International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.1, No.1 (2005), pp. 85-97
© Research India Publications http://www.ijcir.info

Learning Best Concept Approximations from Examples

Mihai Boicu[1] and Gheorghe Tecuci[2]

George Mason University,
Fairfax, VA 22030 USA
[1]E-mail: mboicu@gmu.edu [2]E-mail: tecuci@gmu.edu

Abstract: **This paper addresses the problem of learning the best approximation of a concept from examples, when the concept cannot be expressed in the learner's representation language. It presents a method that determines the version space of the best approximations and demonstrates that for any given approximation of the target concept there is a better approximation in this version space. The method does not depend on the order of examples and has an almost monotonic convergence. This method was developed for the Disciple learning agent that can be taught by a subject matter expert how to perform complex problem solving tasks.**

***KeyWords*:** Machine Learning, Concept Learning from Examples, Plausible Version Space, Ontology, Expert Systems.

I. Introduction

We are researching a general approach to the development of learning agents that can be taught directly by subject matter experts how to solve problems. This research resulted in the development of the Disciple theory, methodology and family of learning agent shells [1]-[3]. The expert teaches the Disciple agent in a way that is similar to how the expert would teach a person, by demonstrating and explaining the agent how to solve specific problems, and by supervising, correcting and explaining agent's errors.

The Disciple approach has already been applied to develop knowledge based agents in complex domains such as course of action critiquing [4], center of gravity determination [5] and intelligence analysis [6], proving to be a good solution to the knowledge acquisition bottleneck [7].

The problem-solving engine of a Disciple agent is based on the general task reduction paradigm [8]. In this paradigm, a problem solving task is successively reduced to simpler tasks. The solutions of the simplest tasks are found and then these solutions are successively combined until they produce the solution of the initial task. To exhibit such a behavior, the knowledge base of the Disciple agent is structured into an object ontology and a set of if-then problem solving rules. The object ontology is a hierarchical representation of the objects and types of objects from the application domain. It represents the different kinds of objects, the properties of each object, and the relationships existing between objects ([9] and [10]). A fragment of the ontology of an agent that assists a student to select a PhD advisor is presented in Fig.1.

The if-then problem solving rules are learned from the

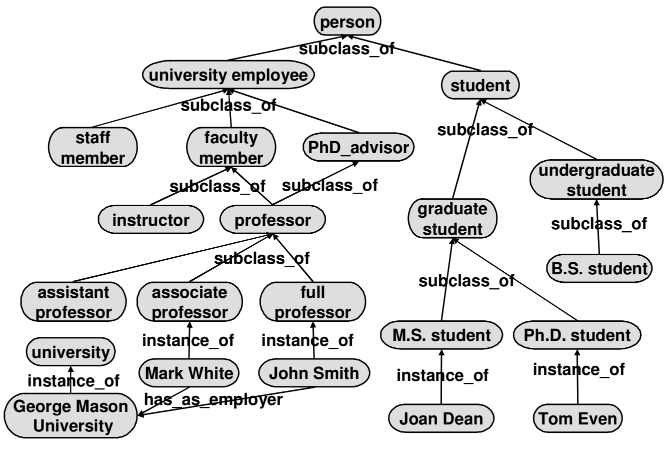

Figure 1. Ontology fragment.

expert's problem solving examples by using the object ontology as a generalization hierarchy ([11] and [12]).

To illustrate the rule learning process consider how an

Task: Determine whether John Smith can be a PhD advisor for Tom Evan in Artificial Intelligence.

Question: *Is John Smith likely to stay on the faculty of George Mason University for the duration of Tom Evan's dissertation?*
Answer: *Yes, because John Smith has a tenured position.*

Subtask: Determine whether John Smith would be a good PhD advisor for Tom Evan in Artificial Intelligence.

Figure 2. Task reduction example.

expert may teach a Disciple agent how to help a student select a PhD advisor. The expert formulates an initial problem solving task, such as "Determine a PhD advisor for

Tom Evan." Then, using the task reduction paradigm, the expert successively reduces this task to simpler tasks, guided by questions and answers, as illustrated by the task reduction step from Fig. 2.

From each such task reduction step, Disciple learns a general task reduction rule. The learned rules are used by Disciple in the task reduction process, and the critiques received from the expert guides their refinement. For instance, the rule learned from the above example is applied to "Determine whether Mark White can be a PhD advisor for Tom Evan in Artificial Intelligence." However, the corresponding reduction is rejected by the expert because "Mark White is likely to move to Stanford University." Consequently, Disciple adds an except-when condition to the rule which takes the form shown in Fig. 3. As illustrated in this figure, the rules learned by Disciple have an IF-THEN part that preserves the expert's language from the example, a main applicability condition and, optionally, several except-when conditions (which should not be satisfied to apply the rule). The conditions are partially learned, containing plausible upper and lower bounds for the acceptable values of the rule's variables. Rule's representative examples are also kept to help in its further refinement [13].

The applicability conditions of the learned rules aim to capture the complex concepts used by the subject matter expert. Often, these concepts are not expressible in the agent's representation language where the correct instance of a variable (e.g. ?O1 in Fig. 3) is specified as being an instance of a concept from the object ontology (e.g. PhD advisor in Fig.1 and Fig.3), and having certain features (e.g. has-as-employer ?O4, and has-as-position ?O5).

In this paper we describe an approach to capturing and representing the subtle and complex concepts of a subject matter expert. One idea of this approach is to significantly extend the generalization hierarchy and, implicitly, the representation language. Another idea is to develop methods for learning approximations of the concepts that cannot be expressed even in the extended representation language.

The rest of this paper is organized as follows. Section 2 defines the learning problem. Section 3 presents a refinement of the generalization hierarchy. Section 4 presents the best approximation of a target concept, including "the target's best approximation theorem." Then section 5 presents a method for learning the upper and lower approximations of a target concept, and demonstrates several convergence theorems. Section 6 discusses related research and future directions.

IF: Determine whether ?O1 can be a PhD advisor for ?O2 in ?O3

Question: *Is ?O1 likely to stay on the faculty of ?O4 for the duration of ?O2 's dissertation?*
Answer: *Yes, because ?O1 has a ?O5*

THEN: Determine whether ?O1 would be a good PhD advisor for ?O2 in ?O3

MAIN CONDITION

```
?O1  is    PUB (PhD_advisor)       PLB (PhD_advisor)
       has_as_employer ?O4
       has_as_position  ?O5
?O2 is   PUB (person)              PLB (PhD_student)
?O3 is   PUB (research_area)    PLB (Artificial_Intelligence)
?O4 is   PUB (employer)           PLB (university)
?O5 is   PUB (position)            PLB (tenured_position)
```

EXCEPT WHEN CONDITION

```
?O1  is   PUB (person)              PLB (PhD_advisor)
       is_likely_to_move_to ?O6
?O6 is   PUB (employer)           PLB (university)
```

Positive Example: (?O1=John_Smith ?O2=Tom_Evan ?O3=Artificial_Intelligence ?O4=George_Mason_University ?O5=tenured_position)
Negative Example: (?O1=Mark_White ?O2=Tom_Evan ?O3=Artificial_Intelligence ?O4=George_Mason_University ?O5=tenured_position ?O6=Stanford_University)

Figure 3. Partially learned plausible version space rule.

II. Learning Concept Approximations from Examples

The learning problem discussed in this paper is presented in Table 1. This learning problem is a basic component of the complex problem of learning problem solving rules (such as the one from Fig.3) from specific examples of problem solving steps (such as the one from Fig.2).

Table 1. Learning Concept Approximations From Examples.

Given:
- A sequence of positive and negative examples (instances) of a target concept C.
- A generalization hierarchy of concepts GH which may or may not include the target concept C.

Determine:
- Either the target concept C, which covers all the positive examples and none of the negative examples, if C is a concept from GH.
- Or the best approximations of the target concept C, expressed with concepts from GH, if C is not included in GH.

This problem is a generalization of the classical problem of incremental concept learning from examples [14], and reduces to it if the target concept C is included in the generalization hierarchy GH. However, the classical learning problem will fail when the target concept C is not part of GH. In such a case, the learning problem from Table 1 does not require the learning of the exact concept C, but its best approximations.To illustrate this learning problem, let us consider the generalization hierarchy from the top of Fig. 4. The sequence of positive and negative examples of the target concept is: I1(+), I2(+), I3(-), and I4 (+). None of the concepts from the generalization hierarchy in Fig. 4 covers all the positive examples, I1, I2, and I4, without covering the negative example I3. The bottom part of Fig. 4 shows two possible views of the same hierarchy, where the concepts are represented as sets and the instances as points. Because the target concept cannot be any of the already represented concepts, it must be a new one, such as those indicated by dashed lines in the figure. The one from the left hand side is a target that does not contain any concept from hierarchy. The one from the right hand side is a target concept that includes the concepts *C* and *D*.

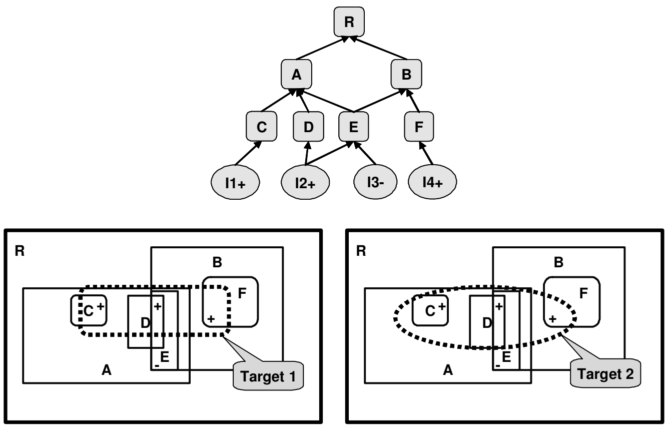

Figure 4. Two potential target concepts.

Fig. 5 shows how the two target concepts from the bottom of Fig. 4 can be introduced into the generalization hierarchy, based on their generalization relationships with the other concepts from the hierarchy. *T1* can be introduced under the root concept as a terminal concept. *T2* can be introduced as a subconcept of R and a superconcept of *C* and *D*.

Thus, another result of the learning problem in Table 1 is the potential extension of the generalization hierarchy with new concepts, such as *T1* or *T2*.

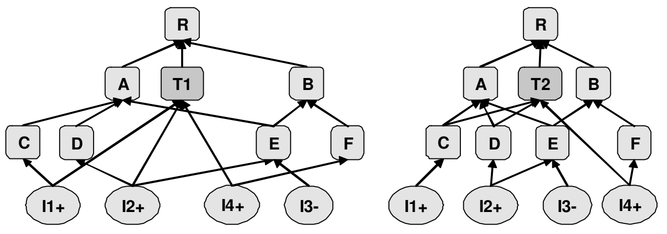

Figure 5. The generalization hierarchy with the target concept.

III. Refinement of a Generalization Hierarchy

A. Generalization hierarchy of concepts

In this section we give a formal definition for the generalization hierarchy of concepts, as a partially ordered set, having the generalization relation as the order relation.

Definition: A generalization hierarchy of concepts is a structure, *GHC(Concepts, root-concept, subconcept-of)* with the following properties:

- *Concepts* is a finite, non-empty set of distinct elements, named concepts. A concept might be viewed as representing a set of elements named instances. Examples of concepts from Fig. 1 are: *professor, student, person.*
- *root-concept* is an element from *Concepts*. We generally use the concept named *object* as the *root-concept*.
- *subconcept-of* is a binary, strict partial order relation on the set *Concepts* (denoted also by $\subset$) having *root-concept* as the maximum element (i.e. any other concept is a subconcept of *root-concept*). An example is: *student subconcept-of person* (see Fig. 1).

Definition: The relation direct-subconcept-of is defined by: c_1 direct-subconcept-of c_2 if and only if c_1 subconcept-of c_2 and there is no other concept c such that c_1 subconcept-of c and c subconcept-of c_2.

B. The concepts' algebra

We will extend the initial set of concepts by using the operations of union, intersection and negation, defined similarly with the set theory [3]. This extension will lead to a much more expressive language.

Definition: The set *E(Concepts)* denotes the extension of the set *Concepts* under the operations: $\cup$, $\cap$, and $\neg$.

Lemma: The extended relation *subconcept-of* $(\subset)$ is a partial order relation on *E(Concepts)* having *Root-Concept* as a maximum element and $\varnothing$ as the minimum element.

Lemma: $L_{GHC}(E(Concepts), \cup, \cap, \subset)$ is a complete lattice, having $sup\{x, y\}= x\cup y$ and $inf\{x, y\}=x\cap y$.

Lemma: $L_{GHC}(E(Concepts), \cup, \cap, \neg, \varnothing, Root\text{-}Concept)$ is a Boolean algebra.

Following the Boolean algebra theory [15], a minterm in the Boolean algebra L_{GHC} is a conjunction of concepts and negated concepts:

$$\bigcap_{t_C \text{ is } C \text{ or } \neg C,\, C \in Concepts} t_C$$

The minterms are the bricks from which all the elements of *E(Concepts)* are constructed. Any element from *E(Concepts)* can be written as a union of minterms. These minterms are disjoint. However, because of the existing partial order relation some of the minterms are always empty, as for instance, one in which both c_1 and $\neg c_2$ appear, and c_1 *subconcept-of* c_2.

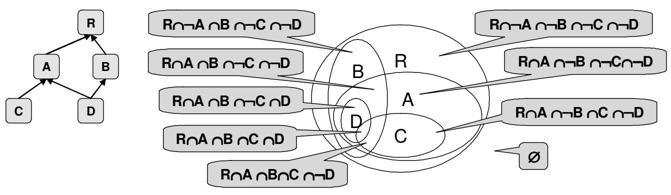

Figure 6. The minterms for a small hierarchy.

Fig. 6 shows all the minterms for a small hierarchy. Most of the minterms are empty, only 8 out of 32 having a meaningful interpretation, as the right hand side of the figure shows. Each minterm will represent one of the disjoint areas in the figure. If a concept appears in a non empty minterm non-negated, all its superconcepts must also appear non-negated. Therefore we may write $D \cap \neg C$ instead of $R \cap A \cap B \cap \neg C \cap D$. Moreover, we may denote this minterm with *minterm(D)*, or *mt(D)* with the convention that *D* and all its superconcepts appear in the minterm non-negated, and all the other concepts appear negated.

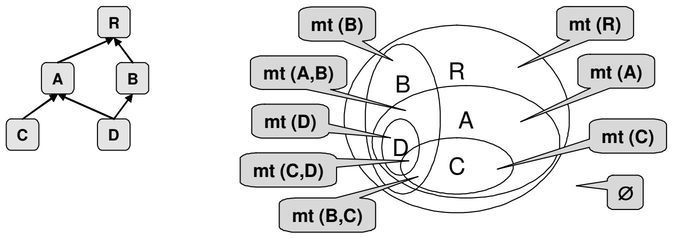

Figure 7. The minterms from Fig. 6 in short notation.

Definition: For a list of concepts c_1, c_2... c_n that are not hierarchically connected (i.e. none of them is a subconcept of another) we denote with $mt(c_1, c_2... c_n)$ the minterm for which these concepts and all their superconcepts appear non-negated, and all the other concepts appear negated.

With this convention the minterms from Fig. 6 may be rewritten as in Fig. 7.

The semantics of concepts may further reduce the number of non-empty minterms. Consider, for example, *assistant-professor, associate-professor* $\in$ *Concepts*. Based on their semantics we know that *assistant-professor* $\cap$ *associate-professor* = $\varnothing$. All the minterms containing both *assistant-professor* and *associate-professor* not negated will be empty.

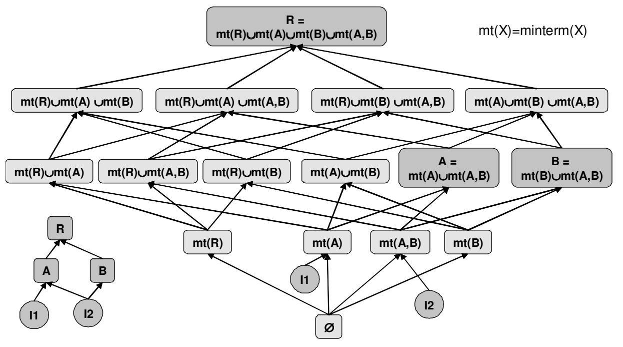

Figure 8. The extension of a small hierarchy.

Fig. 8 shows how the generalization hierarchy from the bottom-left side is extended to a significantly larger hierarchy. This hierarchy contains all the expressions from *E(Concepts)*. Notice that it offers more expressiveness than the initial hierarchy, while still using the same initial concepts. The learning method presented in this paper is based on this extended hierarchy.

C. Generalization hierarchy of objects

A generalization hierarchy of objects contains, in addition to a generalization hierarchy of concepts, some of the concepts' instances and their hierarchical relations.

Definition: A generalization hierarchy of objects is a structure *GH(Concepts, root-concept, subconcept-of, Instances, instance-of)*, that satisfies the following properties:

- *GH(Concepts, Root-Concept, subconcept-of)* is a generalization hierarchy of concepts.
- *Instances* is the set of all known instances in the hierarchy. Examples of instances from Fig. 1 are *Mark-White* and *George-Mason-University*.
- *instance-of* is a relation over *Instances x Concepts*, that has the following properties:
  1. is transitive in conjunction with *subconcept-of*, i.e. $\forall$ *i*$\in$*Instances, c1, c2*$\in$*Concepts, i instance-of c1 subconcept-of c2* $\Rightarrow$ *i instance-of c2*;
  2. is complete over *Instances*, i.e. $\forall$ *i*$\in$*Instances*, $\exists$ *c*$\in$*Concepts, i instance-of c*.

Lemma: All instances are in the relation *instance-of* with the *root-concept*.

A very important aspect of the instances is that not all of them are defined in a given generalization hierarchy of objects. Therefore a concept will not be equivalent with the set of all its instances defined at a particular time in the generalization hierarchy.

Definition: The relation ***direct-instance-of*** is defined by: *i direct-instance-of c* if and only if *i instance-of c* and there is no concept *c'* such that *i instance-of c' subconcept-of c*. *di(C)* represents all known direct instances of *C*.

A minterm of a concept *C* represents all possible direct instances of C (i.e. not only the ones present in the hierarchy) that have no other direct superconcept. Also:

$$minterm(C) \subset C \setminus \bigcup_{S\ directSubConceptOf\ C} S$$

Notice that $mt(c_1, c_2,... c_n)$ has an intuitive interpretation as representing all the possible common direct instances of the concepts c_1, c_2... c_n.

Lemma: The minimal concept from *E(Concepts)* that covers an instance *i* is $mt(c_1, c_2... c_n)$ where c_1, c_2... c_n are all the direct superconcepts of the instance *i*. We will denote this by *minC(i)*.

In the following sections we will denote with *GHT* the extension with the target T of the generalization hierarchy of concepts. Similarly, we will denote with *E(Concepts)*$\cup$*{T}* the extension with T of the generalization hierarchy *E(Concepts)*.

IV. Target's best approximation

A. Partial order relation for target approximations

There are many possible approximations of a target concept in *E(Concepts)*. For example, some of the approximations for *T2* in Fig. 5 are: *C, D,* $C\cup D$, *mt(F),* $A\cup B$.

In order to be able to compare how good the approximations are we will define a partial order relation between different approximations of a target concept. Although, there might be several criteria for comparing the approximations all of them must somehow express how well the approximations overlap the target concept. Because any approximation T_A will only partially overlap the target *T*, the following definitions identify two important sets of instances.

Definition: The ***negative exceptions*** from the point of view of the approximation T_A of a target concept *T* are the elements in the approximation that are not in the target, i.e. $\textit{Negative-exceptions}_T(T_A)=T_A \setminus T$.

Definition: The ***positive exceptions*** from the point of view of the approximation T_A of a target concept *T* are the elements in the target that are not in the approximation, i.e. $\textit{Positive-exceptions}_T(T_A)=T \setminus T_A$.

Notice that we consider only the instances that are represented in the hierarchy. Therefore, in Fig. 5, for the approximation *B* of *T2*, *I3* is a negative exception and *I1* is a positive exception.

Notice that, together, the positive and the negative exceptions characterize the overlap between the target concept and an approximation. Therefore we will define a partial order relation between approximations based on these sets of exceptions.

Definition: An approximation T_{A1} of the target concept *T* is better than (or equally good as) an approximation T_{A2} if and only if $\textit{Negative-exceptions}_T(T_{A1})\subseteq \textit{Negative-exceptions}_T(T_{A2})$ and $\textit{Positive-exceptions}_T(T_{A1})\subseteq \textit{Positive-exceptions}_T(T_{A2})$

Two approximations are equally good if and only if they generate the same positive and negative exceptions.

For example, in Fig. 5 the following order relations hold: $C\cup D$ is a better approximation of *T2* than *A*, because $C\cup D$ will have the same positive exceptions as *A* (I4+), but has no negative exceptions. However *A* has *I3* as negative exception. There is no order relation between *B* and *F* as approximations of *T2*. Because $F\subset B$, $\textit{Positive-exceptions}(B)\subseteq \textit{Positive-exceptions}(F)$. Notice also that *I2* is a positive exception for *F*, but not for *B*. Therefore, we obtain $\textit{Positive-exceptions}(B)\subset \textit{Positive-exceptions}(F)$. Similarly, $\textit{Negative-exceptions}(F)\subseteq \textit{Negative-exceptions}(B)$. Because *I3* is a negative exception for *B*, but not for *F*, we obtain $\textit{Negative-exceptions}(F)\subset \textit{Negative-exceptions}(B)$.

The definition of a better approximation is mathematically justified by the fact that the errors made by a worse approximation always include the errors made by a better approximation. However, a disadvantage of this definition is that it is very computationally expensive, being based on an exhaustive enumeration of the instances of the target concept. There are many other partial order relations that are easier to use. For instance, one may consider the weighted sum of the number of the positive exceptions and the number of the negative exceptions known at some particular moment. However, our definition has a significant advantage, providing a necessary condition that any other partial order relation must satisfy, as stated in following lemma.

Lemma: An approximation that is better in our selected partial order relation must be better in any other partial order relation that is based on a measure of overlap between the target and the approximation. Therefore, any property which holds for our selected partial order relation will also hold for any other partial order relation.

B. The lower approximation of the target concept

In this section we will study approximations that are always included into the target concept.

Definition: An approximation T_A is a ***conservative approximation*** of the target concept *T* if T_A is included into *T*.

Lemma: An approximation T_A of the target concept *T* has the following properties:

- T_A is a conservative approximation if and only if it does not generate negative exceptions;
- a conservative approximation $T_A \neq T$ always generates positive exceptions.

In the following we will construct the best conservative approximation of the target, called target's lower approximation, and show some of its properties.

Definition: The ***target's lower approximation***, T_{LA}, is the union of all the subconcepts of the target from *E(Concepts)*:

$$T_{LA} = \bigcup_{S \textit{ subconceptOf } T \textit{ in } E(\textit{Concepts}) \bigcup T} S$$

Lemma: The following properties hold:

- T_{LA} is a conservative approximation,
- T_{LA} is the only direct subconcept of *T* in the $E(\textit{Concepts})\cup\{T\}$.
- Any other conservative approximation of *T* is a subconcept of (i.e. it is included into) T_{LA}.
- T_{LA} is the best conservative approximation of the target concept *T* in *E(Concepts)*.

For computational and learning reasons we will divide the lower approximation into two parts: a disjunction of concepts from the initial hierarchy of concepts ($T_{\text{LA-FC}}$) and a disjunction of the remaining minterms ($T_{\text{LA-MT}}$).

Lemma: T_{LA} may be decomposed in two disjoint sets, as shown in the following equations:

$$T_{LA} = T_{LA-FC} \bigcup T_{LA-MT}$$

$$T_{LA-FC} = \bigcup_{S \textit{ directSubconceptOf } T \textit{ in } GHT} S$$

$$T_{LA-MT} = \bigcup_{M \textit{ minterm of } T \textit{ in } GHT\ ,\ M \not\subset T_{LA-FC}\ ,\ M \subset T \textit{ in } GHT} M$$

For the two target concepts considered in Fig. 4 and Fig. 5 their respective lower approximations are shown in Fig. 9.

For the hierarchy from the left hand side of Fig. 5 *T1*$_{LA}$ is empty. This situation always happens when the target concept is added as a terminal concept in the existing generalization hierarchy. In such a case there is no other conservative approximation for the target concept. In the hierarchy from the right hand side of Fig. 5, *T2*$_{LA}$=*C*∪*D*. There are other possible conservative approximations: *C, D* and the empty set. All of them are included into the best conservative approximation, which is *T2*$_{LA}$.

C. The upper approximation of the target concept

In this section we will study the target's approximations that always include the target concept.

Definition: An approximation T_A is a ***complete-covering approximation*** of the target concept *T* if *T* is included into T_A.

Lemma: An approximation T_A of the target concept *T* has following properties:

- T_A is a complete-covering approximation if and only if it does not generate positive exceptions.
- A complete-covering approximation $T_A \neq T$ always generates negative exceptions.

In the following we will construct the best complete-covering approximation of the target, called target's upper approximation, and show some of its properties.

Definition: The ***target's upper approximation***, T_{UA}, is the intersection of all superconcepts of the target *T* from *E(Concepts)*:

$$T_{UA} = \bigcap_{S\ superconceptOf\ T\ in\ E(Concepts) \cup T} S$$

Lemma: The following properties hold:

- T_{UA} is a complete-covering approximation.
- T_{UA} is the only direct superconcept of *T* in the *E(Concepts)*∪*{T}*.
- Any other complete-covering approximation of *T* is a superconcept of T_{UA}.
- T_{UA} is the best complete-covering approximation of the target concept *T* from *E(Concepts).*

Lemma: T_{UA} may be decomposed in two disjoint sets, as shown in the following equations:

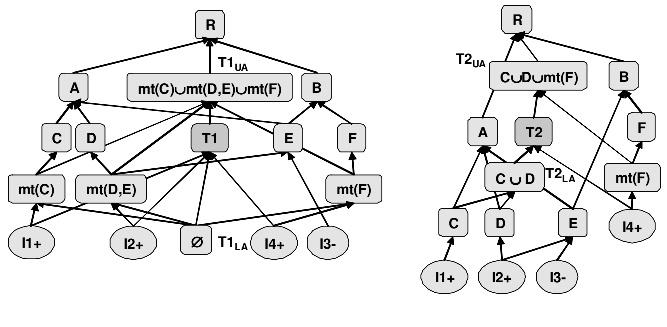

Figure 9. The lower and upper approximations for two target concepts.

$$T_{UA} = T_{LA} \cup T_{UA-MT}$$

$$T_{UA-MT} = \bigcup_{M\ minterm\ of\ T,\ M \not\subset T_{LA},\ M \cap T \neq \Phi\ in\ E(Concepts) \cup T} M$$

For the two target concepts considered in Fig. 4 and Fig. 5 the corresponding target's upper approximations are shown in Fig. 9. For the hierarchy from the left hand side of the Fig. 5 *T1*$_{UA}$ = *mt(C)*∪*mt(D,E)*∪*mt(F).* In the hierarchy from the right hand side of the Fig. 5 *T2*$_{UA}$=*C*∪*D*∪*mt(F).* There are other possible complete-covering approximations: *C*∪*D*∪*F, A*∪*B, R*. Each of them includes the best complete-covering approximation *T2*$_{UA}$.

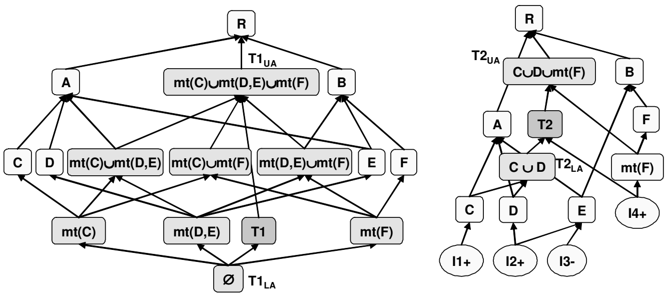

Figure 10. Target's best approximation version space.

D. The version space of target's best approximations

We will consider a special version space bounded by T_{LA} and T_{UA}. This space will contain all the concepts *C* from *E(Concepts)* for which $T_{LA} \subset C \subset T_{UA}$. Fig. 10 shows the version spaces of the best approximations for the previous targets: *T1* and *T2*. Not all the concepts from *E(Concepts)* are shown but only the ones from these version spaces and the original generalization hierarchy. The concepts from these version spaces have a darker background.

For a given target concept, this version space is called the version space of the target's best approximations because it contains all the best approximations of the target for any possible order relation considered, as will result from the following theorem.

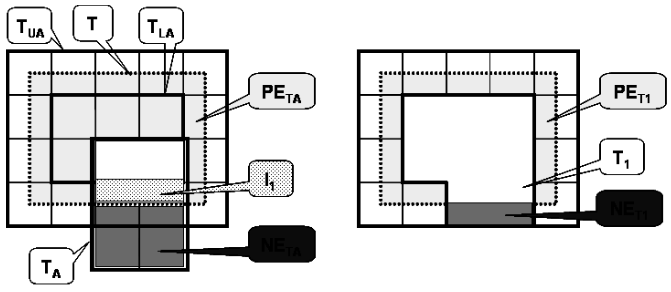

Figure 11. The target's best approximation theorem.

The target's best approximation theorem: For any approximation T_A of the target concept *T* there is an as good as or a better approximation than T_A in the version space bounded by T_{LA} and T_{UA} in *E(Concepts)*.

Proof: Let $di(T)=T\backslash T_{LA}$ representing the direct instances of T in *E(Concepts)*. Let $I_1= T_A \cap di(T)$, representing the direct instances of T covered by T_A. Let,

$$T_1 = T_{LA} \cup \left[\bigcup_{i \in I_1} minC(i) \right]$$

We will show that T_1 is an approximation as good as or better than T_A. It is clear that T_1 is included into the version space bounded by T_{LA} and T_{UA} (because $T_{LA} \subseteq T_1 \subseteq T_{UA}$).

Let us compute the positive exceptions and the negative exceptions of T_1: $PE_{T1}=di(T)\backslash I_1$

$$NE_{T1} = \left[\bigcup_{i \in I_1} minC(i) \right] \backslash T$$

Fig. 11 illustrates the concepts used in this theorem. The small squares represent the minterms. All the concepts from *E(Concepts)* must be formed by a union of minterms. The left hand side represents T_A and the right hand side T_1.

Let us compare the positive exceptions of T_A with the positive exception of T_1. By definition $(di(T)\backslash I_1)\cap T_A=\varnothing$, and $(di(T)\backslash I_1)\subset T$. Therefore $(di(T)\backslash I_1)$ are positive exceptions of T_A, i.e. $(di(T)\backslash I_1)\subseteq PE_{TA}$. Moreover $PE_{T1}=di(T)\backslash I_1$. Therefore $PE_{T1}\subseteq PE_{TA}$.

Regarding the negative exceptions we have that $I_1\subset T_A$, and $I_1\subset T$. However, the instances from I_1 must be introduced in any approximation by their corresponding minterms *minC(i)*, which will conclude that

$$\left[\bigcup_{i \in I_1} minC(i) \right] \subset T_A .$$

Therefore, the instances that do not belong to T from this union are negative exceptions for T_A:

$$\left[\bigcup_{i \in I_1} minC(i) \right] \backslash T \subseteq NE_{TA} .$$

Moreover

$$NE_{T1} = \left[\bigcup_{i \in I_1} minC(i) \right] \backslash T .$$

Therefore: $NE_{T1}\subseteq NE_{TA}$.

Because $PE_{T1}\subseteq PE_{TA}$ and $NE_{T1}\subseteq NE_{TA}$ the approximation T_1 will be better than or as good as T_A. *QED*.

This result is very important because focuses the goal of the learning algorithm when the target concept is not represented in the generalization hierarchy. If the target concept is represented in the hierarchy then the learning algorithm must find it. If the target concept is not in the hierarchy, then the learning algorithm must converge toward one of the target approximations belonging to the version space bounded by T_{LA} and T_{UA} in *E(Concepts)*.

However, a better approach is to have a learning algorithm that will try to determine the version space of the target's best approximations. In order to determine this version space it is enough to determine the version space bounds T_{LA} and T_{UA}. Any other element in the version space can be easily constructed based on them. In the next section we will present a method to learn the target's lower approximation and the target's upper approximation.

V. Learning the lower and the upper approximations of the target concept

A. Concepts' support from examples

As indicated in Table 1, the target concept T, or an approximation of it, is incrementally learned from a sequence of instances i^1, i^2... i^n classified either as positive examples (i.e., covered by the target concept) or as negative examples (i.e., not covered by the target concept).

Definition: We denote with PE^n the set of positive examples, $PE^n=\{i^k, k\leq n \mid type\text{-}of(i^k)=positive\}$.

Definition: We denote with NE^n the set of negative examples, $NE^n=\{i^k, k\leq n \mid type\text{-}of(i^k)=negative\}$.

A positive example which is an instance of a concept C_1 from the generalization hierarchy will support the assumption that C_1 is a part of the target. By contrast, a negative example which is an instance of a concept C_2 will support the assumption that C_2 is not a part of the target. Using the instance-of relations corresponding to the known examples of the target concept, one can classify the concepts from the generalization hierarchy, as indicated by the following definitions and illustrated in Fig. 12.

Definition: A ***direct positive concept*** is a concept that covers at least one positive example and does not cover any negative example, such as C01 and C12 in Fig. 12.

Definition: A ***direct negative concept*** is a concept that covers at least one negative example and does not cover any positive example, such as C03 and C14 in Fig. 12.

Definition: A ***direct irrelevant concept*** is a concept that covers both a positive example and a negative example, such as R and C02 in Fig. 12.

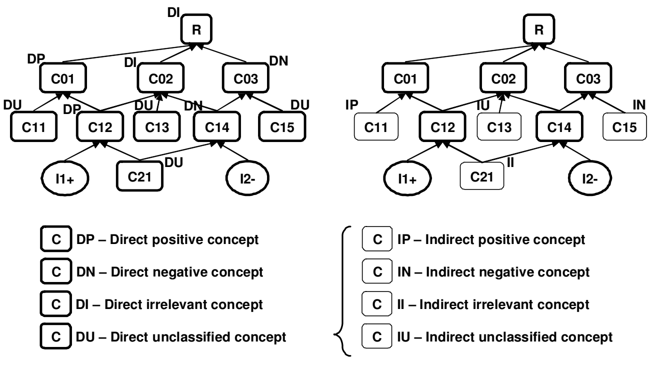
Direct support
Indirect support
DI
R
DP
DI
DN
C01
C02
C03
DU
DP
DU
DN
DU
C11
C12
C13
C14
C15
I1+
C21
DU
I2-
IP
IU
IN
II
C
DP – Direct positive concept
DN – Direct negative concept
DI – Direct irrelevant concept
DU – Direct unclassified concept
IP – Indirect positive concept
IN – Indirect negative concept
II – Indirect irrelevant concept
IU – Indirect unclassified concept

Figure 12. Concept classification based on instances support.

Definition: A ***direct unclassified concept*** is a concept that does not cover any positive example and any negative example, such as C11, C13, C15 and C21 in Fig. 12.

The direct unclassified concepts may be further classified based on their indirect support received from the direct positive and negative concepts.

Definition: An ***indirect positive concept*** is a direct unclassified concept that has a direct positive superconcept and no direct negative superconcept, such as C11 in Fig. 12.

Definition: An ***indirect negative concept*** is a direct unclassified concept that has a direct negative superconcept and no direct positive superconcept, such as C15 in Fig. 12.

Definition: An ***indirect irrelevant concept*** is a direct unclassified concept that has both direct positive superconcepts and direct negative superconcepts, such as C21.

Definition: An ***indirect unclassified concept*** is a direct unclassified concept that has neither a direct positive superconcept nor a direct negative superconcept, such as C13.

As shown in the previous section, the lower approximation of the target concept T is $T_{LA}=T_{LA\text{-}FC}\cup T_{LA\text{-}MT}$. We will divide the learning of T_{LA} into learning its two disjoint parts $T_{LA\text{-}FC}$ and $T_{LA\text{-}MT}$. We will first present methods to learn $T_{LA\text{-}FC}$. By definition, $T_{LA\text{-}FC}$ is the union of direct subconcepts of T in GHT. Because we do not know the direct subconcepts of T we will try to determine all the plausible candidates.

Because T_{LA} must not have negative exceptions it must not cover direct negative concepts or direct irrelevant concepts. Therefore T_{LA} will always be expressed with direct positive concepts and direct unclassified concepts. Moreover, any union of direct positive and direct unclassified concepts from GH (which are not hierarchically connected) will be a plausible candidate for $T_{LA\text{-}FC}$. We will construct a version space with all these plausible candidates, as described in the next section.

B. *The version space for $T_{LA\text{-}FC}$ candidates*

Let us assume that we have *k* examples i^1, i^2... i^k of the target concept. We may accordingly classify the concepts from *GH*. This classification will evolve as new examples will be received.

Let us first look at the direct unclassified concepts. All the subconcepts of a direct unclassified concept will also be direct unclassified concepts. Therefore the most general direct unclassified concepts will determine an upper bound for the set of all direct unclassified concepts. Let UC^k_{UB} denote the upper bound of all the direct unclassified concepts.

A direct positive concept will have all its subconcepts either direct positive concepts or direct unclassified concepts. The set of the most general direct positive concepts will be an upper bound for all the direct positive concepts, but will also cover some direct unclassified concepts. Let PC^k_{UB} denote the upper bound of all the direct positive concepts.

Because our goal is to approximate both the unclassified and the positive concepts the fact that the concepts from PC^k_{UB} cover also unclassified concepts does not create any problem. These two bounds contain distinct concepts but they generally cover common instances. Their union $UC^k_{UB}\cup PC^k_{UB}$ represents the upper bound of the concepts that are either direct positive or direct unclassified.

We use of the following notation where S is a set of sets:

$$\bigcup S = \bigcup_{A\in S} A$$

Lemma: $T^k_{LA\text{-}FC\text{-}UB}=\cup(UC^k_{UB}\cup PC^k_{UB})$ is an upper bound of the candidates for $T_{LA\text{-}FC}$ in *E(Concepts)*.

Proof: Any union of concepts from *GH* covered by $\cup(UC^k_{UB}\cup PC^k_{UB})$ is a candidate for $T_{LA\text{-}FC}$ because it contains concepts from the original generalization hierarchy, and does not cover any known negative example. Moreover, the lower approximation $T_{LA\text{-}FC}$ must be included in this set, because all other concepts from *GH* already cover a negative example and thus cannot be in the union of $T_{LA\text{-}FC}$.

Lemma: $T^k_{LA\text{-}FC\text{-}LB}=\varnothing$ is a lower bound of the candidates for $T_{LA\text{-}FC}$.

Proof: For $T_{LA\text{-}FC}$ the empty set is a candidate until all possible examples are used, because there may always be a negative example under each concept from the union $T^k_{LA\text{-}FC\text{-}UB}$. Real world concepts have very many instances, possibly infinite. Therefore for such real situations we will not use all the examples and $T^k_{LA\text{-}FC\text{-}LB}$ is a lower bound of the candidates.

Lemma: The version space of the candidates for $T_{LA\text{-}FC}$, after *k* examples are given, denoted with $VS(T^k_{LA\text{-}FC})$, is bounded by $T^k_{LA\text{-}FC\text{-}LB}=\varnothing$ as the lower bound and $T^k_{LA\text{-}FC\text{-}UB}=\cup(UC^k_{UB}\cup PC^k_{UB})$ as the upper bound.

Considering the target concept *T2* from the previously given example (see Fig. 9) we obtain the following sequence of bounds, as new examples are classified: $T^1_{LA\text{-}FC\text{-}UB}=R$, $T^2_{LA\text{-}FC\text{-}UB}=R$, $T^3_{LA\text{-}FC\text{-}UB}=C\cup D\cup F$, $T^4_{LA\text{-}FC\text{-}UB}= C\cup D\cup F$, $T^5_{LA\text{-}FC\text{-}UB}= C\cup D$. The lower bound $T^k_{LA\text{-}FC\text{-}LB}$ is the empty set at each of the previous steps. Fig. 13 shows in a darker color the direct positive and the direct unclassified concepts as new instances are classified.

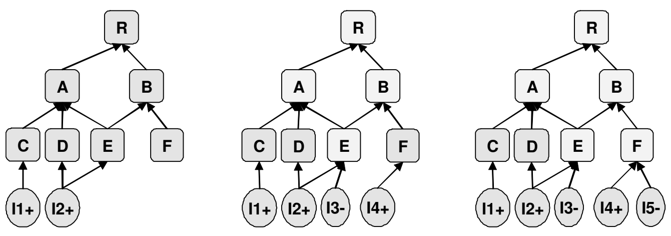

Figure 13. Learning the $T_{LA\text{-}FC}$.

Theorem of monotonic convergence to $T_{LA\text{-}FC}$: The more examples are used the smaller the version space of the $T_{LA\text{-}FC}$ candidates becomes, i.e. for any number of examples $k>0$, $T_{LA\text{-}FC} \subseteq T^{k+1}_{LA\text{-}FC\text{-}UB} \subseteq T^{k}_{LA\text{-}FC\text{-}UB}$.

Proof: Let us assume that the new example is positive. The superconcepts of this example will change their status as follow: the negative concepts will become irrelevant; the irrelevant concepts will remain irrelevant; the positive concepts will remain positive; the unclassified concepts will become positive. Therefore $T^{k+1}_{LA\text{-}FC\text{-}UB} = T^{k}_{LA\text{-}FC\text{-}UB}$. Now let us assume that the new example is negative. The previously negative and irrelevant concepts which are superconcepts of this example will remain the same. However the previously neutral superconcepts will become negative and the previously positive superconcepts will become irrelevant. Both of these types will be removed from $T^{k}_{LA\text{-}FC\text{-}UB}$. Therefore the upper bound will be specialized covering fewer concepts, i.e. $T^{k+1}_{LA\text{-}FC\text{-}UB} \subset T^{k}_{LA\text{-}FC\text{-}UB}$. *QED*

This theorem is very important because it shows that this version space monotonically converges toward $T_{LA\text{-}FC}$. Moreover, we have proved that only negative examples influence the size of this version space.

C. *The plausible version space for $T_{LA\text{-}MT}$ candidates*

The learning of $T_{LA\text{-}MT}$ and $T_{UA\text{-}MT}$ is strongly correlated with the learned expression of $T_{LA\text{-}FC}$, because both of them add some minterms to it. Given $T_{LA\text{-}FC}$, for each positive example not covered by $T_{LA\text{-}FC}$, the minterm that contains this positive example (i.e. *minC(i)*) is added to $T_{LA\text{-}MT}$, if it does not cover any negative example, and to $T_{UA\text{-}MT}$ otherwise.

However, there are many plausible candidates for both $T_{LA\text{-}MT}$ and $T_{UA\text{-}MT}$. For $T_{LA\text{-}MT}$ any minterm not included into $T_{LA\text{-}FC}$ that does not cover any negative example may be part of a plausible candidate. For $T_{UA\text{-}MT}$ any minterm not included into $T_{LA\text{-}FC}$ may be part of a plausible candidate.

Let us first analyze the possible candidates for $T_{LA\text{-}MT}$ for a fixed value of $T_{LA\text{-}FC}$, denoted with $T^{k}_{LA\text{-}FC}$.

Lemma: The version space of the candidates for $T_{LA\text{-}MT}$ is the set $VS(T^{k}_{LA\text{-}MT})$:

$$\left\{ C \in E(Concepts) \middle| \begin{array}{l} C \cap T^{k}_{LA-FC} = \varnothing \wedge C \cap NE^{k} = \varnothing \\ \wedge \forall C_1 \in Concepts, C_1 \not\subset C \end{array} \right\}$$

The lower bound of this version space is the empty set, and the upper bound contains the most general of them. This version space will decrease as $T^{k}_{LA\text{-}FC}$ increases. However this version space contains a huge number of possible candidates and it will decrease very slowly during learning, requiring a very large number of negative examples. Therefore the use of this version space for learning will be non operational.

One way to workaround this problem is to adopt a more conservative condition for a plausible candidate, one that will offer a smaller but more meaningful version space. By definition $T_{LA\text{-}MT}$ is the union of minterms included into the target concept. We may consider that a minterm must cover at least a known positive example and must not cover any known negative example in order to be considered as part of this union, as indicated in the following definition.

Definition: We denote with $LAPM^{k}$ the plausible minterms for the lower approximation:

$$LAPM^{k} = \left\{ \begin{array}{l} M \text{ minterm in } E(Concepts) \mid \\ M \cap PE^{k} \neq \varnothing \wedge M \cap NE^{k} = \varnothing \end{array} \right\}$$

In this case we will obtain the following plausible version space (named plausible because it no longer contains all possible candidates but only the most plausible ones):

Definition: We denote with $PVS(T^{k}_{LA\text{-}MT})$ the plausible version space for $T^{k}_{LA\text{-}MT}$ with candidates containing only minterms from $LAPM^{k}$:

$$PVS(T^{k}_{LA-MT}) = \left\{ \bigcup_{M \in P} M \mid P \subset \left\{ M \in LAPM^{k} \mid M \not\subset T^{k}_{LA-FC} \right\} \right\}$$

Lemma: $PVS(T^{k}_{LA\text{-}MT})$ is bounded by the following plausible lower bound and plausible upper bound:

$T^{k}_{LA\text{-}MT\text{-}PLB} = \varnothing$

$$T^{k}_{LA-MT-PUB} = \bigcup_{\exists i \in PE^{k} \setminus T^{k}_{LA-FC},\ M = \min C(i),\ M \cap NE^{k} = \varnothing} M$$

One may adopt another condition in order to construct the plausible version space. However, the one considered above is based on the intuition that a very specific concept (in our case a minterm) that is supported by some positive examples and is not contradicted by any known negative example is very likely to be a part of the target concept.

Lemma: Let us denote with $T^{k}_{LA\text{-}MT\text{-}CUB}$ the $T^{k}_{LA\text{-}MT\text{-}PUB}$ that correspond to $T^{k}_{LA\text{-}FC\text{-}UB}$. For any selection of $T^{k}_{LA\text{-}FC}$, $T^{k}_{LA\text{-}MT\text{-}CUB}$ is always included into the $T^{k}_{LA\text{-}MT\text{-}PUB}$. It represents a constant part that must appear in all plausible upper bounds.

Lemma: We have the following properties

$T^{k}_{LA\text{-}MT\text{-}CUB}$ is always between $\varnothing$ and $T_{LA\text{-}MT} \cup T_{UA\text{-}MT}$

$T^{k}_{LA\text{-}MT\text{-}CUB} \cap T_{LA\text{-}MT} \subseteq T^{k+1}_{LA\text{-}MT\text{-}CUB} \cap T_{LA\text{-}MT}$

Theorem of oscillatory convergence to $\mathbf{T_{LA\text{-}MT}}$: $T^{k}_{LA\text{-}MT\text{-}CUB}$ has an oscillatory convergence toward $T_{LA\text{-}MT}$ being bounded by $T^{k-1}_{LA\text{-}MT\text{-}CUB} \cap T_{LA\text{-}MT}$ and by $T_{LA\text{-}MT} \cup T_{UA\text{-}MT}$

These results are important because they show that $T^{k}_{LA\text{-}MT\text{-}CUB}$ is a good plausible candidate for $T_{LA\text{-}MT}$. The learning method proposed will be based on it.

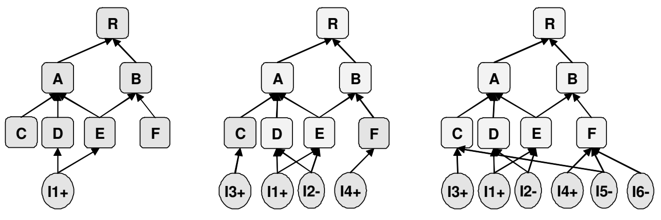

Figure 14. A series of examples for a target concept T3.

Fig. 14 shows a series of examples for a target *T3* that is placed in *E(Concepts)* between *mt(C)* and $mt(C) \cup mt(D,E) \cup mt(F)$. Therefore $T3_{LA\text{-}MT}=mt(C)$.

Table 2 shows how $T_{LA\text{-}MT}$ is learned for the target *T3*. Notice the oscillatory convergence of $T^k_{LA\text{-}MT\text{-}CUB}$.

Table 2. Learning of $T_{LA\text{-}MT.}$

	$T_{LA\text{-}FC}$		$T_{LA\text{-}MT}$		
	LB	UB	PLB	PUB	CUB
I1+	∅		∅	mt(D,E)	∅
		R	∅	∅	
I2-	∅		∅	∅	∅
		C∪F	∅	∅	
I3+	∅		∅	mt(C)	∅
		C∪F	∅	∅	
I4+	∅		∅	mt(C)∪mt(F)	∅
		C∪F	∅	∅	
I5-	∅		∅	mt(C)∪mt(F)	mt(C)∪mt(F)
		∅	∅	mt(C)∪mt(F)	
I6-	∅		∅	mt(C)	mt(C)
		∅	∅	mt(C)	

D. The version space of $T_{UA\text{-}MT}$ candidates

Let us now analyze the possible candidates for $T^k_{UA\text{-}MT}$. Notice that, in order to compute $T^k_{UA\text{-}MT}$, we need to know the values for both $T^k_{LA\text{-}FC}$ and $T^k_{LA\text{-}MT}$. We consider fixed values of $T^k_{LA\text{-}FC}$ and $T^k_{LA\text{-}MT}$, which implies a fixed value for T^k_{LA}. $T_{UA\text{-}MT}$ is the union of minterms that are only partially covered by the target concept. Therefore the minterms that cover both positive and negative examples must be included into $T^k_{UA\text{-}MT}$.

Definition: The set of minterms which are partially covered by the target concept, determined after first *k* examples, is denoted by:

$$PCM^k = \left\{ \begin{array}{l} M \text{ minterm in } E(Concepts) \mid \\ M \cap PE^k \neq \varnothing \wedge M \cap NE^k \neq \varnothing \end{array} \right\}$$

Lemma: The version space for $T^k_{UA\text{-}MT}$ is:

$$VS(T^k_{UA-MT}) = \left\{ C \in E(Concepts) \mid C \cap T^k_{LA} = \varnothing \wedge \bigcup_{M \in PCM^k} M \subset C \right\}$$

Lemma: We consider the target lower approximation as being T^k_{LA}. The version space $VS(T^k_{UA\text{-}MT})$ is bounded by:

$$T^k_{UA-MT-LB} = \bigcup_{M \in PCM^k} M \cup \bigcup_{M \in LAPM^k \wedge M \not\subset T^k_{LA}} M$$

$$T^k_{UA-MT-UB} = R \setminus T^k_{LA}$$

As in the case of $VS(T^k_{LA\text{-}MT})$, this version space will decrease very slowly. Therefore the upper bound is not relevant for learning. However the lower bound will accumulate the minterms that must be in $T^k_{UA\text{-}MT}$ and is relevant for learning.

Lemma: Let

$$T^k_{UA-MT-CLB} = \bigcup_{M \in PCM^k} M \quad .$$

For any selection of $T^k_{LA\text{-}FC}$ and $T^k_{LA\text{-}MT}$, $T^k_{UA\text{-}MT\text{-}CLB}$ is always included into the $T^k_{UA\text{-}MT\text{-}LB}$. It represents a constant part that must appear in all plausible lower bounds.

Lemma: We have the following properties

$T^k_{UA\text{-}MT\text{-}CLB}$ is always between $\varnothing$ and $T_{UA\text{-}MT}$

$T^k_{\text{UA-MT-CLB}} \subseteq T^{k+1}_{\text{UA-MT-CLB}}$

Theorem: $T^k_{UA\text{-}MT\text{-}CLB}$ has a monotonic convergence toward $T_{UA\text{-}MT}$

These results are important because they show that $T^k_{UA\text{-}MT\text{-}CLB}$ is a good plausible candidate for $T_{UA\text{-}MT}$. The learning method proposed will be based on it.

Let us consider again the examples from Fig. 13. Table 3 shows how the bounds of the version spaces evolve as new examples are used. Each row shows the bounds of the version spaces for the indicated value of $T_{LA\text{-}FC}$ after the example from the header was used. When $T_{LA\text{-}FC}$ has a value between its lower and upper bound the lower bounds and the upper bounds of the two version spaces have values between the indicated values.

Table 3. Learning of $T_{\text{UA-MT}}$

	$T_{LA}=T_{LA\text{-}FC} \cup T_{LA\text{-}MT}$				$T_{UA}=T_{LA} \cup T_{UA\text{-}MT}$		
	$T_{LA\text{-}FC}$	$T_{LA\text{-}MT}$			$T_{UA\text{-}MT}$		
		PLB	PUB	CUB	LB	UB	CLB
I1+	LB=∅	∅		∅	mt(C)	R	∅
			mt(C)		∅	R\mt(C)	
	UB=R	∅			∅	∅	
			∅		∅	∅	
I2+	LB=∅	∅		∅	mt(C)∪mt(D,E)	R	∅
			mt(C)∪mt(D,E)		∅	R\(mt(C)∪mt(D,E))	
	UB=R	∅			∅	∅	
			∅		∅	∅	
I3-	LB=∅	∅		∅	mt(C)∪mt(D,E)	R	∅
			mt(C)∪mt(D,E)		∅	R\(mt(C)∪mt(D,E))	
	UB=C∪D ∪F	∅			∅	R\(C∪D ∪F)	
			∅		∅	R\(C∪D ∪F)	
I4+	LB=∅	∅		∅	mt(C)∪mt(D,E)∪mt(F)	R	∅
			mt(C)∪mt(D,E)∪mt(F)		∅	R\(mt(C)∪mt(D,E)∪mt(F))	
	UB=C∪D ∪F	∅			∅	R\(C∪D ∪F)	
			∅		∅	R\(C∪D ∪F)	
I5-	LB=∅	∅		∅	mt(C)∪mt(D,E)∪mt(F)	R	mt(F)
			mt(C)∪mt(D,E)		mt(F)	R\(mt(C)∪mt(D,E))	
	UB=C∪D	∅			mt(F)	R\(C∪D)	
			∅		mt(F)	R\(C∪D)	

E. The learning algorithm

Based on the above analysis of the plausible candidates of the target's lower and upper approximations we have developed the algorithm from Table 4.

After k examples i^1, i^2... i^k the learned bounds are:

- $T^k_{\text{LA-FC-UB}}$=TLA-FC
- $T^k_{\text{LA-MT-CUB}}$=TLA-MT
- $T^k_{\text{UA-MT-CLB}}$=TUA-MT

To avoid storing all the examples, one keeps the minterms that cover them together with their respective classification:

- Minterms covering only positive examples: *OPMT∪TLA-MT*
- Minterms covering both positive and negative examples: *TUA-MT*

- Minterms covering only negative examples: *ONMT*

The bounds of a partially learned concept may be used in problem solving to classify a new instance *i*. If the instance is covered by any of the three disjoint sets (*TLA-FC*, *TLA-MT* or *TUA-MT*) it will be considered as a plausible positive example, otherwise it will be a plausible negative example. However, the level of confidence in the classification is different in each case, as follows:

- If $i \in TLA\text{-}FC$ the instance is only covered by direct positive or direct unclassified concepts. This case gives the highest level of confidence that *i* is a positive example.
- If $i \in TLA\text{-}MT$ the instance is covered by irrelevant or negative concepts, but its minterm does not contain any negative example. In this case there is a lower level of confidence that *i* is a positive example.
- If $i \in TUA\text{-}MT$ the instance is covered by a minterm that also covers a negative example. Therefore *i* might be either a positive or negative example. This case gives the lowest level of confidence that *i* is a positive example.

Table 4. The Learning algorithm

```
Global Structures:
    GH(Concepts, Instances, Root-Concept, subconcept-of, instance-of)
    PVS(TLA-FC, TLA-MT, TUA-MT, OPMT, ONMT)
Initialization:
    TLA-FC $\leftarrow$Root-concept; TLA-MT$\leftarrow\emptyset$; TUA-MT$\leftarrow\emptyset$
    OPMT$\leftarrow\emptyset$;the minterms that cover a positive example covered by TLA-FC
    ONMT$\leftarrow\emptyset$;the minterms that cover a negative example and are not in TUA-MT
Classify-Positive-Example(pi$\in$Instances)
    mt-pi$\leftarrow$minC(pi) (the minterm that covers the positive example)
    if pi covered by a concept from TLA-FC then
        if mt-pi$\notin$OPMT then add minterm mt-pi to OPMT
    else if mt-pi$\in$ONMT then
        add minterm mt-pi to TUA-MT
        delete minterm mt-pi from ONMT
    else if mt-pi$\notin$TLA-MT and mt-pi$\notin$TUA-MT then
        add minterm mt-pi to TLA-MT
Classify-Negative-Example(ni$\in$Instances)
    mt-ni$\leftarrow$minC(ni) (the minterm that covers the negative example)
    if ni covered by a concept from TLA-FC then
        minimally specialize TLA-FC to no longer cover ni
        if mt-ni$\in$OPMT then
            delete mt-ni from OPMT
            add mt-ni to TUA-MT
        else
            add mt-ni to ONMT
        for each mt$\in$OPMT do
            if mt not covered by TLA-FC then
                delete mt from OPMT
                add mt to TLA-MT
    else if mt-ni$\in$TLA-MT then
        delete mt-ni from TLA-MT
        add mt-ni to TUA-MT
    else if mt-ni$\notin$TUA-MT and mt-ni$\notin$ONMT then
        add mt-ni to ONMT
```

This classification of confidence levels justifies why we are considering the plausibility condition for $T_{LA\text{-}MT}$ and the constant part of the lower bound of $T_{UA\text{-}MT}$. Because, during problem solving the plausible candidates for $T_{LA\text{-}MT}$ and $T_{UA\text{-}MT}$ are generating instances that have lower levels of confidence of being positive examples. To minimize the chances for error we consider only the most plausible candidates for these parts.

The learning algorithm has the following strengths (as demonstrated in the previous sections):

- $T^{k}_{LA\text{-}FC\text{-}UB}(=TLA\text{-}FC)$ converges monotonically (through successive specializations) toward the exact value of $T_{LA\text{-}FC}$
- $T^{k}_{LA\text{-}MT\text{-}CUB}(=TLA\text{-}MT)$ converges oscillatorily toward the exact value of $T_{LA\text{-}MT}$ but it is always bounded by $T^{k\text{-}1}_{LA\text{-}MT\text{-}CUB} \cap T_{LA\text{-}MT}$ (as a generalization of it) and by $T_{LA\text{-}MT} \cup T_{UA\text{-}MT} \setminus T^{k}_{UA\text{-}MT\text{-}CLB}$ (as a specialization of it).
- $T^{k}_{UA\text{-}MT\text{-}CLB}(=TUA\text{-}MT)$ converges monotonically (through successive generalizations) toward the exact value of $T_{UA\text{-}MT}$

Therefore the algorithm converges almost monotonically toward the lower and the upper approximations of the target.

The algorithm has also the following weaknesses:

$T^{k}_{LA\text{-}FC\text{-}UB}$ generally covers also many negative examples (because it is specialized toward the exact value). Therefore its use in problem solving to classify new instances or to generate positive examples leads to low confidence solutions.

$T^{k}_{LA\text{-}FC\text{-}UB}$ generally converges slowly toward $T_{LA\text{-}FC}$ (because it also includes the unclassified concepts) and requires a significant number of examples to reach the exact value.

When specializing *TLA-FC* to no longer cover a negative example, if some minterms from $T_{LA\text{-}MT}$ are no longer covered by *TLA-FC* and these minterms do not yet cover a positive example, they will no longer be covered by any of the maintained sets. In this case there is no trivial way in which the problem solving method may generate instances under them in order to reconsider them. As a consequence, the method may "loose" some minterms.

Another potential problem is the execution time that depends on the representation of the generalization hierarchy and of the concept bounds. However one may use an optimized representation of the hierarchy to allow fast computations, as described in [3].

VI. Related Research and Conclusions

The methods that are most related to the method presented in this paper are those based on the version space representation ([16], [14], [17], [1], and [2]), which are discussed in the following.

A. *The version space candidate elimination method*

The roots of our method are in Mitchell's candidate elimination method [16] which had significantly advanced the field of machine learning with a solid theoretical treatment of the concept learning problem. However, the applicability of the candidate elimination method to complex real-world problems is very limited due to several factors. The method assumes that the representation space is correct and complete, that the concept to be learned is in this space, and that there are no errors in the input examples. If the concept to be learned is not in this space, or if there are errors in the examples, the method will fail. Moreover the method suffers from the combinatorial explosion of the size of the bounds of the version space.

B. Extensions of the version space method

In its original form, Mitchell's candidate elimination method learns conjunctive concepts only. Several researchers have proposed extensions of the method so that it can also learn disjunctive concepts. An important example is the disjunctive version spaces method with delayed choice of bias, proposed by Sebag [17]. Her main idea was to search both for the target concept and for its negation (the concept that best characterizes the negative examples) and to consider, for each training example, the hypothesis space that covers this example and does not cover its counter examples. In order to decide on the classification of an instance, the method searches for a neighbor hypothesis space (positive or negative). The method allows noisy data ignoring the first found counter-examples that contradict the instance to be classified. It also avoids overgeneralization by requiring at least M attributes to be satisfied by the instance to be classified.

Both Sebag's method and our method allow noisy data by using a similar approach. For instance, in our method, we can require that a minterm be considered irrelevant only if both the number of covered positive examples, and the number of covered negative examples are above a certain threshold. However, in our method, the error may appear not only in the example but also in the generalization hierarchy. Sebag's method makes the assumption that a positive example and a negative example can always be discriminated in the hypothesis space. Our method allows for exceptions, that is, for negative and positive examples that cannot be distinguished in the current representation space of the concepts. As in the case of Mitchell's method, Sebag's method requires the target concept to be included in the representation language, as opposed to our method which can learn approximations of this concept.

Sebag's method is also computationally expensive, being proportional to the square of the number of examples. In our method the comparison is done between all positive examples and all negative examples at once, making the method more efficient. More significantly, however, is the fact that our method is based on a more complex representation language with an incomplete generalization hierarchy, while Sebag's method uses a much simple feature-vector representation. This makes our method applicable to significantly more complex application domains.

C. The plausible version space method

The plausible version space method developed by Tecuci [1] was a significant development of the version space approach, making it applicable to complex real world problems [2]. The method assumes an incomplete representation space which can be extended with new concepts during learning. This will increase the hypothesis space and will require concept revision. The method may learn an approximation of the target concept, when this is not representable, and may learn concepts in the presence of exceptions (e.g. covered negative examples). However, the method does not guarantee the learning of the best approximations of the target concept, the approximation learned depends on the order of the examples, and the convergence toward this approximation is oscillatory, not monotonic.

The method presented in this paper removes the above limitations. It learns the best approximations of the target concept, minimizes the number of positive and negative exceptions stored, and does not depend on the order of examples. Moreover, it extends the representation of the learned concept which can include conjunctions, disjunctions and negations.

D. Conclusions and future research

The most important aspect of the presented method is that it allows the learning of a version space containing the best approximations of a target concept when the target is not representable in the search space. The method has an almost monotonic convergence toward the best approximations of the bounds of the version space. Also, the learned concept does not depend on the order of examples. These add to the advantages already offered by the plausible version space learning method, such as learning in an incomplete and evolving representation space, and learning in the presence of exceptions.

There are several natural extensions of the presented method. One is to develop an optimized representation for learning in order to allow efficient learning algorithms. Another is to analyze the behavior of the method in the context of an evolving and partially incorrect representation.

In this paper we have addressed the basic problem of learning an approximation of a concept, from its positive and negative examples, in the context of a generalization hierarchy of concepts. However, the goal of a Disciple agent, for which this method was developed, is to learn complex concept expressions, such as the condition of the task reduction rule from Fig. 3 (see [2] for a formal description of such a concept). This condition is only partially learned, and is defined by a plausible upper bound (PUB) concept and a plausible lower bound (PLB) concept. The PLB concept is the set of tuples (?O1,?O2, ?O3, ?O4, ?O5, ?O6) that satisfy the expression of the PUB concept. That is, ?O1 should be a PhD advisor who has as employer ?O4 (which should be a university), and has as position ?O5 (which should be a tenured position). Moreover, ?O2 should be a PhD student, and ?O3 should be Artificial Intelligence. However, ?O1 should not be a PhD advisor who is likely to move to ?O6 (which is a university). A future research direction is to integrate the presented method into Disciple's learning method. This will allow the type of a variable (e.g. PhD advisor for ?O1) to be not just a concept from the generalization hierarchy, but also a union of minterms. This will very significantly increase the ability of Disciple to learn complex problem solving rules from subject matter experts, and therefore its practical applicability.

References

[1] G. Tecuci, "DISCIPLE: A Theory, Methodology and System for Learning Expert Knowledge," Ph.D. dissertation, Univ. of Paris-South, Paris, France, 1988.

[2] G. Tecuci, *BUILDING INTELLIGENT AGENTS: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*. San Diego, CA: Academic Press, 1998.

[3] M. Boicu, "Modeling and Learning with Incomplete Knowledge," Ph.D. dissertation, School of Inf. Tech. and Eng., George Mason Univ., Fairfax, VA, 2002.

[4] G. Tecuci, M. Boicu, M. Bowman, and D. Marcu, "An Innovative Application from the DARPA Knowledge Bases Programs: Rapid Development of a Course of Action Critiquer," *AI Magazine*, vol. 22, pp. 43-61, Feb., 2001.

[5] G. Tecuci, M. Boicu, D. Marcu, B. Stanescu, C. Boicu and J. Comello, "Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain," *AI Magazine*, vol. 24, pp.51 – 68, April, 2002.

[6] G. Tecuci, M. Boicu, C. Ayers, D. Cammons, "Personal Cognitive Assistants for Military Intelligence Analysis: Mixed-Initiative Learning, Tutoring, and Problem Solving," in *Proc. 1st Intern. Conf. Intelligence Analysis*, McLean, VA, 2005. Available: https://analysis.mitre.org/proceedings/index.html

[7] B. G. Buchanan and D. C. Wilkins, Ed. *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems,* San Francisco, CA: Morgan Kaufmann, 1993.

[8] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence*, New York: McGraw-Hill, 1971.

[9] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management*, Berlin: Springer-Verlag, 2000.

[10] B. Stanescu, C. Boicu, G. Balan, M. Barbulescu, M. Boicu, G. Tecuci, "Ontologies for Learning Agents: Problems, Solutions and Directions," in *Proc. IJCAI-03 Workshop on Ontologies and Distributed Systems*, Acapulco, Mexico: AAAI Press, 2003, pp. 75-82.

[11] M. Boicu, G. Tecuci, D. Marcu, M. Bowman, P. Shyr, F. Ciucu, and C. Levcovici, "Disciple-COA: From Agent Programming to Agent Teaching," in *Proc. of the 17th Int. Conf. on Machine Learning*, Stanford, California: Morgan Kaufman, 2000.

[12] G. Tecuci, M. Boicu, C. Boicu, D. Marcu, B. Stanescu, M. Barbulescu, "The Disciple-RKF Learning and Reasoning Agent", *Computational Intelligence*, vol 21, No. 4, 2005, pp 462-479.

[13] C. Boicu, G. Tecuci, M. Boicu, "Rule Refinement by Domain Experts in Complex Knowledge Bases," in *Proc. 20th Nat. Conf. on Artificial Intelligence*, Pittsburgh, PA, 2005.

[14] T. M. Mitchell, *Machine Learning*, New York: McGraw-Hill Companies, 1997.

[15] B. A. Davey and H A. Priestley, *Introduction to lattices and order*, Cambridge, NY: Cambridge University Press, 1990.

[16] T. M. Mitchell. "Version spaces: An approach to concept learning," Ph.D. Dissertation, Elec. Eng. Dept., Stanford Univ., Stanford, CA, 1979.

[17] M. Sebag. "Delaying the Choice of Bias: A Disjunctive Version Space Approach," in Saitta, L. (Ed.) *Machine Learning – Proceedings of the 13th Int. Conf.*, San Francisco, California: Morgan Kaufmann Publishers, Inc, 1996, pp. 444-452.

Authors Biographies

Mihai Boicu is Research Assistant Professor and Associate Director of the Learning Agents Center in the School of Information Technology and Engineering of George Mason University. He received a License in Informatics from the Bucharest University in 1995, and a Ph.D. in Information Technology from George Mason University in 2003. His domains of interest are knowledge representation, knowledge acquisition, multistrategy learning and mixed-initiative reasoning with applications to instructable agents. Dr. Boicu has published around 50 papers in these areas and has received several awards and recognitions for his professional activity, including the Outstanding Graduate Student Award from George Mason University, the Deployed Application Award from the American Association of Artificial Intelligence, two certificates of appreciation and the centennial coin from the US Army War College.

Gheorghe Tecuci is Professor of Computer Science in the School of Information Technology and Engineering and Director of the Learning Agents Center at George Mason University. Professor Tecuci is also a member of the Romanian Academy and former Chair of Artificial Intelligence at the US Army War College. He received an M.S. degree in Computer Science from the Polytechnic University of Bucharest in 1979, and two Ph.D. degrees in Computer Science, from the University of Paris-South and from the Polytechnic University of Bucharest, both in 1988. He joined George Mason University in 1990. Dr. Tecuci has published over 150 papers and 6 books, including "Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool, and Case Studies," "Machine Learning: A Multistrategy Approach," and "Machine Learning and Knowledge Acquisition: Integrated Approaches." His research focuses on creating and applying a general theory of how subject matter experts can directly teach automated agents how to solve problems.