figure 12. The plausible upper bound is a rule the left hand side of which is the top of the general tree in Figure 10, 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 7. 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.

The plausible version space in Figure 12 synthesizes some of the inferential capabilities of the system with respect to the statements of the form "suitable(x, publishing)". This version space corresponds to the version space in Figure 4, as indicated in Figure 13. The set of instances of the plausible upper bound corresponds to *PC*, the set of instances of the plausible lower bound corresponds to *DC*, and the set of instances of the correct rule corresponds to *TS*.

The plausible version space in Figure 12 could be represented in the equivalent form from Figure 14. Note that the statements of the form "isa(Q, something)" are always true.

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 12. The plausible version space (*PVS*).

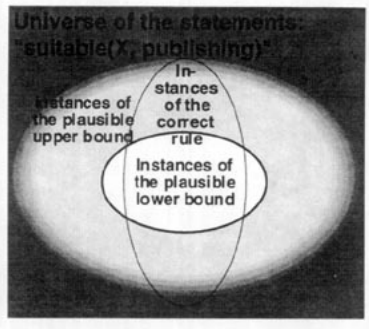


FIGURE 13. Plausible version space of a rule for inferring "suitable(X, publishing)".

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 14. Equivalent form of the plausible version space in Figure 12.

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

FIGURE 15. An instance of the upper bound of the plausible version space in Figure 14.

The version space in Figure 14 serves both for generating statements of the form "suitable(x, publishing)", and for determining the end of the experimentation phase.

To generate such a statement, 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 given in Figure 15.

Based on this instance, the system generates an instance of the general tree in Figure 10 which shows how "suitable(sun01, publishing)" is plausibly entailed by the KB (see Figure 11). The user is asked if "suitable(sun01, publishing)" is true or false, and the KB is updated according to the user's answer, as shown in the following sections.

During experimentation, the lower bound of the plausible version space in Figure 14 is generalized so as to cover the generated statements accepted by the user (the positive examples), and the upper bound (and, possibly, even the lower bound) is specialized so as to no longer cover the generated statements 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

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 16. Updated plausible version space (*PVS*).

- the bounds are not identical, but the KB no longer contains any instance of the upper bound which is not an instance of the lower bound. Therefore, no new statement of the form "suitable(x, publishing)" can be generated.

6.2. THE CASE OF A TRUE STATEMENT THAT IS PLAUSIBLY ENTAILED BY THE KB

Assuming the user accepted "suitable(sun01, publishing)" as a true statement, the KB and the plausible version space are updated as follows:

- the KB is extended 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 11.

The extensions of the KB are similar to those made for the input "suitable(macII02, publishing)" (see Section 5.2).

The plausible lower bound of the PVS is generalized as shown in Figure 16.

6.3. THE CASE OF A FALSE STATEMENT THAT IS PLAUSIBLY ENTAILED BY THE KB

Let us also consider the case of a generated statement which is rejected by the user: "suitable(macplus07, publishing)". The corresponding plausible justification tree is shown in Figure 17. This tree was obtained by instantiating the general tree in Figure 10 with facts from the KB. It shows how a false statement is plausibly

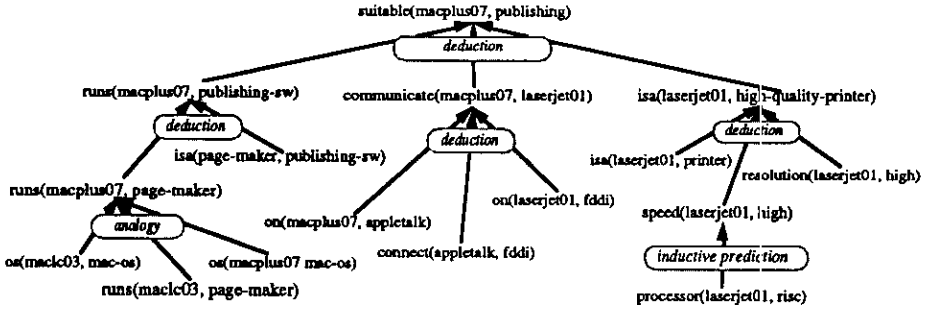


FIGURE 17. Another instance of the plausible justification tree in Figure 10, which shows how a false statement is plausibly entailed by the KB.

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

FIGURE 18. A wrong implication.

entailed by the KB. In such a case one has to detect the wrong implication(s) and to correct them, as well as to update the KB, the general justification tree in Figure 10, and the plausible version space in Figure 16 such that:

- the tree in Figure 17 is no longer a plausible justification tree
- the KB does not entail deductively "suitable(macplus07, publishing)"
- the updated general justification tree no longer covers the tree in Figure 17
- the plausible upper bound of the PVS is specialized so that it no longer covers the leaves of the tree in Figure 17.

6.3.1. Identification of the wrong implication

Detecting the wrong implication from the plausible justification tree in Figure 17 is an intrinsically difficult problem for an autonomous learning system. One possible solution, which has been presented by 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 17, the wrong implication could be identified by the user as being the deduction from the top of the tree, as indicated in Figure 18. 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.

6.3.2. Updating the KB, the plausible justification tree and the plausible version space

The rule which generated the wrong implication is specialized as indicated in Figure 19 (requiring X to have a large display).

The predicate "display(X, Y)" could be defined by the user, or could be suggested by the system as being a predicate which distinguishes the known positive examples of the rule from the discovered negative example (see section 7).

As a result of updating the above rule, the general plausible justification tree in

```
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.
```

FIGURE 19. Updated rule.

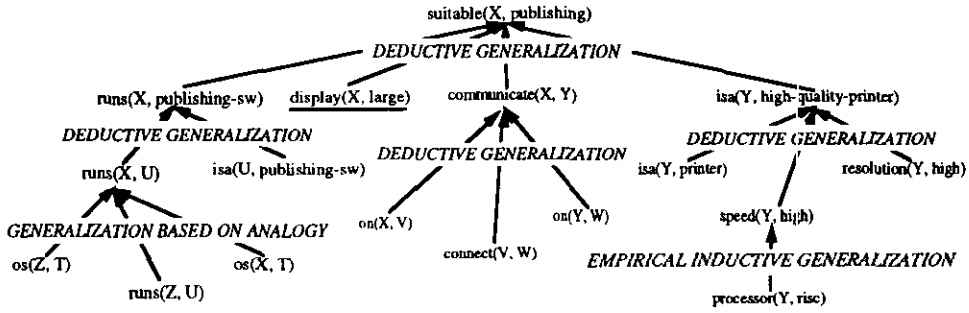


FIGURE 20. Updated general justification tree.

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 21. Updated plausible version space.

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 22. A rule with a negative exception.

Figure 10 is updated as shown in Figure 20, and the version space is updated as shown in Figure 21.

It might not always be easy to identify the problem with a wrong implication, and to specialize the corresponding rule so as no longer to cover the negative example. In such a case, the wrong implication is kept as a negative exception (also called false positive example) of the rule which generated it, as shown in Figure 22.

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 shown in Figure 22. 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 by Tecuci and Hieb (1994). These methods are applied in the third phase of KB refinement, as shown in Figure 5. They will be briefly described using the example of the inconsistent rule in Figure 22.

7.1. GOAL-DRIVEN PROPERTY ELICITATION

The first consistency-driven knowledge elicitation method is to look for a new predicate which could characterize all the positive instances of X ("macII02" and "sun01"), without characterizing any negative exception of X ("macplus07"). For instance:

- one might look for a property of "macII02" which could be a property of "sun01" (i.e. the KB does not deductively imply that "sun01" does not have this property), without being a property of "macplus07"
- or, one might look for a property of "sun01" which could be a property of "macII02", without being a property of "macplus07".

As can be seen in Figure 6, the knowledge base contains the predicate "display(sun01, large)", but nothing could be inferred about the display of "macII02" or about the display of "macplus07". Hence "display(X, large)" is a potentially discriminating predicate and the system will ask the user to confirm if it is true that the display of "macII02" is large and the display of "macplus07" is not large. If both these hypotheses are confirmed, then the system is able to update the rule in Figure 22 by adding the predicate "display(X, large)", as indicated in Figure 19. As one could notice, this is a form of property transfer from one object ("sun01") to another object ("macII02"), driven by the goal of removing the inconsistency in an inference rule.

Obviously, this knowledge elicitation method could be applied to any variable from the rule (i.e. also to Y).

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

7.2. GOAL-DRIVEN RELATIONSHIP ELICITATION

A second method for removing the negative exception from the rule in Figure 22 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. That is, the system will look in the KB for a relationship between "macII02" and "proprinter01" which could also hold between "sun01" and "microlaser03", without holding between "macplus07" and "laserjet01". Or, it could look for a relationship between "sun01" and "microlaser03" which could also hold between "macII02" and "proprinter01", without holding between "macplus07" and "laserjet01". Or it could try to elicit such a relationship from the expert.

7.3. GOAL-DRIVEN CONCEPT ELICITATION

The third method is similar to the one described in (Wrobel, 1989). It consists of trying to elicit a new concept that covers all the positive instances of X ("macII02" and "sun01") without covering the negative exception of X ("macplus07", in our example). Alternatively, one could try to elicit a concept that covers all the positive instances of Y without covering the negative exception of Y.

8. Conclusions

We have presented a general approach to knowledge refinement and we have illustrated it with a specific KB refinement method. This approach, which integrates knowledge acquisition and multistrategy learning, emerged from two related research directions:

- the knowledge acquisition methodology of DISCIPLE (Tecuci, 1988; Tecuci and Kodratoff, 1990) and NeoDISCIPLE (Tecuci, 1992)
- the multistrategy task-adaptive learning method based on plausible justification trees MTL-JT (Tecuci and Michalski, 1991; Tecuci, 1993).

On the one hand, it extends DISCIPLE and NeoDISCIPLE with respect to the knowledge representation used and the types of inference and generalization employed and, on the other hand, it adapts and integrates the MTL-JT method into an interactive knowledge acquisition scenario.

One important feature of the presented KB refinement method is the use of plausible justification trees which dynamically integrate the elementary inferences employed by the single-strategy learning methods, depending on the system's knowledge and the system's input. This allows the system to learn as much as possible from every input received from the human expert, generating by itself much of the knowledge needed to refine the KB. This is an important advantage over the standard knowledge acquisition methods in which the expert provides the needed knowledge to the system. It is also important to stress that the plausible justification tree itself represents a general framework for integrating a whole range of inference types (Tecuci, 1993). Therefore, theoretically, there is no limit with respect to the type or number of inferences employed in building such a tree.

Another important feature of the KB refinement method is the employment of different types of generalizations. The current machine-learning research distinguishes only between deductive generalizations and inductive generalizations. Our research shows that one could make much finer grade distinctions by associating a specific generalization with each type of inference as, for instance, generalization based on analogy. The generalization of an inference process synthesizes the assumptions that support that process, and may therefore be used in reusing the process in new situations. This feature is exploited by our method, which reuses its KB refinement processes, and is therefore very efficient.

Finally, the KB refinement method is based on a cooperation between a learning system and a 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 and relatively easy for an expert.

There are also several ways in which the presented KB refinement method could be improved. First of all, 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 inference should be included, as well as more complex versions of the current ones. Consequently, new types of justified generalizations, corresponding to these new types of inference, should also be defined.

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

Another future research direction consists in developing a qualitative representation of the certainty of the knowledge pieces from the KB (e.g. some statements are characterized as true by the expert, while others are hypothesized as true by the system, some rules are initially defined by the expert, while others are learned by the system). In particular, one has to be able to estimate the confidence in the learned rules, to update the confidence of a rule when new examples or exceptions are discovered, as well as to maintain only a limited number of "representative" examples of a rule (during knowledge refinement) so as neither to overload the system, nor to lose important information.

The authors are grateful to Michael Hieb, Smadar Kedar and Yves Kodratoff for their useful suggestions and criticisms. This research was conducted partly in the Center for Artificial Intelligence of George Mason University and partly in the Romanian Academy Center for Artificial Intelligence. The research was supported in part by the National Science Foundation Grant No. IRI-9020266, the Office of Naval Research grant No. N0014-91-J-1351, the Advanced Research Projects Agency grant No. N0014-91-J-1854, administered by the Office of Naval Research, and by the Romanian Academy.

References

BAREISS, E. R., PORTER, B. W. & MURRAY, K. S. (1989). Supporting start-to-finish development of knowledge bases. *Machine Learning*, **3**.

BAUDIN, C., KEDAR, S. & PELL, B. (1994). Increasing levels of assistance in refinement of knowledge-based retrieval systems, *Knowledge Acquisition*, **6**.

BIRNBAUM, L. A. & COLLINS, G. C., Eds. (1991). *Machine Learning: Proc. Eighth International Workshop (ML91), Part I: Automated Knowledge Acquisition.* San Mateo, CA: Morgan Kaufmann.

BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D. & WARMUTH, M. K. (1987). Occam's razor, *Information Processing Letters*, **24**, 377–380.

BOOSE, J. H. & BRADSHAW, J. M. (1988). Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems, In J. BOOSE & B. GAINES, Eds. *Knowledge Acquisition Tools for Expert Systems,* San Diego, CA: Academic Press.

BUCHANAN, B. G. & WILKINS, D. C. (1993). *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems.* San Mateo, CA: Morgan Kaufmann.

CARBONELL, J. G. (1986). Derivational analogy: a theory of reconstructive problem solving and expertise acquisition, In R. S. MICHALSKI, J. G. CARBONELL & T. M. MITCHELL, Eds. *Machine Learning: An Artificial Intelligence Approach.* Vol. 2. San Mateo, CA: Morgan Kaufmann.

COHEN, W. (1991). The generality of overgenerality, in L. A. BIRNBAUM & G. C. COLLINS, Eds. *Machine Learning: Proc. of the Eighth Int. Workshop,* Chicago: Morgan Kaufmann.

COLLINS, A. & MICHALSKI, R. S. (1989). The logic of plausible reasoning: a core theory, cognitive science, **13,** 1–49.

DAVIES, T. R. & RUSSELL, S. J. (1987). A logical approach to reasoning by analogy. In *Proc. IJCAI-87,* Milan, Italy: Morgan Kaufmann.

DEJONG, G. & MOONEY, R. (1986). Explanation-based learning: an alternative view, *Machine Learning,* **1,** 145–176.

DE RAEDT, L. (1991). Interactive concept learning, Ph.D. Thesis, Catholic University of Leuven.

DE RAEDT, L. & BRUYNOOGHE, M. (1994). Interactive theory revision, In R. S. MICHALSKI, & G. TECUCI, Eds. *Machine Learning: A Multistrategy Approach* Vol. 4, San Mateo, CA: Morgan Kaufmann.

ERICSSON, K. A. & SIMON, H. A. (1984). *Protocol Analysis: Verbal Reports as Data* Cambridge, MA: MIT Press.

FELDMAN, R., KOPPEL, M. & SEGRE, A. (1994). Extending the role of bias in probabilistic theory revision, *Knowledge Acquisition,* **6.**

FIKES, R. E., HART, P. E. & NILSSON, N. J. (1972). Learning and executing generalized robot plans, *Artificial Intelligence,* **3,** 251–288.

GAMMACK, J. G. (1987). Different techniques and different aspects on declarative knowledge, in A. L. KIDD, Ed. *Knowledge Acquisition for Expert Systems: A Practical Handbook* New York: Plenum Press.

GIL, Y. & PARIS, C. (1994). Towards method-independent knowledge acquisition, *Knowledge Acquisition,* **6.**

GINSBERG, A., WEISS, S. M. & POLITAKIS, P. (1988). Automatic knowledge base refinement for classification systems, *Artificial Intelligence* **35,** 197–226.

GENTNER, D. (1990). The mechanisms of analogical reasoning, in J. W. SHAVLIK & T. G. DIETTERICH, Eds *Readings in Machine Learning*, San Mateo, CA: Morgan Kaufmann.

GREINER, R. (1988). Learning by understanding analogies, in A. PRIEDITIS, Ed. *Analogica* London: Pitman.

JOSEPHSON, J. (1991). Abduction: conceptual analysis of a fundamental pattern of inference, Research Report 91-JJ, Laboratory for AI Research, Ohio State University.

KEDAR-CABELLI, S. (1988). Toward a computational model of purpose-directed analogy, In A. PRIEDITIS, Ed. *Analogica,* London: Pitman.

KODRATOFF, Y. (1990). Using abductive recovery of failed proofs for problem solving by analogy, in *Machine Learning: Proc. Seventh Int. Workshop,* B. W. PORTER & R. J. MOONEY, Eds. San Mateo, CA: Morgan Kaufmann.

LA FRANCE, M. (1987). The knowledge acquisition grid: a method for training knowledge engineers, *International Journal of Man–Machine Studies,* **27,** 245–255.

LEE, O. & TECUCI, G. (1993). MTLS: an inference-based multistrategy learning system, In *Proc. InfoScience 93,* Seoul, Korea: Korea Information Science Society.

MAHADEVAN, S. (1989). 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 Kaufmann.

MICHALSKI, R. S. (1983). A theory and methodology of inductive learning, in *Machine Learning: An Artificial Intelligence Approach,* Vol. 1, R. S. MICHALSKI, J. G. CARBONELL & T. M. MITCHELL, Eds. Los Altos, CA: Tioga Publishing Co.

MICHALSKI, R. S. (1994). Inferential learning theory: developing theoretical foundations for multistrategy learning, in R. S. MICHALSKI & G. TECUCI, Eds. *Machine Learning: An Multistrategy Approach,* Vol. 4, San Mateo, CA: Morgan Kaufmann.

MINTON, S. (1990). Quantitative results concerning the utility of explanation-based learning *Artificial Intelligence,* **42,** 363–392.

MITCHELL, T. M. (1978). Version spaces: an approach to concept learning, Doctoral dissertation, Stanford University.

MITCHELL, T. M., MAHADEVAN, S. & STEINBERG, L. I. (1985). LEAP: a learning apprentice

system for VLSI design, In *Proc. International Joint Conference on Artificial Intelligence (IJCAI-85)*, 573–580. Los Angeles: Morgan Kaufmann.

MITCHELL, T. M., KELLER, T. & KEDAR-CABELLI, S. (1986). Explanation-based generalization: a unifying view, *Machine Learning*, Vol. 1, 47–80.

MORIK, K., WROBEL, S., KIETZ, J. U. & EMDE, W. (1993). *Knowledge Acquisition and Machine Learning*. San Diego, CA: Academic Press.

MOONEY, R. J. & OURSTON, D. (1994). A multistrategy approach to theory refinement, In R. S. MICHALSKI & G. TECUCI, Eds *Machine Learning: An Multistrategy Approach*, Vol. 4, San Mateo, CA: Morgan Kaufmann.

MUGGLETON, S. & BUNTINE, W. (1988). Machine invention of first order predicates by inverting resolution, In *Proc. Fifth International Conference on Machine Learning*, 339–352. Ann Arbor, MI: Morgan Kaufmann.

POPLE, H. E. (1973). On the mechanization of abductive logic, *Proc. IJCAI-73*, 147–152. Stanford, CA: William Kaufmann.

PEIRCE, C. S. (1965). Elements of logic, In C. HARTSHORNE & P. WEISS, Eds. *Collected Papers of Charles Sanders Peirce (1839–1914)* Cambridge, MA: The Belknap Press Harvard University Press.

PORTER, B. W., BAREISS, R. & HOLTE, R. C. (1990). Concept learning and heuristic clasification in weak-theory domains, in J. SHAVLIK & T. DIETTERICH, Eds, *Readings in Machine Learning*, San Mateo, CA: Morgan Kaufmann.

QUINLAN, J. R. (1986). Induction of decision trees, *Machine Learning* **1**, 81–106.

RUSSELL, S. J. (1989). *The Use of Knowledge in Analogy and Induction*, London: Pitman.

SAMMUT, C. & BANERJI, R. B. (1986). Learning concepts by asking questions, In R. S. MICHALSKI, J. G. CARBONELL & T. M. MITCHELL, *Machine Learning: An Artificial Intelligence Approach*, Vol. 2, Morgan Kaufmann, pp. 167–191.

SHAW, M. L. G. & GAINES, B. R. (1987). 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 Press.

TECUCI, G. (1988). DISCIPLE: a theory, methodology and system for learning expert knowledge, Ph.D. Thesis, University of Paris—South.

TECUCI, G. (1992). Automating knowledge acquisition as extending, updating and improving a knowledge base, *IEEE Transactions on Systems, Man and Cybernetics*, **22**, 1444–1460.

TECUCI, G. (1993). Plausible justification trees: a framework for the deep and dynamic integration of learning strategies, *Machine Learning*, **11**, 237–261.

TECUCI, G. & HIEB, M. (1994). Consistency-driven knowledge elicitation: using a learning-oriented knowledge representation that supports knowledge elicitation in NeoDISCIPLE, *Knowledge Acquisition*, **6**, 23–46.

TECUCI, G. & KODRATOFF, Y. (1990). Apprenticeship learning in imperfect theory domains, In Y. KODRATOFF & R. S. MICHALSKI, Eds. *Machine Learning: An Artificial Intelligence Approach*, Vol. 3, San Mateo, CA: Morgan Kaufmann.

TECUCI, G. & MICHALSKI, R. S. (1991). A method for multistrategy task-adaptive learning based on plausible justifications, in L. BIRNBAUM & G. COLLINS, Eds *Machine Learning: Proc. Eighth International Workshop*, Chicago, IL: Morgan Kaufmann.

WIDMER, G. (1989). A tight integration of deductive and inductive learning, in *Proc. Sixth Int. Workshop on Machine Learning*, Ithaca, NY: Morgan Kaufmann.

WIELINGA, B. J., SCHREIBER, A. TH. & BREUKER, J. A. (1992). KADS: a modeling approach to knowledge-engineering, *Knowledge Acquisition*, **4**, 5–53.

WILKINS, D. C. (1990). Knowledge base refinement as improving an incorrect and incomplete domain theory, In Y. KODRATOFF & R. S. MICHALSKI, Eds *Machine Learning: An Artificial Intelligence Appraoch*, Vol. 3, pp. 493–513. Morgan Kaufmann.

WINSTON, P. H. (1980). Learning and reasoning by analogy, *Communications of the ACM*, **23**.

WROBEL, S. (1989). Demand-driven Concept Formation, In MORIK, K. Ed. *Knowledge Representation and Organization in Machine Learning*, Berlin: Springer Verlag.