Learning composite concepts in description logics: A first step

Patrick Lambrix and Jalal Maleki

Linköping University Post Print

N.B.: When citing this work, cite the original article.

Original Publication:

Patrick Lambrix and Jalal Maleki, Learning composite concepts in description logics: A first step, 1996, Foundations of Intelligent Systems, 68-77.

http://dx.doi.org/10.1007/3-540-61286-6_132

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-107632>

Learning Composite Concepts in Description Logics: A First Step

Patrick Lambrix and Jalal Maleki

Department of Computer and Information Science
Linköping University
S-581 83 Linköping, Sweden
email: {patla,jalma}@ida.liu.se

Abstract. This paper proposes the use of description logics as a representational framework for learning composite concepts. 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 these formalisms. Also, having recognized the importance of part-whole hierarchies in commonsense reasoning, researchers have started to incorporate part-of reasoning into description logics. In our approach we represent composite concepts in such a formalism. On one hand we have a relatively rich representation language with an infinite space of possible concepts. On the other hand we have special constructs for handling part-of relations that can be used in the learning algorithm to reduce the overall search space.

This is a preprint of the paper published by Springer:

Lambrix P, Maleki J, Learning Composite Concepts in Description Logics: A First Step, Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems - ISMIS96, LNAI 1079, 68-77, Zakopane, Poland, 1996. *The final publication is available at www.springerlink.com. doi: 10.1007/3-540-61286-6_132*

1 Introduction

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. 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 individuals whereas roles are binary predicates which are interpreted as binary relations between individuals. The basic reasoning tasks are classification and subsumption checking.

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 restriction as a learning framework. However, it is only recently that learning algorithms are being developed within this framework [3, 4, 7].

There is also a growing recognition that part-whole hierarchies are a very general form of representation, widely used by humans in commonsense reasoning. Consequently, description logics are extended to integrate part-of reasoning with classification (e.g. [6, 10, 12]).

In this paper we propose learning algorithms to learn composite concepts in a particular description logic which is specially tailored to handle part-of reasoning. On one hand we have a highly expressive representation framework with an infinite space of possible concepts. On the other hand, we have special constructs for handling part-of which can be used in the learning algorithms to reduce the overall search space.

2 The Framework

In this work we take as our framework a relatively simple description logic [10] with only unstructured roles and a limited number of constructs, but which was specifically designed to include the part-of relation. The language introduces part names which are similar to roles to represent different part-of relations. Further, the language allows specification of constraints between the parts of a composite concept. In the remainder of this section we briefly describe the *Tbox* language, the necessary notions about part-of and finally the *Abox* language .

2.1 Tbox: Is-a

<i>concept</i>	::=	<i>role</i>	::= <i>identifier</i>
$\top$		<i>atomic-concept</i>	::= <i>identifier</i>
$\perp$		<i>part-name</i>	::= <i>identifier</i>
<i>atomic-concept</i>		<i>number</i>	::= <i>non-negative-integer</i>
(and <i>concept</i> ⁺)			
(all <i>role concept</i>)			
(atleast <i>number role</i>)			
(atmost <i>number role</i>)			
(part <i>part-name atomic-concept</i>)			
(parts <i>number part-name</i>)			
(pp-constraint <i>role part-name part-name</i>)			

The language we use to express terminological knowledge is defined as above [10]. *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: $A \leq C$ for introducing necessary conditions (primitive concepts), or $A \doteq C$ for introducing necessary and sufficient conditions (defined concepts).

A *terminology (Tbox)* *T* is a finite set of terminological axioms with the additional restrictions that (i) every concept name used must appear exactly once on the left hand side of a terminological axiom, (ii) all concepts must be defined (appear on the left hand side) before they are used, and (iii) *T* must not contain cyclic definitions directly or indirectly, via either $\doteq$, $\leq$, a **part** construct or any combination of these.

The terminological axiom *standard-family* $\doteq$ (**and** (**part** *husband* *man*) (**parts** 1 *husband*) (**part** *wife* *woman*) (**parts** 1 *wife*) (**part** *offspring* *child*) (**parts** 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 a part, *husband*, (that belongs to the concept *man*), a part, *wife*, (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*.

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 $\mathcal{P}$ be the set of part name names, $\mathcal{C}$ be the set of atomic concepts, and $\mathcal{R}$ the set of role names. Then, $\varepsilon: (\mathcal{P} \rightarrow 2^{\mathcal{D} \times \mathcal{D}}) \cup (\mathcal{R} \rightarrow 2^{\mathcal{D} \times \mathcal{D}}) \cup (\mathcal{C} \rightarrow 2^{\mathcal{D}})$.

The semantics for the different terms in the language are defined as follows [10]. For convenience we write $x \triangleleft_n y$ for $\langle x, y \rangle \in \varepsilon[n]$ where $n \in \mathcal{P}$.

$$\begin{aligned} \varepsilon[\top] &= \mathcal{D} \\ \varepsilon[\perp] &= \emptyset \\ \varepsilon[(\mathbf{and} \ A_1 \ \dots \ A_m)] &= \bigcap_{i=1}^m \varepsilon[A_i] \\ \varepsilon[(\mathbf{all} \ r \ A)] &= \{ x \in \mathcal{D} \mid \forall y \in \mathcal{D}: \langle x, y \rangle \in \varepsilon[r] \rightarrow y \in \varepsilon[A] \} \\ \varepsilon[(\mathbf{atleast} \ m \ r)] &= \{ x \in \mathcal{D} \mid \# \{ y \in \mathcal{D} \mid \langle x, y \rangle \in \varepsilon[r] \} \geq m \} \\ \varepsilon[(\mathbf{atmost} \ m \ r)] &= \{ x \in \mathcal{D} \mid \# \{ y \in \mathcal{D} \mid \langle x, y \rangle \in \varepsilon[r] \} \leq m \} \\ \varepsilon[(\mathbf{part} \ n \ A)] &= \{ x \in \mathcal{D} \mid \forall y \in \mathcal{D}: y \triangleleft_n x \rightarrow y \in \varepsilon[A] \} \\ \varepsilon[(\mathbf{parts} \ m \ n)] &= \{ x \in \mathcal{D} \mid \# \{ y \in \mathcal{D} \mid y \triangleleft_n x \} = m \} \\ \varepsilon[(\mathbf{pp-constraint} \ r \ n_1 \ n_2)] &= \{ x \in \mathcal{D} \mid \\ \forall y_1, y_2 \in \mathcal{D}: (y_1 \triangleleft_{n_1} x \wedge y_2 \triangleleft_{n_2} x) \rightarrow \langle y_1, y_2 \rangle \in \varepsilon[r] \} \end{aligned}$$

Subsumption (**is-a**) is defined as usual. $C_2 \Rightarrow C_1$ iff $\varepsilon[C_2] \subseteq \varepsilon[C_1]$ for every interpretation $\langle \mathcal{D}, \varepsilon \rangle$. We obtain the semantics that if $B \Rightarrow A$, then B may have additional kinds of parts, or more specialized parts than A , and the constraints between the parts of B may be stronger than those between the parts of A . If A has m_1 n -parts, B has m_2 n -parts and $m_1 \neq m_2$, then there is no subsumption relationship between A and B .

The language is targeted to the case where we know the number of parts for each occurring part name in a concept definition. To obtain this case we require that each concept definition has a **parts** construct for each occurring part name.

2.2 Tbox: Part-of

We use the following notation and definitions. If (after normalization [9]) (**part** 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 (written $A' \triangleleft_n B$) iff $(\exists A : (A \text{ is a direct } n\text{-part of } B) \wedge (A' \Rightarrow A))$. A is a *part* of B (written $A \triangleleft B$) iff $\exists n : A \triangleleft_n B$.

In the example above we have that *child* is a direct *offspring*-part of *standard-family*. If we know that *young-child* $\Rightarrow$ *child* then we also know that *young-child* is a *offspring*-part of *standard-family*.

If for a part name n , (**parts** m n) occurs in the definition of A , then we write that $N(n,A) = m$, otherwise $N(n,A) = 0$. $N(n,A)$ represents the number of n -parts which occur in the definition of A . The total number of defined parts for a concept A , denoted $N(A)$, is $\sum_n N(n,A)$. If (**pp-constraint** r n_1 n_2) occurs in the definition of A , then we write $A_{n_1 r n_2}$.

Another basic notion is the notion of *module*. (For the formal definition see [10].) In order for A to be a module for B (written $A \triangleleft_{\text{mod}} B$) it is required that (i) the number of defined parts in A is strictly less than the number of defined parts in B ; (ii) that all part names defined for A are also defined for B , with the number of each such defined part being at least as many for B as for A ; (iii) that the domains for part names in A are included in the domains of those part names for B ; and finally (iv) that all constraints defined between part names are at least as strong for A as for B . A module is essentially a collection of parts, such that none are redundant and the appropriate pp-constraints required by the composition are fulfilled.

Let *couple* be defined as *couple* $\doteq$ (**and** (**part** *husband* *man*) (**parts** *1* *husband*) (**part** *wife* *woman*) (**parts** *1* *wife*) (**pp-constraint** *married* *husband* *wife*)), then *couple* is a module of *standard-family*.

2.3 Abox

The language in which we can state information about individuals is defined as follows.

statement ::=
 (**concept-filler** *individual* *concept*)
 | (**role-fillers** *individual* *role* *individual*)
 | (**part-fillers** *individual* *part-name* *individual*)
individual ::= *identifier*

The terms *concept*, *part-name* and *role* are defined as for the *Tbox* syntax. If (**concept-filler** x C) appears in an *Abox*, then this means that $x \in \varepsilon[C]$. Similarly, (**role-fillers** x r y) means that $\langle x, y \rangle \in \varepsilon[r]$ and (**part-fillers** y n x) means that $y \triangleleft_n x$.

An *Abox* is a finite set of statements in the above language such that there are no cycles of parts. We will assume that within one *Abox* an individual has a unique name.

3 Learning Task

The learning task is defined as follows:

Given:

- *Tbox* language to describe concepts
- background knowledge described in *Tbox* language $\cup$ *Abox* language
- example set described in *Tbox* language $\cup$ *Abox* language. We allow the user to present several kinds of information to the learning system. Assume that the

concept to learn is C^* . Then the system handles the following cases (positive and negative examples):

membership and non-membership of individuals:

- $x \in \varepsilon[C^*]$, i.e. a particular individual belongs to the extension of the concept to learn
- $x \notin \varepsilon[C^*]$, i.e. a particular individual does not belong to the extension of the concept to learn

and a set of constraints that C^* should fulfill:

- $C^* \Rightarrow C$, i.e. a particular concept subsumes the concept to learn
- $C \Rightarrow C^*$, i.e. a particular concept is subsumed by the concept to learn
- $C \triangleleft_n C^*$, i.e. a particular concept is an n -part of the concept to learn
- $C \triangleleft_{\text{mod}} C^*$, i.e. a particular concept is a module of the concept to learn

Find:

- a concept description for C^* that satisfies the conditions in the examples

4 Useful Operations

In the learning algorithms we need generalization and specialization operations. Given two concepts C_1 and C_2 we can generalize to obtain the *least common subsumer* of the two concepts. We give the definition and a computation strategy in this section. As the language contains the **and**-construct, finding a more specific concept for the pair C_1 and C_2 can be defined as (**and** C_1 C_2).

Another useful operation is to associate with an individual a concept. This concept should reflect the properties of the individual as close as possible. We give a possible way of defining this notion.

4.1 LCS

The *least common subsumer* (LCS) of a pair of concepts is the most specific description in the infinite space of possible descriptions that subsumes the pair [2]. As our language includes the **and**-construct this LCS is unique.

We extend the computation of the LCS in [2] to also cope with the constructs related to part-of in figure 1. Figure 2 defines a *standard-family-with-boys* as a *standard-family* where the two *offsprings* are *boys*. A *family-with-2-girls* is also similar to a *standard-family* but the two *offsprings* are *girls* and we do not have the constraint that the *wife* and the *husband* are *married*. The LCS of the two concepts gives us then a family where the *offspring* belong to the concept *child* and we do not have the *married*-constraint. During the computation we make use of the fact that $\text{LCS}(\text{boy}, \text{girl}) = \text{child}$.

$\text{LCS}((\text{and } C_{11} \dots C_{1k}), (\text{and } C_{21} \dots C_{2l})) := (\text{and } C_{1121} \dots C_{112l} \dots C_{1k21} \dots C_{1k2l})$
 with $C_{1i2j} = \text{LCS}(C_{1i}, C_{2j})$
 $\text{LCS}((\text{all } r_1 \ C_1), (\text{all } r_2 \ C_2)) := \text{if } r_1=r_2 \text{ then } (\text{all } r_1 \ \text{LCS}(C_1, C_2)) \text{ else } \top$
 $\text{LCS}((\text{atmost } m_1 \ r_1), (\text{atmost } m_2 \ r_2)) := \text{if } r_1=r_2 \text{ then } (\text{atmost } \max(m_1, m_2) \ r_1) \text{ else } \top$
 $\text{LCS}((\text{atleast } m_1 \ r_1), (\text{atleast } m_2 \ r_2)) := \text{if } r_1=r_2 \text{ then } (\text{atleast } \min(m_1, m_2) \ r_1) \text{ else } \top$
 $\text{LCS}((\text{part } n_1 \ C_1), (\text{part } n_2 \ C_2)) := \text{if } n_1=n_2 \text{ then } (\text{part } n_1 \ \text{LCS}(C_1, C_2)) \text{ else } \top$
 $\text{LCS}((\text{parts } m_1 \ n_1), (\text{parts } m_2 \ n_2)) := \text{if } (n_1=n_2 \text{ and } m_1=m_2) \text{ then } (\text{parts } m_1 \ n_1) \text{ else } \top$
 $\text{LCS}((\text{pp-constraint } r_1 \ n_{11} \ n_{12}), (\text{pp-constraint } r_2 \ n_{21} \ n_{22})) :=$
 $\text{if } (n_{11}=n_{21} \text{ and } n_{12}=n_{22} \text{ and } r_1=r_2) \text{ then } (\text{pp-constraint } r_1 \ n_{11} \ n_{12}) \text{ else } \top$
 For primitive concepts C_1 and C_2 : $\text{LCS}(C_1, C_2) := \text{if } C_1=C_2 \text{ then } C_1 \text{ else } \top$
 other cases (with different constructors): $\text{LCS}(C_1, C_2) := \top$

Fig. 1. Least Common Subsumer

$\text{boy} \doteq (\text{and } \text{male child})$
 $\text{girl} \doteq (\text{and } \text{female child})$
 $\text{standard-family-with-boys} \doteq$
 $(\text{and } (\text{part husband man}) (\text{parts 1 husband}) (\text{part wife woman}) (\text{parts 1 wife})$
 $(\text{part offspring boy}) (\text{parts 2 offspring}) (\text{pp-constraint married husband wife})$
 $(\text{pp-constraint mother wife offspring}) (\text{pp-constraint father husband offspring}))$
 $\text{family-with-2-girls} \doteq (\text{and } (\text{part husband man}) (\text{parts 1 husband}) (\text{part wife woman})$
 $(\text{parts 1 wife}) (\text{part offspring girl}) (\text{parts 2 offspring}) (\text{pp-constraint mother wife offspring})$
 $(\text{pp-constraint father husband offspring}))$
 $\text{LCS}(\text{standard-family-with-boys}, \text{family-with-2-girls}) = (\text{family-with-2-children} \doteq)$
 $(\text{and } (\text{part husband man}) (\text{parts 1 husband}) (\text{part wife woman}) (\text{parts 1 wife})$
 $(\text{part offspring child}) (\text{parts 2 offspring}) (\text{pp-constraint mother wife offspring})$
 $(\text{pp-constraint father husband offspring}))$

Fig. 2. LCS of two kinds of family concepts

4.2 Specific Concepts

We repeat the definition from [5]. A concept C is a *specific concept* for an individual x with respect to a knowledge base $\langle Tbox, Abox \rangle$ iff (i) $x \in \varepsilon[C]$ and (ii) $\forall D: x \in \varepsilon[D] \rightarrow C \Rightarrow D$ in $Tbox$.

Depending on extra language constraints there are several possibilities for computing specific concepts. In general it is not possible to completely fit all the relevant information about an individual into a single concept in the language [11]. As a solution to this problem in [5] extra constructs are defined which are only to be used internally by the system but not by the user. Here we associate with an individual x a concept $\text{SC}(x)$ which can be defined in our language. We give some properties of this concept in figure 3. In the example in figure 4 we assume that we have complete knowledge of the world and that all roles and part names are closed.

concept-filler

$$\frac{kb \vdash x \rightarrow C}{SC(x) \Rightarrow C}$$

role-filler

$$\frac{\frac{kb \vdash \langle x, y \rangle \in \varepsilon[r]}{\exists C : (y \rightarrow C \wedge SC(x) \Rightarrow (\mathbf{all} \ r \ C))} \quad \frac{kb \vdash \#\{y \mid \langle x, y \rangle \in \varepsilon[r]\} = m}{SC(x) \Rightarrow (\mathbf{atleast} \ m \ r)}}{In \ the \ case \ that \ the \ role \ r \ is \ closed: \quad \frac{kb \vdash \#\{y \mid \langle x, y \rangle \in \varepsilon[r]\} = m}{SC(x) \Rightarrow (\mathbf{atmost} \ m \ r)}}$$

role-filler involving the parts

$$\frac{kb \vdash (\forall y_1 \triangleleft_{n_1} x, y_2 \triangleleft_{n_2} x : \langle y_1, y_2 \rangle \in \varepsilon[r])}{SC(x) \Rightarrow (\mathbf{pp-constraint} \ r \ n_1 \ n_2)}$$

part-filler

$$\frac{\frac{kb \vdash y \triangleleft_n x}{\exists C : (y \rightarrow C \wedge SC(x) \Rightarrow (\mathbf{allp} \ n \ C))} \quad \frac{kb \vdash \#\{y \mid y \triangleleft_n x\} = m}{N(SC(x), n) \geq m}}{In \ the \ case \ that \ the \ part \ name \ n \ is \ closed: \quad \frac{kb \vdash \#\{y \mid y \triangleleft_n x\} = m}{SC(x) \Rightarrow (\mathbf{parts} \ m \ n)}}$$

Fig. 3. Specific Concepts.

(concept-filler <i>f</i> (all accounts large))	(concept-filler <i>John</i> man)
(concept-filler <i>Mary</i> woman)	(concept-filler <i>Marc</i> child)
(concept-filler <i>Jane</i> child)	(part-fillers <i>John</i> husband <i>f</i>)
(part-fillers <i>Mary</i> wife <i>f</i>)	(part-fillers <i>Marc</i> offspring <i>f</i>)
(part-fillers <i>Jane</i> offspring <i>f</i>)	(role-fillers <i>John</i> married <i>Mary</i>)
(role-fillers <i>John</i> father <i>Marc</i>)	(role-fillers <i>John</i> father <i>Jane</i>)
(role-fillers <i>Mary</i> mother <i>Marc</i>)	(role-fillers <i>Mary</i> mother <i>Jane</i>)

$SC(Marc) = child$
 $SC(Jane) = child$
 $SC(Mary) = (\mathbf{and} \ woman \ (\mathbf{atleast} \ 2 \ mother) \ (\mathbf{atmost} \ 2 \ mother) \ (\mathbf{all} \ mother \ child))$
 $SC(John) = (\mathbf{and} \ man \ (\mathbf{atleast} \ 1 \ married) \ (\mathbf{atmost} \ 1 \ married) \ (\mathbf{all} \ married \ SC(Mary)))$
 $(\mathbf{atleast} \ 2 \ father) \ (\mathbf{atmost} \ 2 \ father) \ (\mathbf{all} \ father \ child)$
 $SC(f) = (\mathbf{and} \ (\mathbf{all} \ accounts \ large) \ (\mathbf{part} \ husband \ SC(John)) \ (\mathbf{parts} \ 1 \ husband) \ (\mathbf{part} \ wife \ SC(Mary)) \ (\mathbf{parts} \ 1 \ wife) \ (\mathbf{part} \ offspring \ child) \ (\mathbf{parts} \ 2 \ offspring) \ (\mathbf{pp-constraint} \ married \ husband \ wife) \ (\mathbf{pp-constraint} \ mother \ wife \ offspring) \ (\mathbf{pp-constraint} \ father \ husband \ offspring))$

Fig. 4. Specific concept: example

5 Learning Concepts

The algorithms we propose are targeted to learn descriptions of composite concepts. We propose to maintain two version spaces[8]: one for the is-a relation (or subsumption) and one for the part-of relation (i.e. parts and modules). The first version space is represented¹ by a set $\mathcal{G}$ of concepts that are more general than the concept to learn C^* , a set $\mathcal{S}$ of concepts that are more specific than

¹ As the **and** of two more general concepts is also a more general concept, and the LCS of two more specific concepts is also a more specific concept, $\mathcal{G}$ and $\mathcal{S}$ actually are singletons or the empty set.

C^* and a set $\mathcal{N}$ of individuals which do not belong to the extension of C^* . The second version space is represented by a set $\mathcal{C}$ of concepts for which C^* can be used as a building block and a set $\mathcal{B}$ of concepts which C^* can use as building blocks. Finally, we also keep information about the possibility or necessity of occurrence of part names. $\mathcal{P}^-$ denotes the part names which cannot occur in the definition of C^* . $\mathcal{P}_n$ is a set of tuples $\langle n, min, max, C \rangle$ for the part names which necessarily occur in the definition of C^* where n is the part name, min and max specify the interval to which the number of n -parts in C^* belongs, and C is the domain of n for C^* . $\mathcal{P}_p$ is a similar set for part names which possibly occur in C^* . Finally, we also keep information about which constraints between the parts are possible.

In this section we discuss the learning algorithms informally and point out the extra information we obtain by a special handling of part-of². In the examples involving subsumption and individuals we obtain also extra information about part-of. Similarly, the examples involving part-of give information for the is-a hierarchy. Below we assume that the concept to learn C^* is *standard-family*.

5.1 Learning Using Concepts and Subsumption

In the case where we know that $C^* \Rightarrow C$ we have immediately a more general concept and the standard updates can be performed. However, we also know the following: (i) the part names which occur in the definition of C also occur in the definition of C^* , (ii) the part names which occur in the definition of C occur in C^* with the same number, and (iii) the domain for the part name in C subsumes the domain for the corresponding part name in C^* . For instance, if we know that *standard-family* $\Rightarrow$ *family-with-2-children* (figure 2), then *husband*, *wife* and *offspring* must occur in the definition of *standard-family* with $N(husband, C^*) = 1$, $N(wife, C^*) = 1$ and $N(offspring, C^*) = 2$. We also know that the *husband*-part for *standard-family* is a *man*, the *wife*-part for *standard-family* is a *woman*, and each *offspring*-part for *standard-family* is a *child* (or more specific).

In the case where $C \Rightarrow C^*$ we have immediately a more specific concept. We use LCS in the updates. We also know that: (i) the part names which occur in the definition of C are the only possible part names for the definition of C^* , (ii) if some part name occurs in both definitions then the numbers are the same, and (iii) if some part name occurs in both definitions then the domain for the part name in C^* subsumes the domain for the corresponding part name in C . Knowing that *standard-family-with-boys* $\Rightarrow$ *standard-family* would tell us that the only part names occurring in the definition of *standard-family* can be *husband*, *wife* and *offspring*, although not all have to occur. However for the part names which do occur we also know the exact number. Similarly we know that for the occurring part names that the domains are more general than the ones in *standard-family-with-boys*. For instance, an *offspring* in *standard-family* is more general than a *boy*.

² With the standard updates below we mean the basic updates of the $\mathcal{G}$, $\mathcal{S}$, $\mathcal{C}$ and $\mathcal{B}$ sets. The updates in the data structures originating from the extra information are straightforward and we do not discuss the actual updates.

5.2 Learning Using Individuals

In the case where $x \in \varepsilon[C^*]$ we know that a specific concept covering x is more specific than C^* . Therefore we can use $SC(x)$ as a learning example. Furthermore, we also know that (i) the definition of C^* can only contain the part names for which there is a part in x . Having f as in figure 4 as an example for *standard-family* tells us that $SC(f) \Rightarrow \text{standard-family}$.

The examples where $x \notin \varepsilon[C^*]$ are used to find inconsistencies or concepts for which there is no definition in the language.

5.3 Learning Using Concepts and Part-of

In the case where $C \triangleleft_n C^*$ we have a building block for C^* and we have that: (i) if **(part** n A) occurs in the definition of C^* then $C \Rightarrow A$, and (ii) $N(n, C^*) \geq 1$. Knowing that *young-child* is an *offspring*-part of *standard-family*, gives us that **(part** *offspring* A) occurs in the definition of *standard-family* and *young-child* $\Rightarrow A$.

In the case where $C \triangleleft_{\text{mod}} C^*$ we have a building block for C^* and we have that: (i) if **(part** n A) occurs in C then **(part** n A') occurs in C^* such that $A \Rightarrow A'$, (ii) for the part names occurring both in C and C^* no other **pp-constraint**-terms can occur in the definition of C^* than the ones which occur in the definition of C , (iii) $N(C^*) > N(C)$, and (iv) $N(n, C^*) \geq N(n, C)$. The fact that *couple* $\triangleleft_{\text{mod}}$ *standard-family* implies that a *standard-family* has at least 1 *husband*-part which is at least as general than a *man* and at least 1 *wife*-part which is at least as general than a *woman*. We also know that the only constraint between *husband* and *wife* can be that they are *married*.

6 Related Work

In [3, 4] concepts are learned in CLASSIC [1]. The language is more expressive than the standard part of the language we use, but there are no constructs to deal with part-of. 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 C-CLASSIC³ is PAC-learnable. The algorithm is based on the LCS version in [2]. Learning from individuals is done by generalizing the individuals into concepts. A number of experiments have been performed.

KLUSTER [7] 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 BACK [9]. 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.

³ C-CLASSIC is a restriction of CLASSIC and has the constructors **and**, **all**, **atleast**, **atmost**, **fills**, **one-of**, **min** and **max**.

7 Future Work

In the case of role fillers and part fillers in the computation of $SC(x)$ for an individual x there is still a certain degree of freedom to find the concept the role or part filler belongs to. A trivial possibility is to assign to this concept the value $\top$. Another possibility is to use the SC of this other individual in the computation and be aware of the possibility of cycles (as we did in figure 4). We will investigate other ways and experiment.

The description logic we use as framework does not allow disjunction. However, [3] shows how disjunctive concepts may be learned in CLASSIC. As far as our algorithms are similar this approach seems to be mappable to our algorithm.

Acknowledgements

The authors would like to thank Nada Lavrač for useful comments on a previous draft. The first author is supported by grant 95-176 from the Swedish Research Council for Engineering Sciences (TFR).

References

1. Borgida, A., Brachman, R.J., McGuinness, D.L., Resnick, 'CLASSIC : A Structural Data Model for Objects', *Proceedings of SIGMOD 89*, pp 59-67, 1989.
2. Cohen, W., Borgida, A., Hirsh, H., 'Computing Least Common Subsumers in Description Logics', *Proceedings of AAAI 92*, pp 754-760, 1992.
3. Cohen, W., Hirsh, H., '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.
4. Cohen, W., Hirsh, H., 'The Learnability of Description Logics with Equality Constraints', *Machine Learning*, Vol 17, pp 169-199, 1994.
5. Donini, F., Era, A., 'Most Specific Concepts Technique for Knowledge Bases with Incomplete Information', *Proceedings of the First International Conference on Information and Knowledge Management*, pp 545-551, 1992.
6. Franconi, E., 'A Treatment of Plurals and Plural Qualifications based on a Theory of Collections', *Minds and Machines*, Vol 3(4), pp 453-474, 1993.
7. Kietz, J.-U., Morik, K., 'A Polynomial Approach to the Constructive Induction of Structural Knowledge', *Machine Learning*, Vol 14, pp 193-217, 1994.
8. Mitchell, T., 'Generalization as Search', *Artificial Intelligence*, Vol 18(2), pp 203-226, 1982.
9. Nebel, B., *Reasoning and Revision in Hybrid Representation Systems*, Lecture Notes in Artificial Intelligence, 422, Springer-Verlag, 1990.
10. Padgham, L., Lambrix, P., '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.
11. Schaerf, A., 'Which Semantics for Individuals in the Tbox?', *Proceedings of the International Workshop on Description Logics*, pp 5-8, 1994.

12. Speel, P.-H., Patel-Schneider, P., ‘CLASSIC extended with physical whole-part relations’, *Proceedings of the International Workshop on Description Logics*, pp 45-50, 1994.