Learning Composite Concepts

Patrick Lambrix and Pierpaolo Larocchia
Department of Computer and Information Science
Linköping University
S-581 83 Linköping, Sweden
e-mail: patla@ida.liu.se

Abstract

This paper proposes a framework to learn concepts from different kinds of observations. We define a language to describe meta-concepts, that represent the sets of possible concepts that can be the result of learning given a set of observations. The kinds of observations that we have studied are subsumption, membership and part-of. We exemplify the framework by showing how composite concepts can be learned in a specific description logic and we show that previous machine learning approaches in description logics can be reformulated in our framework.

1 Introduction

In [LM96] the problem of learning composite concepts was formulated in the framework of description logics. Description logics are restricted variants of first-order logic providing a form of logical bias that dates back to semantic networks. Some recent work investigates concept learning in the context of description logics [CH94a, CH94b, FP94, KM94, LM96] with the motivation that given the fact that first-order logic has been restricted in several ways for its use in the field of machine learning, description logics seem to make another good candidate as a learning framework. Also, having recognized the importance of part-of hierarchies in commonsense reasoning, researchers have started to incorporate part-of reasoning into description logics (see overview in e.g. [AFGP96, Lam96]). In [LM96] an attempt is made to combine machine learning in description logics and reasoning about part-of.

In this paper we extend this approach. We provide a framework for learning concepts described in a concept description language from different kinds of observations. These observations can be statements such as a specific concept is more general than the concept to learn or the concept to learn is part of another concept or an object belongs to the concept to learn or not. The different kinds of observations may interact with each other. Our framework consists of a meta-concept language that has similarities to description logics. The learning task is divided into two parts: a normalization part and a selection part. In the normalization part a meta-concept representing the observations (or learning examples) is transformed into one of three canonical forms. The resulting meta-concept represents the set of all possible concepts that satisfy the observations. We show the use of our framework by showing how concepts described in a particular description logic that allows for representation of part-of can be learned. As description logics also allow for representation of is-a and membership, we have a language that allows for observations of different kinds. We also show that previous approaches in machine learning in description logics can be reformulated in our framework.

The remainder of the paper is organized as follows. In section 2 we describe the language we use for describing concepts. The meta-concept language and the normalization procedure are defined in section 3. We describe our approach to learning in section 4. Related work is found in section 5. The paper concludes in section 6.

2 Concept Description Language

The concept description language that we use in this work is an extension of the description logic that was used in [LM96] and was based on the framework proposed in [PL94]. Description logics are languages tailored for expressing knowledge about concepts and concept hierarchies. They are usually given a Tarski style declarative semantics, which allows them to be seen as sub-languages of predicate logic. The main entities in description logics are concepts, roles and individuals. One starts with primitive concepts and roles, and can use the language constructs (such as intersection, role quantification etc.) to define new concepts and roles. Concepts can be considered as unary predicates which are interpreted as sets of objects of a domain whereas roles are binary predicates which are interpreted as binary relations between objects. Individuals are interpreted as objects. The basic reasoning tasks are classification and subsumption checking. The description logics in [PL94, LM96] were specifically designed to include the part-of relation. The languages introduce part names which are similar to roles to represent different part-of relations. The syntax of the language we use is as follows:

concept ::=
 $\top$
 | $\bot$
 | *atomic-concept*
 | (**and** *concept*$^+$)
 | (**all** *role concept*)
 | (**atleast** *number role*)
 | (**atmost** *number role*)
 | (**fills** *role individual*$^+$)
 | (**allp** *part-name atomic-concept*)
 | (**atleastp** *number part-name*)
 | (**atmostp** *number part-name*)
 | (**part-fills** *part-name individual*$^+$)
 | (**pp-constraint** *role part-name part-name*)
role ::= *identifier*
individual ::= *identifier*
atomic-concept ::= *identifier*
part-name ::= *identifier*
number ::= *non-negative-integer*

An interpretation of the language consists of a tuple $\langle \mathcal{D}, \varepsilon \rangle$, where $\mathcal{D}$ is the domain of individuals and ε the extension function. Let $\mathcal{C}$ be the set of atomic concepts, $\mathcal{R}$ the set of roles and $\mathcal{P}$ be the set of part names. Then, ε: $(\mathcal{C} \rightarrow \mathbf{2}^{\mathcal{D}}) \cup (\mathcal{R} \rightarrow \mathbf{2}^{\mathcal{D}\times\mathcal{D}}) \cup (\mathcal{P} \rightarrow \mathbf{2}^{\mathcal{D}\times\mathcal{D}})$. The semantics for the different terms in the language are defined as follows. For convenience we write $x \lhd_n y$ for $\langle x,y \rangle \in \varepsilon[n]$ where $n \in \mathcal{P}$.

$\varepsilon[\top] = \mathcal{D}$
$\varepsilon[\bot] = \emptyset$
$\varepsilon[(\textbf{and}\ A_1 \ldots A_m)] = \bigcap_{i=1}^{m} \varepsilon[A_i]$
$\varepsilon[(\textbf{all}\ r\ A)] = \{x \in \mathcal{D} \mid \forall\, y \in \mathcal{D}\text{: } \langle x,y\rangle \in \varepsilon[r] \rightarrow y \in \varepsilon[A]\}$
$\varepsilon[(\textbf{atleast}\ m\ r)] = \{x \in \mathcal{D} \mid \sharp\, \{y \in \mathcal{D} \mid \langle x,y\rangle \in \varepsilon[r]\} \geq m\}$
$\varepsilon[(\textbf{atmost}\ m\ r)] = \{x \in \mathcal{D} \mid \sharp\, \{ y \in \mathcal{D} \mid \langle x,y\rangle \in \varepsilon[r]\} \leq m\}$
$\varepsilon[(\textbf{fills}\ r\ i_1 \ldots i_m)] = \{x \in \mathcal{D} \mid$
$<\text{x},\varepsilon[i_1]> \in \varepsilon[r] \wedge \ldots \wedge <\text{x},\varepsilon[i_m]> \in \varepsilon[r]\}$
$\varepsilon[(\textbf{allp}\ n\ A)] = \{x \in \mathcal{D} \mid \forall\, y \in \mathcal{D}\text{: } y \lhd_n x \rightarrow y \in \varepsilon[A]\}$
$\varepsilon[(\textbf{atleastp}\ m\ n)] = \{x \in \mathcal{D} \mid \sharp\, \{y \in \mathcal{D} \mid y \lhd_n x\} \geq m\}$
$\varepsilon[(\textbf{atmostp}\ m\ n)] = \{x \in \mathcal{D} \mid \sharp\, \{y \in \mathcal{D} \mid y \lhd_n x\} \leq m\}$
$\varepsilon[(\textbf{part-fills}\ n\ i_1 \ldots i_m)] = \{x \in \mathcal{D} \mid$
$\varepsilon[i_1] \lhd_n \text{x} \wedge \ldots \wedge \varepsilon[i_m] \lhd_n \text{x}\}$
$\varepsilon[(\textbf{pp-constraint}\ r\ n_1\ n_2)] = \{x \in \mathcal{D} \mid$
$\forall\ y_1,y_2 \in \mathcal{D}\text{: } (y_1 \lhd_{n_1} x \wedge y_2 \lhd_{n_2} x) \rightarrow \langle y_1,y_2\rangle \in \varepsilon[r]\}$

Terminological axioms are used to introduce names for concepts and definitions of those concepts. Let A be a concept name (*identifier*) and C be a concept description, (*concept*), then terminological axioms can be of the form: A $\dot{\leq}$ C for introducing necessary conditions (primitive concepts), or A $\doteq$ C for introducing necessary and sufficient conditions (defined concepts).

For instance, the terminological axiom *standard-family* $\doteq$ (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring child*) (**atleastp** *2 offspring*) (**atmostp** *2 offspring*) (**pp-constraint** *married husband wife*) (**pp-constraint** *mother wife offspring*) (**pp-constraint** *father husband offspring*)) describes the concept of a *standard-family* which is defined as being composed of one *husband* part (that belongs to the concept *man*), one *wife* part (that belongs to the concept *woman*), and two *offspring* parts (that belong to the concept *child*) with the constraints that the *husband* is *married* to the *wife*, the *wife* is the *mother* of the *offspring* and the *husband* is the *father* of the *offspring*.

Assertional axioms are used for describing information about individuals. For instance, the assertional axiom *Jones* :: (**and** *standard-family* (**part-fills** *wife Jane*)) tells us that *Jones* is a standard family and *Jane* is the wife part in this family.

The terminological axioms form a Tbox while the assertional axioms form an Abox. A knowledge base consists then of a Tbox and an Abox.

As we are going to learn concepts, observations for our learning process are about relations between the concept to learn and other concepts and relations between the concept to learn and individuals. The relations that we consider are:

- *Subsumption between concepts.* The usual definition in description logics states that C_1 subsumes C_2 (notation: $C_2 \Rightarrow C_1$) iff $\varepsilon[C_2] \subseteq \varepsilon[C_1]$.
- *Part-of between concepts.* [LM96] If after normalization (**allp** n A) occurs in the definition of B, then A is a *direct n-part* of B. We say that A' is an *n-part* of B (notation: $A' \lhd_n B$) iff ($\exists A$: (A is a direct n-part of B) $\wedge$ ($A' \Rightarrow A$)).
  Thus part-of and subsumption interact.
- *Membership between an individual and a concept.* We say that an individual i belongs to a concept C with respect to a knowledge base $< Tbox, Abox >$ (notation: $i \longrightarrow C$) iff $\varepsilon[i] \in \varepsilon[C]$ for every model of the knowledge base.

3 Meta-Concept Language

The meta-language defines how we can represent sets of concepts and in the same way observations about a concept to learn. The atomic observations define which kinds of learning examples we can use. In our case the examples involve subsumption, part-of and membership. A meta-concept is a combination of atomic observations. The syntax for the language is as follows.

$< meta-concept >$ ::=
 MetaThing
 | **MetaNothing**
 | $< atomic-observation >$
 | (**and** $< atomic-observation >^+$)
$< atomic-observation >$ ::=
 | (**is-more-general-than** $< concept >$)
 | (**is-not-more-general-than** $< concept >$)
 | (**is-more-specific-than** $< concept >$)
 | (**is-not-more-specific-than** $< concept >$)
 | (**has-member** $< individual >$)
 | (**does-not-have-member** $< individual >$)
 | (**has-as** $< part-name >$ **part** $< concept >$)
 | (**does-not-have-as** $< part-name >$ **part** $< concept >$)
 | (**is-not-a** $< part-name >$ **part-of** $< concept >$)
 | (**is-a** $< part-name >$ **part-of** $< concept >$)
 | (**is-one-of** $< concept >^+$)
$< meta-concept-name >$ ::= $< identifier >$
$< concept >$ is defined in the concept representation language

$<$ *individual* $>$ ::= $<$ *identifier* $>$

An interpretation of the language consists of a tuple $\langle kb, Ext\rangle$, where kb is a knowledge base and Ext the extension function. Let $\mathcal{C}_{kb}$ be the set of all concepts that can be created using the constructors of the concept description language, the primitive concepts, roles and part names and the individuals in a knowledge base kb. Then, Ext maps meta-concepts to sub-sets of $\mathcal{C}_{kb}$. The semantics for the different terms in the language are defined as follows. The definitions rely on the availability of subsumption, part-of and membership.

Ext[(**and** A_1 ... A_h)] = $Ext[A_1] \cap ... \cap Ext[A_h]$
Ext[**MetaThing**] = $\mathcal{C}_{kb}$
Ext[**MetaNothing**] = $\emptyset$
Ext[(**is-more-general-than** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C \Rightarrow C^\star\}$
Ext[(**is-not-more-general-than** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C \not\Rightarrow C^\star\}$
Ext[(**is-more-specific-than** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C^\star \Rightarrow C\}$
Ext[(**is-not-more-specific-than** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C^\star \not\Rightarrow C\}$
Ext[(**has-member** i)] = $\{C^\star \in \mathcal{C}_{kb} \mid i \longrightarrow C^\star\}$
Ext[(**does-not-have-member** i)] = $\{C^\star \in \mathcal{C}_{kb} \mid i \not\longrightarrow C^\star\}$
Ext[(**has-as** n **part** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C \lhd_n C^\star\}$
Ext[(**does-not-have-as** n **part** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C \ntriangleleft_n C^\star\}$
Ext[(**is-a** n **part-of** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C^\star \lhd_n C\}$
Ext[(**is-not-a** n **part-of** C)] = $\{C^\star \in \mathcal{C}_{kb} \mid C^\star \ntriangleleft_n C\}$
Ext[(**is-one-of** C_1 ... C_h)] =
$\{C^\star \in \mathcal{C}_{kb} \mid C_1 \equiv C^\star \vee ... \vee C_h \equiv C^\star\}$

Observational axioms are used to introduce names for meta-concepts and definitions of those meta-concepts. Let MA be a meta-concept name (*identifier*) and MC be a meta-concept description, ($meta - concept$), then observational axioms are of the form MA $\doteq$ MC.

3.1 Normalization of Meta-Concepts

Given the language above we can rewrite every meta-concept description into an equivalent meta-concept description that is of one of three canonical forms. Two meta-concepts are equivalent if their extensions are the same for all interpretations. These canonical forms are:

(a) (**and**
(**is-more-specific-than** G)
(**is-more-general-than** S)
(**is-not-more-specific-than** G_1) ...
(**is-not-more-specific-than** G_h)
(**is-not-more-general-than** S_1) ...
(**is-not-more-general-than** S_k)
(**has-as** n_1 **part** D_1) ...
(**has-as** n_l **part** D_l))
(**does-not-have-as** n'_1 **part** D'_1) ...
(**does-not-have-as** n'_m **part** D'_m))
with $h, k, l, m \geq 0$
(b) (**is-one-of** C_1 ... C_l)
with l>0
(c) **MetaNothing**

The normalization uses a number of rules involving the properties of subsumption, part-of, membership and their interactions. The rules use some functions defined on concepts. The first function is a generalization function and calculates for two given concepts the least common subsumer (LCS). In a concept description language where disjunction is available the LCS of two concepts would just be their disjunction. In our language we use an extended version of the LCS defined in [LM96] which in itself was based on [CBH92]. The greatest common subsumee (GCS) of two concepts is in our language defined as follows: GCS(C_1,C_2) = (**and** C_1 C_2). GCS is used for specializing concepts. The DOM function allows for retrieving the domain for a particular part name for a concept. The UPDATE function allows for replacing the domain of a part name for a concept with another domain.

We have divided the normalization rules into a number of categories. The *initial rephrasing rules* translate one kind of observation into another kind of observation. For instance, it is easy to show that (**is-a** n **part-of** C) is equivalent to (**is-more-specific-than** DOM(n,C)).

The *single type synthesis rules* rewrite combinations of observations of the same kind into one observation of this kind. For instance, we can show that (**and** (**is-more-general-than** C_1) (**is-more-general-than** C_2)) is equivalent to (**is-more-general-than** LCS(C_1,C_2)), thereby generalizing the more specific concepts. Similarly, (**and** (**is-more-specific-than** C_1) (**is-more-specific-than** C_2)) is equivalent to (**is-more-specific-than** GCS(C_1,C_2)), thereby specializing the more general concepts.

The *interaction synthesis rules* allow for rewriting combinations of different kinds of observations. For instance, it can be shown that (**and** (**is-more-general-than** C_1) (**has-as** n **part** C_2)) can be rewritten as (**and** (**is-more-general-than** UPDATE(C_1,n,D) (**has-as** n **part** D)) where D is defined as LCS(DOM(n,C_1),C_2). In this case we have generalized a more specific concept as well as generalized a more specific part.

The *singleton rephrasing rules* allow for deciding when we can add a new **is-one-of** observation. For instance, if we have both (**is-more-general-than** C) and (**is-more-specific-than** C) the extension of the meta-concept must be the singleton $\{C\}$ and thus we can remove these observations and add (**is-one-of** C).

Finally, we have also rules that check whether the meta-concept is incoherent, i.e. equivalent to **MetaNothing**. An example of such a rule is: if C_2 does not subsume C_1 then (**and** (**is-more-general-than** C_1) (**is-more-specific-than** C_2)) is equivalent to **MetaNothing**.

For more rules, proofs of the rules and examples of all these categories we refer to [Lar96].

In general all the normalization rules should be applied until no more rule can be applied. However, we can show that for our language the algorithm below computes the canonical form of a meta-concept. In every step of the algorithm we remove an observation, generalize a more specific concept, specialize a more general concept, or find a contradiction. Given the fact that we have a finite number of observations, and the fact that a finite number of observations only allows for a finite number of LCS-generalizations and GCS-specializations, the algorithm terminates.

input meta-concept M
$M_1 :=$ apply *initial rephrasing rules* to M
$M_2 :=$ apply *single type synthesis rules* to M_1
if *singleton rephrasing rule* is applicable
then
  $M_3 :=$ apply the rule to M_2
    (M_3 contains now (**is-one-of** C))
  if C satisfies the other observations
  then return (**is-one-of** C)
  else return MetaNothing
else
  $M_3 :=$ apply *interaction synthesis rules* to M_2
  if *singleton rephrasing rule* is applicable
  then
    $M_4 :=$ apply the rule to M_3
      (M_4 contains now (**is-one-of** C))
    if C satisfies the other observations
    then return (**is-one-of** C)
    else return MetaNothing
  else
    if there is an inconsistency
    then return MetaNothing
    else return M_4

3.2 Example

In the following we show how the normalization of a meta-description works. We assume that C^* is a meta-concept and that we have the following observations. The definitions of all the concepts used in the observations are defined in the knowledge base.

- All concepts in C^* are more general than *standard-family-with-adolescent-boys* with *standard-family-with-adolescent-boys* $\doteq$ (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring adolescent-boy*) (**atleastp** *2 offspring*) (**atmostp** *2 offspring*) (**pp-constraint** *married husband wife*) (**pp-constraint** *mother wife offspring*) (**pp-constraint** *father husband offspring*)).
- All concepts in C^* are more general than *standard-family-with-adolescent-girls* with *standard-family-with-adolescent-girls* $\doteq$ (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring adolescent-girl*) (**atleastp** *2 offspring*) (**atmostp** *2 offspring*) (**pp-constraint** *married husband wife*) (**pp-constraint** *mother wife offspring*) (**pp-constraint** *father husband offspring*)).
- All concepts in C^* are more specific than *family-with-2-children* with *family-with-2-children* $\doteq$ (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring child*) (**atleastp** *2 offspring*) (**atmostp** *2 offspring*)).
- All concepts in C^* are more specific than *married-family-with-children* with *married-family-with-children* $\doteq$ (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring child*) (**atleastp** *1 offspring*) (**pp-constraint** *married husband wife*) (**pp-constraint** *mother wife offspring*) (**pp-constraint** *father husband offspring*)).
- *Young-child* is an *offspring*-part of all concepts in C^*.

These observations give us the following definition for the meta-concept C^*.

$C^* \stackrel{.}{\equiv}$ (**and**
  (**is-more-general-than**
    standard-family-with-adolescent-boys)
  (**is-more-general-than**
    standard-family-with-adolescent-girls)
  (**is-more-specific-than** *family-with-2-children*)
  (**is-more-specific-than** *married-family-with-children*)
  (**has-as** *offspring* **part** *young-child*))

Following the algorithm the first normalization rules that we apply are the initial rephrasing rules. In this example there are no such rules applicable. In the next step we apply the single type synthesis rules. We can apply the rules for **is-more-general-than** and **is-more-specific-than** that were given as examples in section 3.1. We therefore need to compute the following: LCS(*standard-family-with-adolescent-boys,standard-family-with-adolescent-girls*) = (*standard-family-with-adolescent-children* $\doteq$) (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring adolescent-child*) (**atleastp** *2 offspring*) (**atmostp** *2 offspring*) (**pp-constraint** *married husband wife*) (**pp-constraint** *mother wife offspring*) (**pp-constraint** *father husband offspring*)). We assume here that for our knowledge base LCS(*adolescent-girl,adolescent-boy*) = *adolescent-child*.[1]
We also need to compute the GCS of two concepts: GCS(*married-family-with-children,family-with-2-children*) = (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring child*) (**atleastp** *2 offspring*) (**atmostp** *2 offspring*) (**pp-constraint** *married husband wife*) (**pp-constraint** *mother wife offspring*) (**pp-constraint** *father husband offspring*)). This is exactly the definition of *standard-family* as given in section 2.
Applying the rules gives us then:

$C^* \stackrel{.}{\equiv}$ (**and**
  (**is-more-general-than**
    standard-family-with-adolescent-children)
  (**is-more-specific-than** *standard-family*)
  (**has-as** *offspring* **part** *young-child*))

We cannot apply a singleton rephrasing rule, such that we go on with the interaction synthesis rules. The

[1] This would have to have been derived from the definitions of *adolescent-girl* and *adolescent-boy* in our knowledge base.

interaction synthesis rule given in section 3.1 allows for rewriting the meta-concept. To apply the rule we first need to compute DOM(*offspring*,*standard-family-with-adolescent-children*) = *adolescent-child*. Then we need to find the LCS of this concept and *young-child*. We assume that LCS(*adolescent-child*,*young-child*) = *child*. Updating the *offspring*-part of *standard-family-with-adolescent-children* gives us: UPDATE(*standard-family-with-adolescent-children*,*offspring*,*child*) = (**and** (**allp** *husband man*) (**atleastp** *1 husband*) (**atmostp** *1 husband*) (**allp** *wife woman*) (**atleastp** *1 wife*) (**atmostp** *1 wife*) (**allp** *offspring child*) (**atleastp** *2 offspring*) (**atmostp** *2 offspring*) (**pp-constraint** *married husband wife*) (**pp-constraint** *mother wife offspring*) (**pp-constraint** *father husband offspring*)), which is the definition of *standard-family* as given in section 2. Applying the rule gives then the following.

$C^* \doteq$ (**and**
(**is-more-general-than** *standard-family*)
(**is-more-specific-than** *standard-family*)
(**has-as** *offspring* **part** *young-child*))

Now we can apply a singleton rephrasing rule and get:

$C^* \doteq$ (**and**
(**is-one-of** *standard-family*)
(**has-as** *offspring* **part** *young-child*))

There are no inconsistencies such that the final result is:

$C^* \doteq$ (**is-one-of** *standard-family*)

4 Learning

The learning task that we want to tackle is formulated as follows:

Given:
- a knowledge base expressed in a specific description logic
- observations expressed in an observation language (the observations may belong to different kinds of observations and may interact with each other)
- a selection criterion
Find:
a concept description that satisfies the observations and the selection criterion.

The learning of concepts proceeds in a number of steps. All background information is defined in a knowledge base <Tbox,Abox>. In the knowledge base all concepts and individuals that are used are defined. Other information may be available as well. The steps are the following:

1. Translate learning examples (observations) into a meta-concept description.
2. Normalize the meta-concept description.
3. Select a concept that is an instance of the normalized meta-concept description.

If we want to find all possible concepts that satisfy the observations, we can stop after step 2. Step 3 is used to retrieve one concept satisfying the observations. In the case the normalized meta-concept is **MetaNothing**, there is no solution to the learning problem. In the case we have (**is-one-of** C) the only concept that we can retrieve is C. In the other case there may be different possibilities. One selection criterion could require the most specific concept that satisfies the observations. This concept is found in the normalized meta-concept in the **is-more-general-than** observation. Another selection criterion may require the most general concept that satisfies the observations. This concept is found in the normalized meta-concept in the **is-more-specific-than** observation.

In the remainder of this section we show how the LCSLearn algorithm in [CH94b] applied to our language can be seen as a special case of our learning algorithm. In LCSLearn the positive and negative examples are subsumption statements that can be reformulated in our framework using **is-more-general-than** and **is-not-more-general-than** observations respectively. If there are no positive examples LCSLearn returns $\bot$. Otherwise, if there are inconsistent observations the algorithm aborts with failure. In the other case the LCS of all positive examples is returned. The fact that LCSLearn returns the LCS of the positive examples can be seen as having a selection criterion that requires the most specific concept that satisfies the observations.

In our framework LCSLearn is translated into the algorithm below. The single type synthesis during the normalization of the meta-concept computes the LCS of the positive examples and stores this LCS in the **is-more-general-than** observation. The case where the normalized meta-concept is of the form (**is-one-of** C) cannot occur with observations that are only of the kinds **is-more-general-than** and **is-not-more-general-than**.

1. Translate observations into meta-concept M
2. Normalize M
3. **if** the normalized meta-concept
  is of the form **MetaNothing**
 then return failure
 else if the normalized meta-concept does not
  contain a **is-more-general-than** observation
 then return $\bot$
 else return the concept
  in the **is-more-general-than** observation

5 Related Work

The work that is closest to our work is [LM96]. The authors propose a learning algorithm that learns composite concepts. The language that was proposed is a sub-set of our language. The learning examples that were proposed also included the *module* relation that is a relation with part-of intuition. Our work can be seen as an extension of [LM96] by introducing a framework in which the problem of [LM96] can be reformulated. Our meta-concept language also gives a clear and intuitive way to describe learning examples and manipulate them. Further, our learning algorithm distinguishes between the normalization phase and the selection phase. This distinction was not clear in [LM96].

In [CH94a, CH94b] concepts are learned in the description logic system CLASSIC (e.g. [BBMR89]). The language is more expressive than the standard part of the language (i.e. without constructs for part-of) we use, but there are no constructs to deal with part-of. Learning examples are concepts. A concept is a positive example if it is subsumed by the concept to learn and negative otherwise. It is shown that a restriction of CLASSIC, called C-CLASSIC is PAC-learnable. The algorithm is based on the LCS version in [CBH92]. Learning from individuals is done by generalizing the individuals into concepts. A number of experiments have been performed.

Other work that uses CLASSIC in a machine learning setting is described in [FP94]. The authors show that CLASSIC sentences are learnable in polynomial time in the exact learning model using equivalence and membership queries (e.g. [Ang88]). The membership queries in the description logic setting are actually subsumption queries. They also show that both kinds of queries are necessary for efficient learning.

KLUSTER [KM94] starts from a knowledge base of individuals linked together by roles. The first step in KLUSTER's learning is to build a basic taxonomy which is expressed in a sub-language of the description logic system BACK (e.g [Neb90]). This sub-language is the same as the standard part of our language except for the fact that KLUSTER allows role constructs. The learning problem for KLUSTER is to build discriminating concept definitions starting from the basic taxonomy.

6 Discussion and Conclusion

In this paper we have presented a framework that can be used for learning from different kinds of observations. This is done by introducing a meta-concept description language. We have shown how we can rewrite descriptions in this language into one of three canonical forms. The learning is performed by rewriting observations as a meta-concept, normalize this meta-concept and select an instance of the normalized form. We have shown that other approaches can be reformulated in our framework.

References

[Ang88] D. Angluin. Queries and Concept Learning. *Machine Learning*, 2:319-342, 1988.

[AFGP96] A. Artale, E. Franconi, N. Guarino and L. Pazzi. 'Part-Whole Relations in Object-Centered Systems: An Overview', *Data and Knowledge Engineering*, 20(3):347-383, 1996.

[BBMR89] A. Borgida, R. Brachman, D. McGuinness and L. Resnick. CLASSIC : A Structural Data Model for Objects'. *Proceedings of the International Conference on Management of Data - SIGMOD 89*, pp 59-67, 1989.

[CBH92] W. Cohen, A. Borgida and H. Hirsh. Computing Least Common Subsumers in Description Logics. *Proceedings of the National Conference on Artificial Intelligence - AAAI 92*, pp 754-760, 1992.

[CH94a] W. Cohen and H. Hirsh. The Learnability of Description Logics with Equality Constraints. *Machine Learning*, 17:169-199, 1994.

[CH94b] W. Cohen and H. Hirsh. Learning the CLASSIC Description Logic: Theoretical and Experimental Results. *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference - KR 94*, pp 121-133, 1994.

[FP94] M. Frazier and L. Pitt. CLASSIC Learning. *Proceedings of the International Conference on Computational Learning Theory - COLT 49*, pp 23-34, 1994.

[KM94] J.-U. Kietz and K. Morik. A Polynomial Approach to the Constructive Induction of Structural Knowledge. *Machine Learning*, 14:193-217, 1994.

[Lam96] P. Lambrix. *Part-Whole Reasoning in Description Logics*, Ph.D. thesis, Department of Computer and Information Science, Linköping University, Sweden, 1996.

[LM96] P. Lambrix and J. Maleki. Learning Composite Concepts in Description Logics: A First Step. *Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems - ISMIS 96*, LNAI 1079, pp 68-77, 1996.

[Lar96] P. Larocchia. *Learning Composite Concepts in Description Logics*, M.Sc. Thesis, Department of Computer and Information Science, Linköping University, Sweden. LiTH-IDA-Ex-9657.

[Neb90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, LNAI 422, Springer-Verlag, 1990.

[PL94] L. Padgham and P. Lambrix. A Framework for Part-of Hierarchies in Terminological Logics. *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference - KR 94*, pp 485-496, 1994.