LEARNING-BASED KNOWLEDGE REPRESENTATION

Gheorghe Tecuci and Mihai Boicu

This paper presents a learning-based representation of knowledge which is at the basis of the family of Disciple learning agents. It introduces a representation for concepts, generalization and specialization rules, different types of generalizations and specializations, and the representation of the main elements of a knowledge base, including partially learned concepts, problems, and rules. Finally, it provides a formal definition of generalization based on substitutions.

GEORGE MASON UNIVERSITY

RESEARCH REPORT 4

VOLUME 2008

Learning Agents Center MASON

LEARNING-BASED KNOWLEDGE REPRESENTATION

Gheorghe Tecuci and Mihai Boicu

Learning Agents Center, George Mason University, Fairfax, VA, 22030, USA

http://lac.gmu.edu

Keywords: concepts, rules, generalization rules, plausible bound, plausible version space, formal definition of generalization.

***Disclaimer**: The views and opinions expressed in this article are those of the authors and do not necessarily reflect the official policy or position of the National Science Foundation, the Air Force Office of Scientific Research, or any other agency of the U.S. government.*

8 September 2008

Learning-based Knowledge Representation

The knowledge representation of the Disciple agents was designed to facilitate the basic operations involved in learning, including comparing the generality of concepts, generalizing concepts, and specializing concepts. This has led to the learning-based representation presented in the following.

1. Concept Representation

Using the (object and feature) concepts from the object ontology (Tecuci and Boicu, 2008), one can define more complex concepts as logical expressions involving these primary concepts. For example, the concept "PhD student interested in a PhD research area" may be expressed as shown in [1].

$?O_1$	instance of	PhD student	[1]
	is interested in	$?O_2$	
$?O_2$	instance of	PhD research area	

One may interpret this concept as representing the set of instances of the pair $\{?O_1, ?O_2\}$ which satisfy the expression [1]. In general, the basic representation unit (BRU) for a more complex concept has the form of a sequence $(?O_1, ?O_2, \ldots, ?O_n)$, where each $?O_i$ has the structure indicated by [2], called ***clause***.

$?O_i$	instance of	$concept_i$	[2]
	$feature_{i1}$	$?O_{i1}$	
	. . .		
	$feature_{in}$	$?O_{im}$	

$Concept_i$ is either an object concept from the object ontology, or a numeric interval, or a set of numbers, or a set of strings, or an ordered set of intervals. $?O_{i1} \ldots ?O_{im}$ are distinct variables from the sequence $(?O_1, ?O_2, \ldots, ?O_n)$.

A concept may be a conjunctive expression of form [3], meaning that any instance of the concept satisfies BRU and does not satisfy BRU_1 and … and does not satisfy BRU_p.

$$BRU \wedge \text{not } BRU_1 \wedge \ldots \wedge \text{not } BRU_p \qquad [3]$$

However, instead of "not" we write "Except When". For example, expression [4] represents the concept "PhD student interested in a PhD research area that does not require programming".

$?O_1$	instance of	PhD student	[4]
	is interested in	$?O_2$	
$?O_2$	instance of	PhD research area	
Except When			
$?O_2$	instance of	PhD research area	
	requires	"programming"	

Exercise

What does the following concept represents?

$?O_1$	instance of	course
	has as reading	$?O_2$
$?O_2$	instance of	publication
	has as author	$?O_3$
$?O_3$	instance of	professor

The above concept represents the set of triples ($?O_1$, $?O_2$,$?O_3$) which, informally, can be expressed as "courses that have as readings publications by professors".

In the next sections we will describe in detail the basic learning operations dealing with concepts: comparing the generality of concepts, generalizing concepts, and specializing concepts.

2. Generalization and Specialization Rules

*A **generalization rule** is a rule that transforms a concept into a more general concept.* The generalization rules are usually inductive transformations. The inductive transformations are not truth-preserving but falsity-preserving. That is, if 'P' is true and is inductively generalized to 'Q', then the truth of 'Q' is not guaranteed. However, if P is false then Q is also false.

There are two other types of transformation rules: specialization rules and reformulation rules.

*A **specialization rule** transforms a concept into a less general one.* The reverse of any generalization rule is a specialization rule. Specialization rules are deductive, truth-preserving transformations.

*A **reformulation rule** transforms a concept into another, logically equivalent concept.* Reformulation rules are also deductive, truth-preserving transformations.

In (Tecuci and Boicu, 2008) a concept was defined as representing a set of instances. In order to show that a concept P is more general than a concept Q, that definition would require the computation of the (possibly infinite) sets of the instances of P and Q.

The generalization rules allow one to prove that a concept P is *more general than* another concept Q by manipulating the descriptions of P and Q, without computing the sets of instances that they represent:

If one can transform concept P into concept Q by applying a sequence of generalization rules, then Q is more general than P.

Some of the most used generalization rules, which are described in the following, are:

Turning constants into variables;
Turning occurrences of a variable into different variables;
Climbing the generalization hierarchies;

Dropping conditions;
Extending intervals;
Extending ordered sets of intervals;
Extending discrete sets;
Using feature definitions;
Using inference rules.

Turning Constants into Variables

The turning constants into variables generalization rule consists in generalizing an expression by replacing a constant with a variable. For example, expression [5] represents the following concept: "the set of professors with 55 publications".

E_1 =	$?O_1$	instance of	professor	[5]
		number of publications	55	

By replacing 55 with a variable $?N_1$ that can take any value, we generalize this concept to the one shown in [6]: "the set of professors with any number of publications". In particular, $?N_1$ could be 55. Therefore the second concept includes the first one.

E_2 =	$?O_1$	instance of	professor	[6]
		number of publications	$?N_1$	

Conversely, by replacing $?N_1$ with 55, we specialize the concept [6] to the concept [5]. The important thing to notice here is that by a simple syntactic operation (transforming a number into a variable) we can generalize a concept. This is one way in which an agent generalizes concepts.

Turning Occurrences of a Variable into Different Variables

According to this rule, the expression [7] may be generalized to expression [8] by turning the two occurrences of the variable $?O_3$ in E_1 into two variables, $?O_{31}$ and $?O_{32}$:

E_1 =	$?O_1$	instance of	paper	[7]
		is authored by	$?O_3$	
	$?O_2$	instance of	paper	
		is authored by	$?O_3$	
	$?O_3$	instance of	professor	

E_2 =	$?O_1$	instance of	paper	[8]
		is authored by	$?O_{31}$	
	$?O_2$	instance of	paper	
		is authored by	$?O_{32}$	
	$?O_{31}$	instance of	professor	
	$?O_{32}$	instance of	professor	

E_1 may be interpreted as representing the set of sequences ($?O_1$, $?O_2$,$?O_3$, $?O_3$) expressing: "papers $?O_1$ and $?O_2$ authored by the same professor $?O_3$".

E_2 may be interpreted as representing the set of sequences ($?O_1$, $?O_2$,$?O_{31}$, $?O_{32}$) expressing: "papers $?O_1$ and $?O_2$ authored by the professors $?O_{31}$ and$?O_{31}$, respectively". In particular, $?O_{31}$ and$?O_{31}$, may represent the same professor. Therefore, the second set includes the first one, and the second expression is more general than the first one.

Climbing the Generalization Hierarchies

One can also generalize an expression by replacing a concept from its description with a more general concept, according to some generalization hierarchy. For instance, the expression [9]

E_1 =	$?O_1$	instance of	assistant professor	[9]
		has as employer	$?O_2$	
	$?O_2$	instance of	state university	

may be generalized to [10]

E_2 =	$?O_1$	instance of	professor	[10]
		has as employer	$?O_2$	
	$?O_2$	instance of	state university	

by replacing the concept assistant professor with the more general concept professor (see the generalization hierarchy in Figure 1).

The reverse operation, of replacing a concept with a less general one, leads to the specialization of an expression.

Dropping Conditions

The agent can also generalize a concept by dropping a condition, that is, by dropping a constraint that its instances must satisfy. For example, the expression [11]

E_1 =	$?O_1$	instance of	assistant professor	[11]
		has as employer	$?O_2$	
	$?O_2$	instance of	state university	

may be generalized to [12]

E_2 =	$?O_1$	instance of	assistant professor	[12]

by removing a constraint on the professor to be employed by a state university.

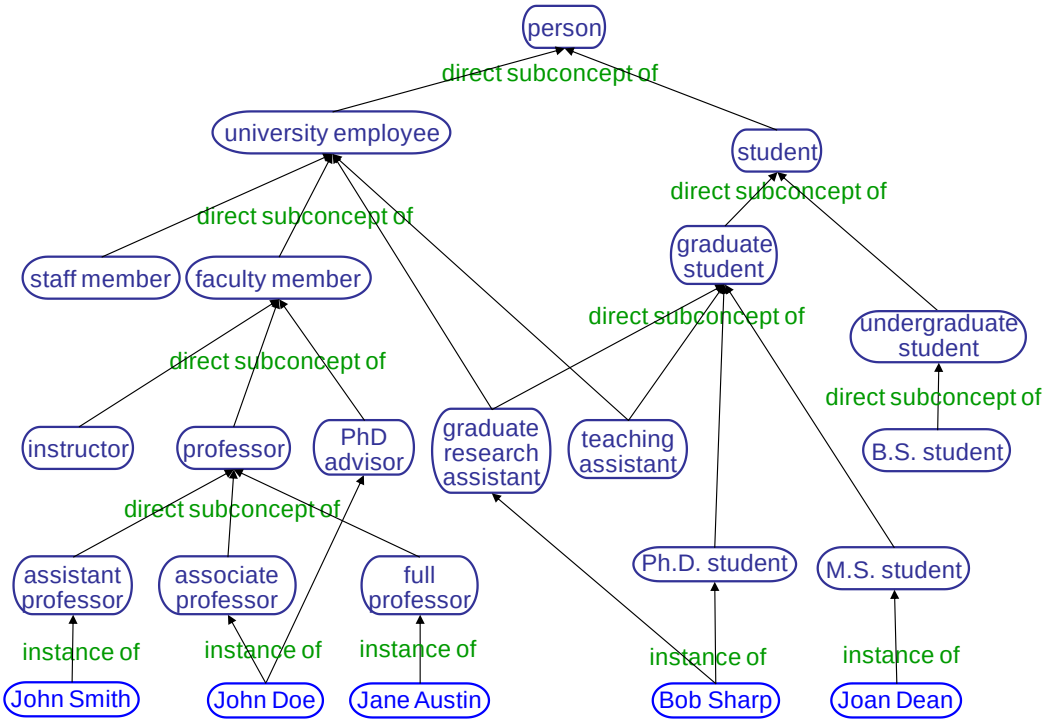


Figure 1. Sample generalization hierarchy.

Extending Intervals

A number may be generalized to an interval containing it. For example, the expression [15] (the set of professors with 55 publications) may be generalized to [16] (the set of professors with 50 to 60 publications) which, in turn, may be generalized to [17] (the set of professors with 25 to 75 publications).

E_1 =	$?O_1$	instance of	professor	[15]
		number of publications	55	
E_2 =	$?O_1$	instance of	professor	[16]
		number of publications	$?N_1$	
	$?N_1$	is in	[50 .. 60]	
E_3 =	$?O_1$	instance of	professor	[17]
		number of publications	$?N_1$	
	$?N_1$	is in	[25 .. 75]	

Extending Ordered Sets of Intervals

An ordered set of intervals may be regarded as an ordered generalization hierarchy, where the nodes are ordered from left to right, as illustrated in Figure 2.

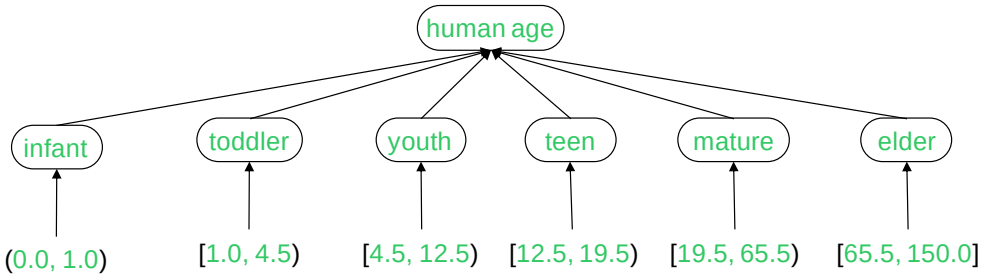


Figure 2. Ordered set of intervals as ordered generalization hierarchy.

Using such an ordered set of intervals, one may generalize the expression [18] (persons from youth to teens) to the expression [19] (persons from youth to mature), by replacing the symbolic interval "[youth, teen]" with the larger interval "[youth, mature]".

$E_1 = ?O_1$	instance of	person	[18]
	has as age	[youth, teen]	

$E_2 = ?O_1$	instance of	person	[19]
	has as age	[youth, mature]	

Extending Discrete Sets

An expression may also be generalized by replacing a discrete set with a larger set. For example, the expression [20] (the set of flags with white or red colors) may be generalized to the expression [21] (the set of flags with white or red or blue colors).

$E_1 = ?O_1$	instance of	flag	[20]
	has as component color	{white, red}	

$E_2 = ?O_1$	instance of	flag	[21]
	has as component color	{white, red, blue}	

Using Feature Definitions

This rule generalizes an expression containing a feature, such as, "A feature$_1$ B", by replacing A and B with the domain and the range of feature$_1$, respectively. This is illustrated by the generalization of expression [22] (professors who are experts in Computer Science) to expression [23] (professors who are experts in some area of expertise).

$E_1 =$	$?O_1$	instance of	professor	[22]
		is expert in	$?O_2$	
	$?O_2$	instance of	Computer Science	

E_2 = $?O_1$ instance of professor [23]
is expert in $?O_2$
$?O_2$ instance of area of expertise

In the above example, Computer Science was generalized to area of expertise. In general, a generalization rule indicates how one can generalize a concept through a simple syntactic transformation, without suggesting the actual generalization to perform, but only a set of possible generalization. The using feature definition rule actually suggests a generalization to perform, which will be useful during learning. In fact, it indicates the most general generalization that can be performed. Indeed $?O_2$ in [23] is the value of the feature is expert in. Therefore it has to be in the range of this feature which is area of expertise.

Using Inference Rules

Given an inference rule of the form "A → B", one may generalize an expression by replacing A with B. For example, using the theorem

$\forall X \forall Y ((X \text{ has as PhD advisor } Y) \rightarrow (X \text{ knows } Y))$

one may generalize the expression [24] (students and their PhD advisors)

E_1 = $?O_1$ instance of student [24]
has as PhD advisor $?O_2$
$?O_2$ instance of professor

to the expression [25] (students and the professors they know)

E_2 = $?O_1$ instance of student [25]
knows $?O_2$
$?O_2$ instance of professor

Indeed, by applying the above inference rule one may transform E_1 into the equivalent expression:

E'_1 = $?O_1$ instance of student
has as PhD advisor $?O_2$
knows $?O_2$
$?O_2$ instance of professor

Then, by dropping the relation knows, one generalizes E'_1 to E_2.

3. Types of Generalizations and Specializations

Up to this point we have only defined when a concept is more general than another concept. Learning agents, however, would need to determine generalizations of sets of examples and concepts. In the following we define some of these generalizations.

Generalization of Two Concepts

The concept C_g is a generalization of the concepts C_1 and C_2 if and only if C_g is more general than C_1 and C_g is more general than C_2.

An operational definition of this generalization is the following one: *If by applying generalization rules each of the concepts C_1 and C_2 can be transformed into the concept C_g, then the concept C_g is a generalization of C_1 and C_2.*

For example, the concept C [28] is a generalization of the concepts C_1 [26] and C_2 [27]. Indeed, by generalizing assistant professor to professor (through climbing the generalization hierarchy), by generalizing 10 to [10 .. 35] (through replacing a number with an interval containing it) and by dropping the condition "?O1 is employed by George Mason University", one generalizes C_1 to C. Similarly, by generalizing associate professor to professor and 35 to [10 .. 35], one generalizes C_2 into C.

C_1 =	$?O_1$	instance of	assistant professor	[26]
		number of publications	10	
		is employed by	George Mason University	
C_2 =	$?O_1$	instance of	associate professor	[27]
		number of publications	35	
C =	$?O_1$	instance of	professor	[28]
		number of publications	$?N_1$	
	$?N_1$	is in	[10 .. 35]	

In general, to build a generalization of two clauses one first applies the dropping condition rule to remove the unmatched features (and possibly even matched features). Then one applies other generalization rules to determine the generalizations of the matched feature values. Notice that there may be more than one generalization of two expressions.

In a similar way one can determine a generalization G of two expressions E_1 and E_2, each consisting of a conjunction of clauses corresponding to the same set of variables. G consists of the conjunction of the generalizations of some of the corresponding clauses in the two expressions E_1 and E_2.

There are usually more than one generalization of two concepts, as illustrated in Figure 3. Further distinctions between these generalizations are defined in the following section.

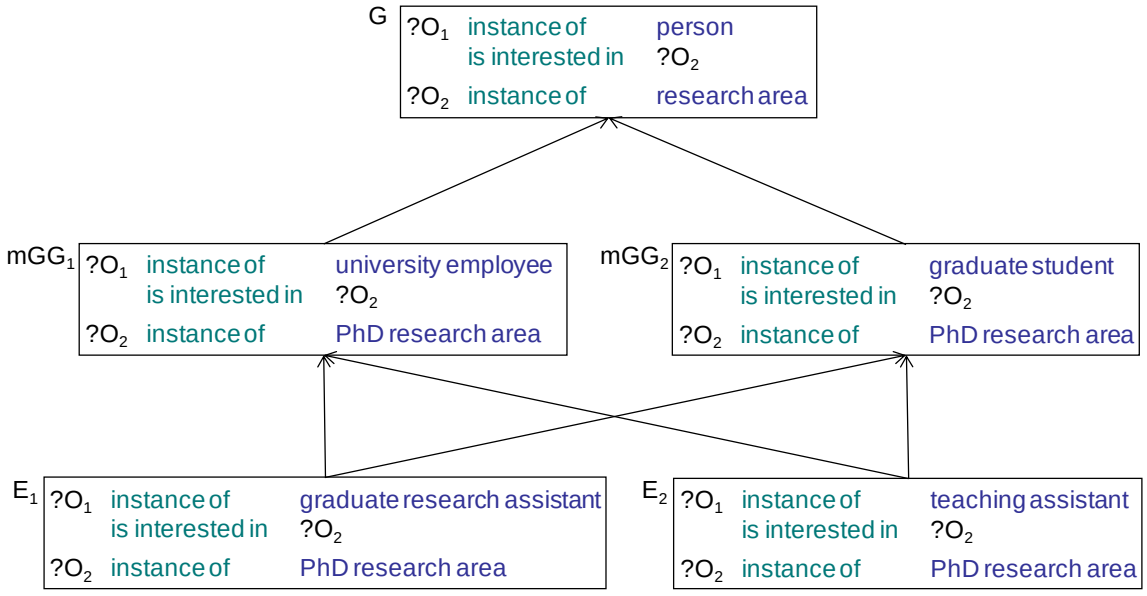


Figure 3. Several generalizations of the concepts E1 and E2

Minimally General Generalization of Two Concepts

*The concept G is a **minimally general generalization (mGG)** of A and B if and only if G is a generalization of A and B, and G is not more general than any other generalization of A and B.*

To determine a minimally general generalization of two clauses, one has to keep ALL the common features of the clauses and to determine a minimally general generalization of each of the matched feature values. In a similar way one determines the mGG of two conjunctions of clauses. One keeps ALL the matched clauses and determines the mGG of each pair of matched clauses. These procedures are correct if we assume that there are no other common features due to theorems. Otherwise, all the common features will have to first be made explicit by applying the theorems.

Notice, however, that there may be more than one mGG of two expressions. For instance, according to the generalization hierarchy from the middle of Figure 4, there are two mGG's of graduate research assistant and teaching assistant. They are university employee and graduate student. Consequently, there are two mGG's of E_1 and E_2 in Figure 1. They are mGG_1 and mGG_2. The generalization mGG_1 in Figure 1 was obtained by generalizing graduate research assistant and teaching assistant to university employee. mGG_2 was obtained in a similar fashion, except that graduate research assistant and teaching assistant have been generalized to graduate student. Neither of mGG_1 nor mGG_2 is more general than the other. However, G is more general than each of them.

Least General Generalization of Two Concepts

If there is only one minimally general generalization of two concepts A and B, then this generalization is called the **least general generalization (LGG)** of A and B.

Disciple agents employ minimally-general generalizations, also called maximally specific generalizations (Plotkin, 1970; Kodratoff & Ganascia, 1986). They also employ over-generalizations (Tecuci and Kodratoff, 1990; Tecuci, 1992; Tecuci 1998).

Specialization of Two Concepts

As has been mentioned in the previous section, the reverse of each generalization rule is a specialization rule. Therefore, for each of the above definitions of generalization there is a corresponding definition of a specialization.

The concept C_s is a specialization of the concepts C_1 and C_2 if and only if Cs is less general than C_1 and C_s is less general than C_2. For example, research professor is a specialization of researcher and professor.

An operational definition of this specialization is the following one: *If by applying specialization rules each of the concepts C_1 and C_2 can be transformed into the concept C_s, then C_s is a specialization of C_1 and C_2.*

Alternatively, if by applying generalization rules C_s can be transformed into C_1, and C_s can also be transformed into C_2, then C_s is a specialization of C_1 and C_2.

Figure 4 shows several specializations of two concepts G_1 and G_2.

Maximally General Specialization of Two Concepts

*The concept C is **a maximally general specialization (MGS)** of two concepts A and B if and only if C is a specialization of A and B and no other specialization of A and B is more general than C.*

The MGS of two clauses consists of the MGS of the matched feature-value pairs and all the unmatched feature-value pairs. This procedure assumes that no new caluse feature can be made explicit by applying theorems. Otherwise, one has first to make all the features explicit.

The MGS of two conjunctions of clauses C_1 and C_2 consists of the conjunction of the MGS of each of the matched clauses of C_1 and C_2 and all the unmatched clauses from C_1 and C_2.

Figure 4 shows several specializations of the concepts G_1 and G_2. MGS_1 and MGS_2 are two maximally general specializations of G_1 and G_2 because graduate research assistant and teaching assistant are two maximally general specializations of university employee and graduate student.

Notice also that in all the above definitions and illustrations, we have assumed that the clauses to be generalized correspond to the same variables. If this assumption is not satisfied, then one would need first to match the variables, and then to compute the generalizations. In general, this process is computationally expensive because one may need to try different matchings.

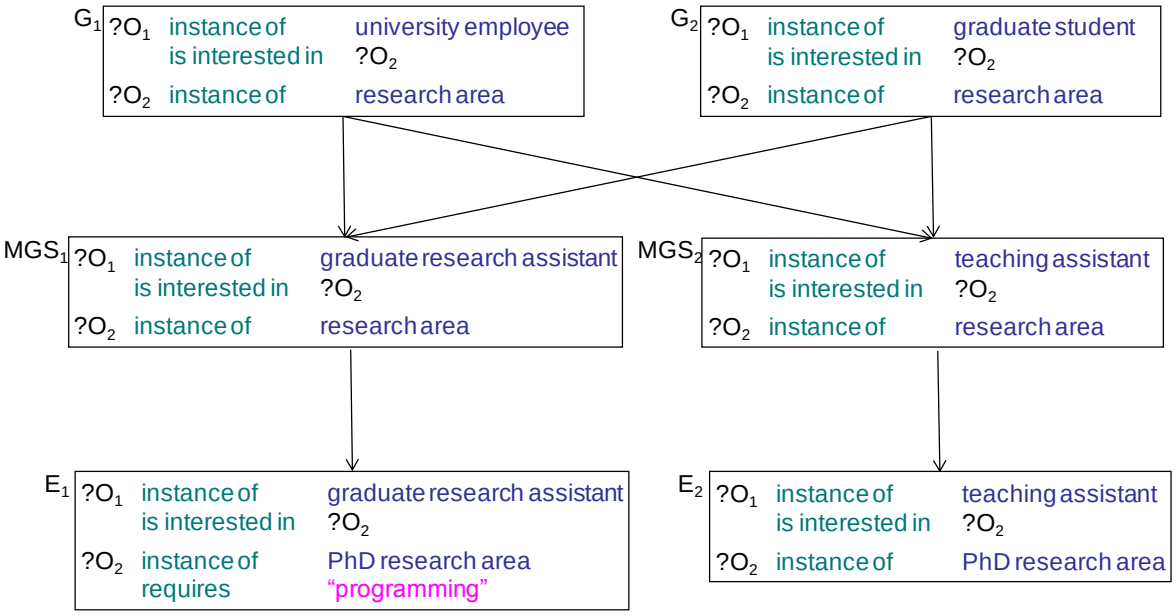


Figure 4. Several specializations of the concepts G1 and G2

4. Partially Learned Concepts

Disciple learns general concepts from examples and explanations (Tecuci, 1998; Tecuci et al., 2005; Tecuci et al., 2008). During the learning process it maintains a set of possible versions of the concept to be learned, called a **version space** (Mitchell, 1977; Mitchell, 1997; Tecuci, 1998). The concepts in this space are partially ordered, based on the generalization relationship. This means that a concept from this space can be obtained from another concept from the space by applying generalization or specialization rules. For that reason, the version space can be represented by an upper bound and a lower bound.

*The **upper bound** of the version space contains the most general concepts from the version space and the **lower bound** contains the least general concepts from the version space. Any concept which is more general than a concept from the lower bound and less general than a concept from the upper bound is part of the version space and may be the actual concept to be learned. Therefore a version space may be regarded as a **partially learned concept**.*

*The version spaces built by Disciple during the learning process are called **plausible version spaces** because their upper and lower bounds are generalizations based on an incomplete object ontology.* Therefore a plausible version space is only a plausible approximation of the concept to be learned, as illustrated in Figure 5.

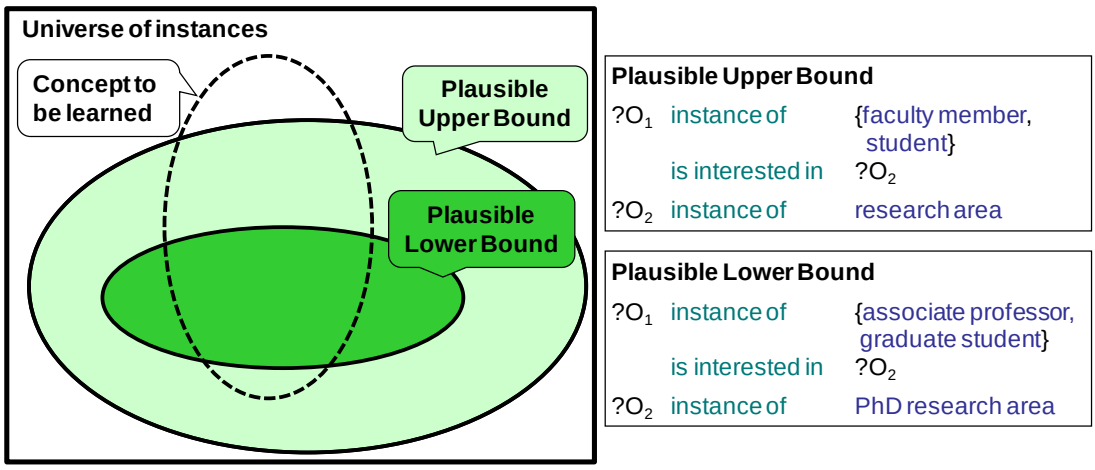


Figure 5.A plausible version space for a concept to be learned.

The plausible upper bound of the version space from the right hand side of Figure 5 contains two concepts: "faculty member interested in a research area" and "student interested in a research area" (see [29]).

$?O_1$	instance of	faculty member	[29]
	is interested in	$?O_2$	
$?O_2$	instance of	research area	

and

$?O_1$	instance of	student
	is interested in	$?O_2$
$?O_2$	instance of	research area

Similarly, the plausible lower bound of this version space contains two concepts, "associate professor interested in a PhD research area" and "graduate student interested in a PhD research area".

The concept to be learned (see Figure 5) is, ***as an approximation****, less general than* ***one*** *of the concepts from the plausible upper bound. This concept is also, again,* ***as an approximation****, more general than* ***any*** *of the concepts from the plausible lower bound.*

As Disciple encounters additional positive and negative examples of the concept to be learned, and as it understands it better, it generalizes and/or specializes the two bounds so that they converge toward one another and approximate better and better the concept to be learned. This behavior is different from that of the version spaces introduced by Mitchell (1978), where **one** of the concepts from the upper bound **is always more general** than the concept to be learned (and the upper bound is always specialized during learning), and **any** of the concepts from the lower bound **is always less general** than the concept to be learned (and the lower bound is always generalized during learning). The major difference is that the version spaces introduced by Mitchell (1978) are based on a complete representation space that includes the concept to be learned. On the contrary, the representation space

for Disciple is based on an incomplete and evolving object ontology. Therefore, Disciple addresses the more complex and more realistic problem of learning in the context of an evolving representation space.

The notion of plausible version space is fundamental to the knowledge representation, problem solving, and learning methods of Disciple because all the partially learned concepts are represented using this construct. For instance, a partially learned feature has its domain and range represented as plausible version space. A partially learned problem or rule is also represented as a plausible version space.

5. Examples and Exceptions of a Partially Learned Concept

*An entity belonging to the set of instances represented by a concept is called a **positive example** of the concept.*

*An entity that does not belong to the set of instances represented by a concept is called a **negative example** of the concept.*

Figure 6 shows the representation of a partially learned concept, consisting of a ***main plausible version space condition*** (in light and dark green) and ***an except-when plausible version space condition*** (in light and dark red).

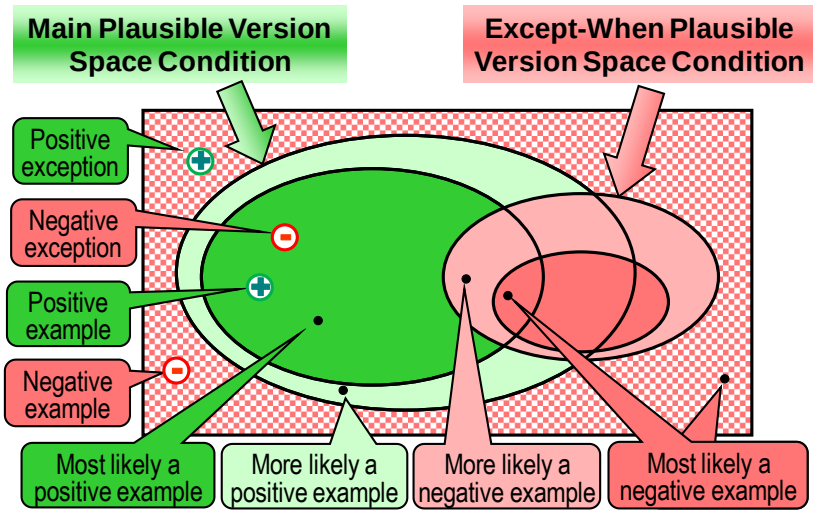


Figure 6.Examples and exceptions of a concept

A partially learned concept may have known positive and negative examples. For the other instances of the representation space one may estimate their nature based on their actual position with respect to the plausible bounds of the concept, as illustrated in Figure 6 and defined in the following.

An instance covered by the plausible lower bound of the main condition of a concept and not covered by any except when plausible version space condition is most likely a positive example of the concept.

An instance covered by the plausible upper bound of the main condition of a concept, but neither covered by the plausible lower bound of the main condition, nor by any except when plausible version space condition is likely to be a positive example of the concept.

An instance covered by the plausible lower bound of one of the except-when plausible version space conditions of a concept is most likely a negative example of the concept.

Finally, an instance covered by the plausible upper bound of an except when plausible version space condition of a concept is likely to be a negative example of the concept.

6. Partially Learned Features

A feature is characterized by a domain and a range (Tecuci and Boicu, 2008). *The **domain** is the concept that represents the set of objects that could have that feature. The **range** is the set of possible values of the feature.*

Most often, the domains and the ranges of the features are basic concepts from the object ontology. However, they could also be complex concepts of the form shown in [3]. Moreover, in the case of partially learned features they are plausible version spaces, as illustrated in Figure 7.

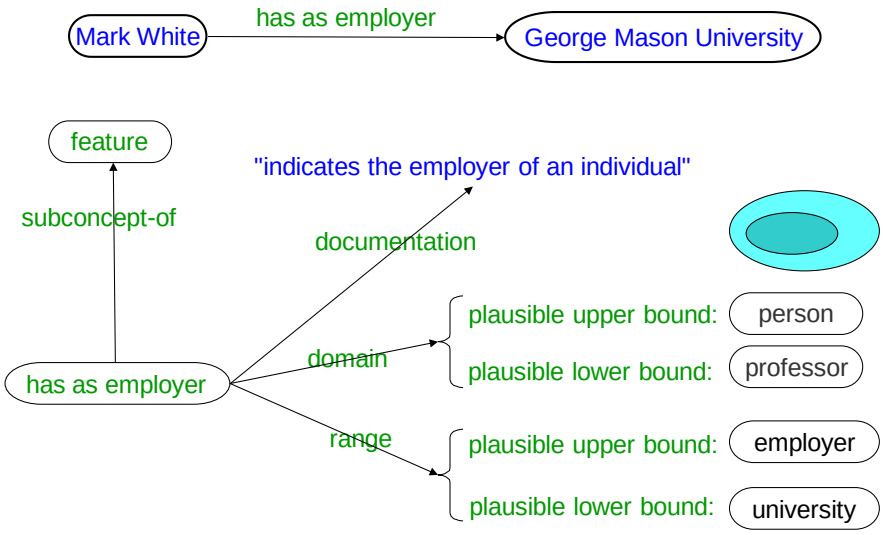


Figure 7.A feature and its partially learned definition.

7. Problems and Problem Definitions

An agent solves problems, represented as natural language patterns, such as [30], which is an example of a specific problem.

Assess whether John Doe is a potential PhD advisor for Bob Sharp. [30]

A problem pattern consists of natural language text (e.g. “Assess whether”), constants (e.g. the instance John Doe), and concepts (e.g. PhD advisor).

The definition of a problem is a pattern with variables and a precondition that the variables must satisfy. For example, the definition of the problem [30] is the one from [31].

Precondition [31]

?O_1 instance of faculty member

?O_2 instance of person

Name

Assess whether ?O_1 is a potential PhD advisor for ?O_2

The precondition is a concept which, in general, may have the form [3]. This precondition is a plausible version space in the case of partially learned problems.

The purpose of a problem's precondition is to ensure that the problem makes sense for each problem instantiation that satisfies it. For example, the following problem does not make sense:

"Assess whether 45 is a potential PhD advisor for Bob Sharp."

8. Rules

A Disciple-type agent solves problems by using a general "divide and conquer" approach that involves problem reduction and solution synthesis. In this paradigm, which is illustrated in Figure 8, a complex problem is solved by successively reducing it to simpler and simpler problems, finding the solutions of the simplest problems, and then successively combining these solutions, from the bottom up, until the solution of the initial problem is obtained.

In the illustration from Figure 8, the initial problem P_1 is reduced to the simpler problems P_{11}, … , P_{1n}. This means that the problem P_1 may be solved by solving the problems P_{11}, … , P_{1n}. Then P_{11} is reduced to P_{21}, … , P_{2m}. Then P_{2m} is reduced to P_{31}, … , P_{3p}. These problems are simple enough to find their solutions S_{31}, … , S_{3p}. These solutions are composed into S_{2m}, the solution of P_{2m}. Then the solutions S_{21}, … , S_{2m} of the problems P_{21}, … , P_{2m} are composed into S_{11}, the solution of P_{11}. Finally, the solutions S_{11}, … , S_{1n} are composed into S_1, the solution of the initial problem P_1.

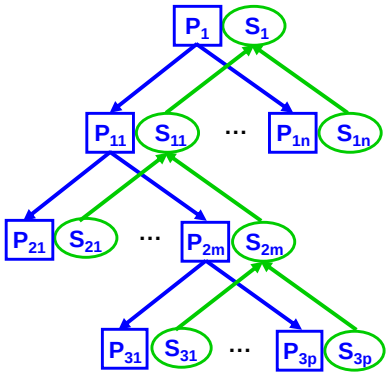


Figure 8. The problem reduction paradigm of problem solving

The reduction and synthesis operations are performed by applying problem reduction rules and solution synthesis rules.

A *problem reduction rule* has the form [32] and expresses how and under what conditions a generic problem can be reduced to one or several simpler generic problems.

IF <Problem> [32]
<Applicability condition>
THEN <Subproblem 1>
<Subproblem 2>
…
<Subproblem n>

The applicability condition is the concept representing the set of instances for which the reduction is correct. In the case of a partially learned rule, the applicability condition is a plausible version space. An example of such a rule is presented in Figure 9.

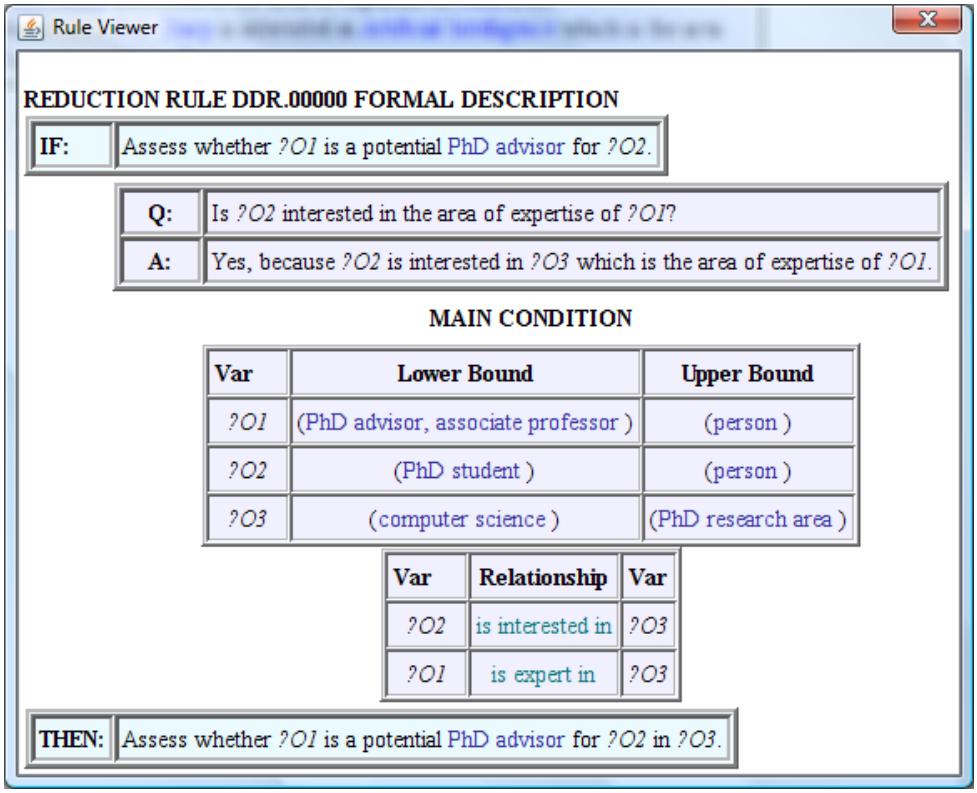


Figure 9. Partially learned problem reduction rule.

The solutions generated by a partially learned rule will have different degrees of plausibility, as indicated in Figure 10.

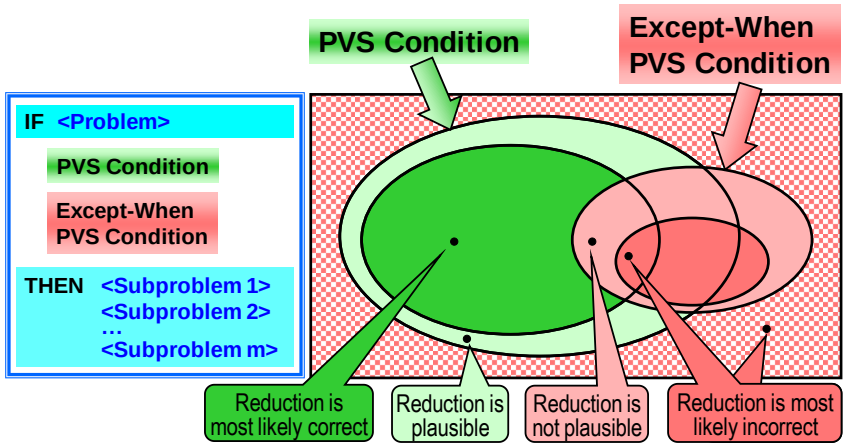


Figure 10. Plausible reasoning based on a partially learned rule.

A *solution synthesis rule* expresses how and under what conditions the solutions of generic subproblems can be combined into the solution of a generic problem.

9. Formal Definition of Generalization

Formal representation language for concepts

A knowledge representation language defines a syntax and semantics for expressing knowledge in a form that an agent can use. We define a formal representation language for concepts as follows:

- Let '$\mathcal{V}$' be a set of **variables**. For convenience in identifying variables, their names start with '?' as, for instance, **?X**. Variables are used to denote unspecified instances of concepts.
- Let '$\mathcal{C}$' be a set of **constants**. Examples of constants are the numbers (such as 5), strings (such as "programming"), symbolic probability values (such as very high) and instances. We define a **term** to be either a variable or a constant.
- Let '$\mathcal{F}$' be a set of **features**. The set F includes the domain independent features instance of, subconcept of, and direct subconcept of, as well as other domain specific features, such as is interested in.
- Let '$\mathcal{O}$' be an object ontology consisting of a set of concepts and instances defined using the clause representation [2] presented in Section 1, where the feature values (v_{i1} … v_{im}) are constants, concepts, or instances. That is, there are no variables in the definition of a concept or an instance from $\mathcal{O}$:

$?O_i$	instance of	$concept_j$
	$feature_{i1}$	v_{i1}
	. . .	
	$feature_{in}$	v_{im}

The concepts and the instances from O are related by the generalization relations instance of and subconcept of. O includes the concept object which represents all the instances from the application domain and is therefore more general than any other object concept.

- Let 'H' be the set of theorems and properties of the features, variables, and constants.

  Two properties of any feature are its domain and its range. Other features may have special properties. For instance, the relations instance of and subconcept of are transitive (Tecuci and Boicu, 2008). Also, a concept or an instance inherits the features of the concepts that are more general than it (Tecuci and Boicu, 2008).

- Let 'N' be a set of connectors. L includes the logical connectors AND ($\wedge$),OR ($\vee$) and NOT (Except When), the connectors '{' and '}' for defining alternative values of a feature, the connectors '[', ']', '(' and ')' for defining a numeric interval, the delimiter ',', and the **symbols** '**Plausible Upper Bound**', and '**Plausible Lower Bound**'.

We call the tuple $\mathcal{L}$ = (V, C, F, O, H, N) a *representation language* for concepts.

In the representation language $\mathcal{L}$, a concept is defined as indicated in section 1 (see [2] and [3]).

In the following sections we will provide a formal definition of generalization in the representation language $\mathcal{L}$, based on substitutions.

A substitution is a function $\sigma = (x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n)$, where each x_i (i=1,...,n) is a variable and each t_i (i=1,...,n) is a term. If l_i is an expression in the representation language $\mathcal{L}$, then σl_i is the expression obtained by substituting each x_i from l_i with t_i.

Term generalization

In the representation language $\mathcal{L}$, a term is a constant (e.g. number, string, symbolic interval, or instance) or a variable. An unrestricted variable ?X is more general than any constant and is as general as any other unrestricted variable (such as ?Y).

Clause generalization

Let us consider the concepts described by the following two clauses, C_1 and C_2, where v1, v2, v11, …, v2n are variables, c1 and c2 are concepts, and f11, …, f2n are feature names.

C_1 = v1 instance of c1
f11 v11
...
f1m v1m

C_2 =	v2	instance of	c2
		f21	v21
		...	
		f2n	v2n

We say that the clause C_1 is *more general than* the clause C_2 if there exists a substitution σ such that:

$\sigma v_1 = v_2$

$c_1 = c_2$

$\forall i \in \{1,...,m\}, \exists j \in \{1,...,n\}$ such that $f_{1i} = f_{2j}$ and $\sigma v_{1i} = v_{2j}$.

For example, the concept

$\mathbf{C_1}$ =	?x	instance of	student
		has as age	adult

is more general than the concept

$\mathbf{C_2}$ =	?y	instance of	student
		has as age	adult
		has as sex	female

Indeed, let σ = (**?X** ← **?Y**). As one can see, σC_1 is a part of C_2, that is, each feature of C_1 is also a feature of C_2. The first concept represents the set of all adult students, while the second one represents the set of all adult students that are females. Obviously the first set includes the second one, and therefore the first concept is more general than the second one.

Let us notice, however, that this definition of generalization does not take into account the theorems and properties of the representation language $\mathcal{L}$. In general one needs to use these theorems and properties to transform the clauses C_1 and C_2 into equivalent clauses C'_1 and C'_2 respectively. Then one shows that C'_1 is *more general than* C'_2. Therefore, the definition of the *more general than* relation in $\mathcal{L}$ is the following one:

A clause C_1 is *more general than* another clause C_2 if and only if there exist C'_1, C'_2, and a substitution σ, such that:

$C'_1 =_{\mathcal{L}} C_1$

$C'_2 =_{\mathcal{L}} C_2$

$\sigma v_1 =_{\mathcal{L}} v_2$

c_1 is more general than c_2 in $\mathcal{L}$

$\forall$ i$\in${1,...,m}, $\exists$ j$\in${1,...,n} such that $f'_{1i} =_{\mathcal{L}} f'_{2j}$ and $\sigma v'_{1i} =_{\mathcal{L}} v'_{2j}$.

In the following we will always assume that the equality is in $\mathcal{L}$ and we will no longer indicate this.

Exercise

Illustrate clause generalization with an example from the PhD Advisor Assessment domain.

BRU generalization

As discussed in section 1, a BRU (basic representation unit) is a conjunction of clauses. An example of BRU is the following one:

$?O_1$	instance of	course
	has as reading	$?O_2$
$?O_2$	instance of	publication
	has as author	$?O_3$
$?O_3$	instance of	professor

where, for notation convenience, we have dropped the AND connector between the clauses. Therefore, anytime there is a sequence of clauses, they are to be considered as being connected by AND.

Let us consider two concepts, A and B, defined by the following expressions

$A = A_1 \wedge A_2 \wedge ... \wedge A_n$

$B = B_1 \wedge B_2 \wedge ... \wedge B_m$

where each A_i (i = 1, ... ,n) and each B_j (j = 1, ... ,m) is a clause.

A is *more general than* B if and only if there exist A', B', and σ such that:

$A' = A, \quad A' = A'_1 \wedge A'_2 \wedge ... \wedge A'_p$

$B' = B, \quad B' = B'_1 \wedge B'_2 \wedge ... \wedge B'_q$

$\forall$ i$\in${1,...,p}, $\exists$ j$\in${1,...,q} such that $\sigma A'_i = B'_j$.

Otherwise stated, one transforms the concepts A and B, using the theorems and the properties of the representation language, so as to make each clause from A' more general than a corresponding clause from B'. Notice that some clauses from B' may be “left-over”, that is, they are matched by no clause of A', as in the following example.

Exercise

Illustrate BRU generalization with an example from the PhD Advisor Assessment domain.

Generalization of concepts with negations

By concept with negations we mean an expression of form [3]:

$$BRU \wedge \text{not } BRU_1 \wedge \ldots \wedge \text{not } BRU_p$$

where each BRU is a conjunction of clauses.

Let us consider two concepts with negations, A and B, defined by the following expressions

$$A = BRU_a \wedge \text{not } BRU_{a1} \wedge \ldots \wedge \text{not } BRU_{ap}$$
$$B = BRU_b \wedge \text{not } BRU_{b1} \wedge \ldots \wedge \text{not } BRU_{bq}$$

A is *more general than* B if and only if there exist A', B', and σ such that:

$A' = A$, $A' = BRU'_a \wedge \text{not } BRU'_{a1} \wedge \ldots \wedge \text{not } BRU'_{ap}$
$B' = B$, $B' = BRU'_b \wedge \text{not } BRU'_{b1} \wedge \ldots \wedge \text{not } BRU'_{bq}$
$\sigma BRU'_a$ *is more general than* BRU'_b
$\forall\, i \in \{1,...,p\}$, $\exists\, j \in \{1,...,q\}$ such that BRU'_{bj} *is more general than* $\sigma BRU'_{ai}$.

Exercise

Illustrate the generalization of concepts with negations by using an example from the PhD Advisor Assessment domain.

10. Substitutions and the Generalization Rules

One can use the definition of generalization based on substitution to prove that the generalization rules transform concepts into more general concepts.

As an illustration, let us consider the turning constants into variables generalization rule that transformed the expression E_1 (see [5]) into the expression E_2 (see [6]). E_2 is indeed a generalization of E_1 because $E_1 = \sigma E_2$, where σ=(**?N1**← **55**).

Exercise

Use the definition of generalization based on substitution to prove that each of the generalization rules discussed in Section x transforms a concept into a more general concept.

11. Exercises

1. What is a positive example of a concept? What is a negative example of a concept?
2. What is a generalization rule? What is a specialization rule? What is a reformulation rule?
3. Name all the generalization rules you know.

4. Briefly describe and illustrate with an example the “turning constants into variables” generalization rule.

5. Define and illustrate the dropping conditions generalization rule.

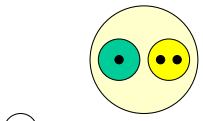
6. Define the following: a) a generalization of two concepts; b) a minimally general generalization of two concepts; c) the least general generalization of two concepts; d) the maximally general specialization of two concepts.

7. What is a negative exception? What is a positive exception?

8. Draw a picture representing a plausible version space, as well as a positive example, a negative example, a positive exception and a negative exception. Then briefly define each of these elements.

9. Consider the cells consisting of two bodies, each body having two attributes, color (which may be yellow or green) and number of nuclei (1 or 2). The relative position of the bodies is not relevant because they can move inside the cell. You should assume that any generalization of a cell is described as a single pair ((s t) (u v)).

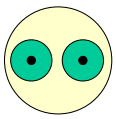
a) Indicate all the possible generalizations of the following cell, and the generalization relations between them:



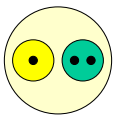
⊕ **((1 green) (2 yellow))**

b) Determine the number of the distinct sets of instances and the number of concept descriptions for this problem.

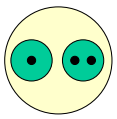
c) Given the following cell descriptions



((1 green) (1 green))



((1 yellow) (2 green))



((1 green) (2 green))

Determine the following minimal generalizations: g(E1, E2), g(E2, E3), g(E3, E1), g(E1, E2, E3)

12. References

Kodratoff, Y. and Ganascia, J-G. (1986). Improving the Generalization Step in Learning, In Michalski, R., Carbonell, J. and Mitchell, T. (editors), *Machine Learning: An Artificial Intelligence Approach, Vol. 2,* pp. 215-244. Morgan Kaufmann.

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

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Plotkin, G. D. (1970). A Note on Inductive Generalization. In Meltzer, B. and Michie, D. (editors), *Machine Intelligence* 5, pp. 165-179. Edinburgh: Edinburgh University Press.

Tecuci, G. (1992). Automating Knowledge Acquisition as Extending, Updating, and Improving a Knowledge Base, IEEE Trans. on Systems, Man and Cybernetics, 22, pp. 1444-1460.

Tecuci G. (1998). *Building Intelligent Agents,* Academic Press.

Tecuci G., Boicu M. (2008). A Guide for Ontology Development with Disciple. *Research Report 3,* Learning Agents Center, George Mason University.

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

Tecuci G., Boicu M., Marcu D., Barbulescu M., Boicu C., Le V., Hajduk T., Teaching Virtual Experts for Multi-Domain Collaborative Planning, *Journal of Software,* Volume 3, Number 3, pp. 38-59, March 2008.

Tecuci, G. and Kodratoff, Y. (1990). Apprenticeship Learning in Imperfect Theory Domains, In Kodratoff, Y. and Michalski, R. S. (editors), *Machine Learning: An Artificial Intelligence Approach, vol. 3,* pp. 514-551. Morgan Kaufmann.

13. Acknowledgements

This material is based on research partially sponsored by the Air Force Office of Scientific Research (FA9550-07-1-0268), the Air Force Research Laboratory (FA8750-04-1-0257), and the National Science Foundation (0750461).

Dorin Marcu, Cristina Boicu, Marcel Barbulescu, and Vu Le have contributed significantly to the recent versions of Disciple.

The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official polices or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the Air Force Research Laboratory or the U.S. Government.