Computing the Least Common Subsumer w.r.t. a Background Terminology[*]

Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan
Theoretical Computer Science, TU Dresden, Germany
{baader,sertkaya,turhan}@tcs.inf.tu-dresden.de

Abstract

Methods for computing the least common subsumer (lcs) are usually restricted to rather inexpressive DLs whereas existing knowledge bases are written in very expressive DLs. In order to allow the user to re-use concepts defined in such terminologies and still support the definition of new concepts by computing the lcs, we extend the notion of the lcs of concept descriptions to the notion of the lcs w.r.t. a background terminology.

1 Introduction and problem definition

Non-standard inferences such as computing the least common subsumer can be used to support the *bottom-up* construction of DL knowledge bases, as introduced in [4, 5]: instead of directly defining a new concept, the knowledge engineer introduces several typical examples as objects, which are then automatically generalized into a concept description by the system. This description is offered to the knowledge engineer as a possible candidate for a definition of the concept. The task of computing such a concept description can be split into two subtasks: computing the most specific concepts of the given objects, and then computing the least common subsumer of these concepts. The *most specific concept* (msc) of an object o (the *least common subsumer* (lcs) of concept descriptions $C_1, \dots, C_n$) is the most specific concept description C expressible in the given DL language that has o as an instance (that subsumes $C_1, \dots, C_n$). The problem of computing the lcs and (to a more limited extent) the msc has already been investigated in the literature [11, 12, 4, 5, 21, 20, 19, 3, 9].

The methods for computing the least common subsumer are restricted to rather inexpressive descriptions logics not allowing for disjunction (and thus not allowing for full negation). In fact, for languages with disjunction, the lcs of a collection of concepts is just their disjunction, and nothing new can be learned from building it. In contrast, for languages without disjunction, the lcs extracts the "commonalities" of the given collection of concepts. Modern DL systems like FaCT [18] and Racer [17] are based on very expressive DLs, and there exist large knowledge bases that use this

[*]This work has been supported by the German Research Foundation (DFG) under grants GRK 334/3 and BA 1122/4-3 and National ICT Australia Limited, Canberra Research Lab.

expressive power and can be processed by these systems [22, 23, 16]. In order to allow the user to re-use concepts defined in such existing knowledge bases and still support the user during the definition of new concepts with the bottom-up approach sketched above, we propose the following extended bottom-up approach.

Consider a *background terminology* $\mathcal{T}$ defined in an expressive DL $\mathcal{L}_2$. When defining new concepts, the user employs only a sublanguage $\mathcal{L}_1$ of $\mathcal{L}_2$, for which computing the lcs makes sense. However, in addition to primitive concepts and roles, the concept descriptions written in the DL $\mathcal{L}_1$ may also contain names of concepts defined in $\mathcal{T}$. Let us call such concept descriptions $\mathcal{L}_1(\mathcal{T})$-concept descriptions.

Definition 1 *Given an $\mathcal{L}_2$-TBox $\mathcal{T}$ and a collection $C_1, \ldots, C_n$ of $\mathcal{L}_1(\mathcal{T})$-concept descriptions, the* least common subsumer *(lcs) of $C_1, \ldots, C_n$* w.r.t. *$\mathcal{T}$ is the most specific $\mathcal{L}_1(\mathcal{T})$-concept description C that subsumes $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$, i.e., it is an $\mathcal{L}_1(\mathcal{T})$-concept description D such that*

1. *$C_i \sqsubseteq_{\mathcal{T}} D$ for $i = 1, \ldots, n$;* D is a common subsumer.
2. *if E is an $\mathcal{L}_1(\mathcal{T})$-concept description satisfying $C_i \sqsubseteq_{\mathcal{T}} E$ for $i = 1, \ldots, n$, then $D \sqsubseteq_{\mathcal{T}} E$.* D is least.

Depending on the DLs $\mathcal{L}_1$ and $\mathcal{L}_2$, least common subsumers of $\mathcal{L}_1(\mathcal{T})$-concept descriptions w.r.t. an $\mathcal{L}_2$-TBox $\mathcal{T}$ may exist or not.

Note that the lcs only uses concept constructors from $\mathcal{L}_1$, but may also contain concept names defined in the $\mathcal{L}_2$-TBox. This is the main distinguishing feature of this new notion of a least common subsumer w.r.t. a background terminology. Let us illustrate this by a small example.

Example 2 Assume that $\mathcal{L}_1$ is the DL $\mathcal{EL}$ (allowing for conjunction, existential restrictions, and the top concept) and $\mathcal{L}_2$ is $\mathcal{ALC}$ (extending $\mathcal{EL}$ by negation, disjunction, and value restrictions). Consider the $\mathcal{ALC}$-TBox

$$\mathcal{T} := \{A \equiv P \sqcup Q\},$$

and assume that we want to compute the lcs of the $\mathcal{EL}(\mathcal{T})$-concept descriptions P and Q. Obviously, A is the lcs of P and Q w.r.t. $\mathcal{T}$. If we were not allowed to use the name A defined in $\mathcal{T}$, then the only common subsumer of P and Q in $\mathcal{EL}$ would be the top concept $\top$.

In the following we always assume that DLs $\mathcal{L}_1$ and $\mathcal{L}_2$ and an $\mathcal{L}_2$-TBox are given, and if we talk about (least) common subsumers we mean the ones in $\mathcal{L}_1(\mathcal{T})$, and not in $\mathcal{L}_1$ or $\mathcal{L}_2$. In the next section, we consider the case where $\mathcal{L}_1$ is $\mathcal{EL}$ and $\mathcal{L}_2$ is $\mathcal{ALC}$ in more detail. We show the following two results:

- If $\mathcal{T}$ is an acyclic $\mathcal{ALC}$-TBox, then the lcs w.r.t. $\mathcal{T}$ of $\mathcal{EL}(\mathcal{T})$-concept descriptions always exists;
- If $\mathcal{T}$ is a general $\mathcal{ALC}$-TBox allowing for general concept inclusion axioms (GCIs), then the lcs w.r.t. $\mathcal{T}$ of $\mathcal{EL}(\mathcal{T})$-concept descriptions need not exist.

At first sight, one might assume that the first result can be shown using results on approximation of DLs [10]. In fact, given an acyclic $\mathcal{ALC}$-TBox $\mathcal{T}$ and $\mathcal{EL}(\mathcal{T})$-concept descriptions $C_1, \ldots, C_n$, one can first unfold $C_1, \ldots, C_n$ into $\mathcal{ALC}$-concept descriptions $C'_1, \ldots, C'_n$, then build the $\mathcal{ALC}$-concept description $C := C'_1 \sqcup \ldots \sqcup C'_n$, and finally approximate C from above by an $\mathcal{EL}$-concept description E. However, E then does not contain concept names defined in $\mathcal{T}$, and thus it is not necessarily the least $\mathcal{EL}(\mathcal{T})$-concept description subsuming $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ (see Example 2 above). One might now assume that this can be overcome by applying known results on rewriting concept descriptions w.r.t. a terminology [6]. However, in Example 2, the concept description E obtained using the approach based on approximation sketched above is $\top$, and this concept cannot be rewritten using the TBox $\mathcal{T} := \{A \equiv P \sqcup Q\}$.

The result on the existence and computability of the lcs w.r.t. a background terminology shown in the next section is theoretical in the sense that it does not yield a practical algorithm. In Section 3 we follow a more practical approach. Assume that $\mathcal{L}_1$ is a DL for which least common subsumers (without background TBox) always exist. Given $\mathcal{L}_1(\mathcal{T})$-concept descriptions $C_1, \ldots, C_n$, one can compute a common subsumer w.r.t. $\mathcal{T}$ by just ignoring $\mathcal{T}$, i.e., by treating the defined names in $C_1, \ldots, C_n$ as primitive and computing the lcs of $C_1, \ldots, C_n$ in $\mathcal{L}_1$. In Section 3 we sketch practical methods for computing "good" common subsumers w.r.t. background TBoxes, which may not be the *least* common subsumers, but which are better than the common subsumers computed by just ignoring the TBox.

2 Two exact theoretical results

In this section, we assume that $\mathcal{L}_1$ is $\mathcal{EL}$ and $\mathcal{L}_2$ is $\mathcal{ALC}$. In addition, we assume that the sets of concept and role names available for building concept descriptions are finite. First, we consider the case of acyclic TBoxes.

Theorem 3 *Let $\mathcal{T}$ be an acyclic $\mathcal{ALC}$-TBox. The lcs of $\mathcal{EL}(\mathcal{T})$-concept descriptions w.r.t. $\mathcal{T}$ always exists and can effectively be computed.*

The theorem is an easy consequence of the following facts:

1. If D is an $\mathcal{EL}(\mathcal{T})$-concept description of role depth k, then there are (not necessarily distinct) roles $r_1, \ldots, r_k$ such that $D \sqsubseteq \exists r_1.\exists r_2.\ldots\exists r_k.\top$

2. Let C be an $\mathcal{EL}(\mathcal{T})$-concept description, and assume that the $\mathcal{ALC}$-concept description C' obtained by unfolding C w.r.t. $\mathcal{T}$ is satisfiable and has the role depth $\ell < k$. Then $C' \not\sqsubseteq \exists r_1.\exists r_2.\ldots\exists r_k.\top$, and thus $C \not\sqsubseteq_{\mathcal{T}} \exists r_1.\exists r_2.\ldots\exists r_k.\top$. In fact, the standard tableau-based algorithm for $\mathcal{ALC}$ applied to C' constructs a tree-shaped interpretation of depth at most ℓ whose root individual belongs to C', but not to $\exists r_1.\exists r_2.\ldots\exists r_k.\top$.

3. For a given bound k on the role depth, there is only a finite number of inequivalent $\mathcal{EL}$-concept descriptions of role depth at most k. This is a consequence of the fact that we have assumed that the sets of concept and role names are finite, and can be shown by induction on k.

To show that these facts imply Theorem 3 consider the $\mathcal{EL}(\mathcal{T})$-concept descriptions $C_1, \dots, C_n$. If all of them are unsatisfiable w.r.t. $\mathcal{T}$, then one of them (e.g., C_1) can be taken as their lcs w.r.t. $\mathcal{T}$. Otherwise, assume that C_i is satisfiable w.r.t. $\mathcal{T}$. Let C_i' be the $\mathcal{ALC}$-concept description obtained by unfolding C_i w.r.t. $\mathcal{T}$, and assume that its role depth is ℓ. Now, take an arbitrary $\mathcal{EL}(\mathcal{T})$-concept description E that is a common subsumer of $C_1, \dots, C_n$ w.r.t. $\mathcal{T}$. Then, the role depth of E is at most ℓ. Otherwise, $C_i \sqsubseteq_\mathcal{T} E$ would be in contradiction to the above facts 1. and 2. Thus, fact 3. implies that, up to equivalence, there are only finitely many common subsumers of $C_1, \dots, C_n$ in $\mathcal{EL}(\mathcal{T})$. The least common subsumer is simply the conjunction of these finitely many $\mathcal{EL}(\mathcal{T})$-concept descriptions.

It is not hard to see that the above proof is effective in the sense that one can effectively compute (representatives of the equivalence classes of) all common subsumers of $C_1, \dots, C_n$, and then build their conjunction. However, this brute-force algorithm is probably not useful in practice.

Second, we consider the case of TBoxes allowing for GCIs.

Theorem 4 *Let $\mathcal{T} := \{A \sqsubseteq \exists r.A,\ B \sqsubseteq \exists r.B\}$. Then, the lcs of the $\mathcal{EL}(\mathcal{T})$-concept descriptions A, B w.r.t. $\mathcal{T}$ does not exist.*

Proof. Let E_n denote the $\mathcal{EL}$-concept description $\exists r.\exists r.\dots\exists r.\top$ of role depth n. For all $n \geq 0$, E_n is a common subsumer of A and B w.r.t. $\mathcal{T}$. Assume that D is a least common subsumer of A and B, and let ℓ be the role depth of D. If D contains neither A nor B, then $D \not\sqsubseteq_\mathcal{T} E_n$ for all $n > \ell$, which is a contradiction. However, if D contains A, then it is easy to see that D cannot be a subsumer of B, and if D contains B, then it cannot be a subsumer of A. Consequently, such a least common subsumer D cannot exist. □

Note that this example is very similar to the one showing non-existence of the lcs in $\mathcal{EL}$ with cyclic terminologies interpreted with descriptive semantics [2]. However, the proof of the result in [2] is more complicated since there one is allowed to extend the terminology in order to build the lcs.

3 A practical approximative approach

We have seen above that the lcs w.r.t. general background TBoxes need not exist. In addition, even in the case of acyclic TBoxes, where the lcs always exists, we do not have a practical algorithm for computing the lcs. In the bottom-up construction of DL knowledge bases, it is not really necessary to use the *least* common subsumer,[1] a common subsumer that is not too general can also be used. In this section, we introduce an approach for computing such "good" common subsumers w.r.t. a background TBox. In order to explain this approach, we must first recall how the lcs of $\mathcal{EL}$-concept descriptions can be computed.

[1]Using it may even result in over-fitting.

The lcs of $\mathcal{EL}$-concept descriptions

Since the lcs of n concept descriptions can be obtained by iterating the application of the binary lcs, we describe how to compute the lcs $\mathsf{lcs}_{\mathcal{EL}}(C, D)$ of two $\mathcal{EL}$-concept descriptions C, D.

In order to describe this algorithm, we need to introduce some notation. Let C be an $\mathcal{EL}$-concept description. Then $\mathsf{names}(C)$ denotes the set of concept names occurring in the top-level conjunction of C, $\mathsf{roles}(C)$ the set of role names occurring in an existential restriction on the top-level of C, and $\mathsf{restrict}_r(C)$ denotes the set of all concept descriptions occurring in an existential restriction on the role r on the top-level of C.

Now, let C, D be $\mathcal{EL}$-concept descriptions. Then we have

$$\begin{aligned}\mathsf{lcs}_{\mathcal{EL}}(C, D) \quad = \quad & \bigsqcap_{A\in\mathsf{names}(C)\cap\mathsf{names}(D)} A \ \sqcap \\ & \bigsqcap_{r\in\mathsf{roles}(C)\cap\mathsf{roles}(D)} \ \bigsqcap_{E\in\mathsf{restrict}_r(C), F\in\mathsf{restrict}_r(D)} \exists r.\mathsf{lcs}_{\mathcal{EL}}(E, F)\end{aligned}$$

Here, the empty conjunction stands for the top concept $\top$. The recursive call of $\mathsf{lcs}_{\mathcal{EL}}$ is well-founded since the role depth of the concept descriptions in $\mathsf{restrict}_r(C)$ ($\mathsf{restrict}_r(D)$) is strictly smaller than the role depth of C (D).

A good common subsumer in $\mathcal{EL}$ w.r.t. a background TBox

Let $\mathcal{T}$ be a background TBox (acyclic or general) in some DL $\mathcal{L}_2$ extending $\mathcal{EL}$ such that subsumption in $\mathcal{L}_2$ w.r.t. this class of TBoxes is decidable. Let C, D be $\mathcal{EL}(\mathcal{T})$-concept descriptions. If we ignore the TBox, then we can simply apply the above algorithm for $\mathcal{EL}$-concept descriptions to compute a common subsumer. However, in this context taking

$$\bigsqcap_{A\in\mathsf{names}(C)\cap\mathsf{names}(D)} A$$

is not the best we can do. In fact, some of these concept names may be constrained by the TBox, and thus there may be relationships between them that we ignore by simply using the intersection.

Instead, we propose to take the smallest (w.r.t. subsumption w.r.t. $\mathcal{T}$) conjunction of concept names that subsumes (w.r.t. $\mathcal{T}$) both

$$\bigsqcap_{A\in\mathsf{names}(C)} A \qquad \text{and} \qquad \bigsqcap_{B\in\mathsf{names}(D)} B.$$

We modify the above lcs algorithm in this way, not only on the top level of the input concepts, but also in the recursive steps. It is easy to show that the $\mathcal{EL}(\mathcal{T})$-concept description computed by this modified algorithm still is a common subsumer of A, B w.r.t. $\mathcal{T}$. In general, this common subsumer will be more specific than the one obtained by ignoring $\mathcal{T}$, though it need not be the least common subsumer.

As a simple example, consider the $\mathcal{ALC}$-TBox $\mathcal{T}$:

$$\begin{array}{rcl}
\mathsf{NoSon} & \equiv & \forall\mathsf{has\text{-}child.Female}, \\
\mathsf{NoDaughter} & \equiv & \forall\mathsf{has\text{-}child.}\neg\mathsf{Female}, \\
\mathsf{SonRichDoctor} & \equiv & \forall\mathsf{has\text{-}child.(Female} \sqcup (\mathsf{Doctor} \sqcap \mathsf{Rich})) \\
\mathsf{DaughterHappyDoctor} & \equiv & \forall\mathsf{has\text{-}child.}(\neg\mathsf{Female} \sqcup (\mathsf{Doctor} \sqcap \mathsf{Happy})) \\
\mathsf{ChildrenDoctor} & \equiv & \forall\mathsf{has\text{-}child.Doctor}
\end{array}$$

and the $\mathcal{EL}$-concept descriptions

$$\begin{array}{rcl}
C & := & \exists\mathsf{has\text{-}child.(NoSon} \sqcap \mathsf{DaughterHappyDoctor}), \\
D & := & \exists\mathsf{has\text{-}child.(NoDaughter} \sqcap \mathsf{SonRichDoctor}).
\end{array}$$

If we ignore the TBox, then we obtain the $\mathcal{EL}$-concept description $\exists\mathsf{has\text{-}child.}\top$ as common subsumer of C, D. However, if we take into account that both $\mathsf{NoSon} \sqcap \mathsf{DaughterHappyDoctor}$ and $\mathsf{NoDaughter} \sqcap \mathsf{SonRichDoctor}$ are subsumed by the concept $\mathsf{ChildrenDoctor}$, then we obtain the more specific common subsumer

$$\exists\mathsf{has\text{-}child.ChildrenDoctor}.$$

Computing the subsumption lattice of conjunctions of concept names

In order to obtain a practical lcs algorithm realizing the approach described above, we must be able to compute in an efficient way the smallest conjunction of concept names that subsumes two such conjunctions w.r.t. $\mathcal{T}$. We propose to precompute this information using methods from formal concept analysis (FCA) [15]. In FCA, the knowledge about an application domain is given by means of a formal context.

Definition 5 *A* formal context *is a triple $\mathcal{K} = (\mathcal{O}, \mathcal{P}, \mathcal{S})$, where $\mathcal{O}$ is a set of objects, $\mathcal{P}$ is a set of attributes (or properties), and $\mathcal{S} \subseteq \mathcal{O} \times \mathcal{P}$ is a relation that connects each object o with the attributes satisfied by o.*

Let $\mathcal{K} = (\mathcal{O}, \mathcal{P}, \mathcal{S})$ be a formal context. For a set of objects $A \subseteq \mathcal{O}$, A' is the set of attributes that are satisfied by all objects in A, i.e.,

$$A' := \{p \in \mathcal{P} \mid \forall a \in A\colon (a, p) \in \mathcal{S}\}.$$

Similarly, for a set of attributes $B \subseteq \mathcal{P}$, B' is the set of objects that satisfy all attributes in B, i.e.,

$$B' := \{o \in \mathcal{O} \mid \forall b \in B\colon (o, b) \in \mathcal{S}\}.$$

A *formal concept* is a pair (A, B) consisting of an *extent* $A \subseteq \mathcal{O}$ and an *intent* $B \subseteq \mathcal{P}$ such that $A' = B$ and $B' = A$. Such formal concepts can be hierarchically ordered by inclusion of their extents, and this order (denoted by $\leq$ in the following) induces a complete lattice, called the *concept lattice* of the context. Given a formal context, the first step for analyzing this context is usually to compute the concept lattice.

In many applications, one has a large (or even infinite) set of objects, but only a relatively small set of attributes. Also, the context is not necessarily given explicitly as a cross table; it is rather "known" to a domain "expert". In such a situation, Ganter's *attribute exploration* algorithm [13, 15] has turned out to be an efficient approach for computing an appropriate representation of the concept lattice. This algorithm is interactive in the sense that at certain stages it asks the "expert" certain questions about the context, and then continues using the answers provided by the expert. Once the representation of the concept lattice is computed, certain questions about the lattice (e.g. "What is the supremum of two given concepts?") can efficiently be answered using this representation.

Recall that we are interested in the subsumption lattice[2] of conjunctions of concept names (some of which may occur in GCIs or concept definitions of an $\mathcal{L}_2$-TBox $\mathcal{T}$). In order to apply attribute exploration to this task, we define a formal context whose concept lattice is isomorphic to the subsumption lattice we are interested in. This problem was first addressed in [1], where the objects of the context were basically all possible counterexamples to subsumption relationships, i.e., interpretations together with an element of the interpretation domain. The resulting "semantic context" has the disadvantage that an "expert" for this context must be able to deliver such counterexample, i.e., it is not sufficient to have a simple subsumption algorithm for the DL in question. One needs one that, given a subsumption problem "$C \sqsubseteq D$?", is able to compute a counterexample if the subsumption relationship does not hold, i.e., an interpretation $\mathcal{I}$ and an element d of its domain such that $d \in C^{\mathcal{I}} \setminus D^{\mathcal{I}}$.

To overcome this problem, a new "syntactic context" was recently defined in [8]:

Definition 6 *The context $\mathcal{K}_{\mathcal{T}} = (\mathcal{O}, \mathcal{P}, \mathcal{S})$ is defined as follows:*

$$\begin{aligned}
\mathcal{O} &:= \{E \mid E \textit{ is an } \mathcal{L}_2 \textit{ concept description}\};\\
\mathcal{P} &:= \{A_1, \ldots, A_n\} \textit{ is the set of concept names occurring in } \mathcal{T},\\
\mathcal{S} &:= \{(E, A) \mid E \sqsubseteq_{\mathcal{T}} A\}.
\end{aligned}$$

The following is shown in [8]:

Theorem 7 *(1) The concept lattice of the context $\mathcal{K}_{\mathcal{T}}$ is isomorphic to the subsumption hierarchy of all conjunctions of subsets of $\mathcal{P}$ w.r.t. $\mathcal{T}$.*
(2) Any decision procedure for subsumption w.r.t. TBoxes in $\mathcal{L}_2$ functions as an expert for the context $\mathcal{K}_{\mathcal{T}}$.

It should be noted that formal concept analysis and attribute exploration has already been applied in a different context to the problem of computing the least common subsumer. In [7], the following problem is addressed: given a finite collection $\mathcal{C}$ of concept descriptions, compute the subsumption hierarchy of all least common subsumers of subsets of $\mathcal{C}$. Again, this extended subsumption hierarchy can be computed by defining a formal context whose concept lattice is isomorphic to the subsumption

[2]In general, the subsumption relation induces a partial order, and not a lattice structure on concepts. However, in the case of conjunctions of concept names, all infima exist, and thus also all suprema.

lattice we are interested in, and then applying attribute exploration (see [7] for details). In [8], it is shown that this approach and the one sketched above can be seen as two instances of a more abstract approach.

Extension to DLs more expressive than $\mathcal{EL}$

For the DL $\mathcal{ALE}$ (which extends $\mathcal{EL}$ by value restrictions and atomic negation), an lcs algorithm similar to the one described for $\mathcal{EL}$ exists [5]. The main differences are that (i) the concept descriptions must first be normalized (which may lead to an exponential blow-up); (ii) the recursive calls also deal with value restrictions, and not just existential restrictions; and (iii) on the top level, one has to deal with a conjunction of concept names and *negated* concept names. In the lcs algorithm, the conjunctions mentioned in (iii) are treated similarly to the case of $\mathcal{EL}$ (unless they are contradictory): one separately computes the intersections of the positive and of the negative concept names.

When adapting this algorithm to one that computes "good" common subsumers in $\mathcal{ALE}$ w.r.t. a background TBox, all we have to change is to compute a conjunction of concept names and negated concept names that is the most specific such conjunction subsuming the given conjunctions w.r.t. the TBox, rather than building the intersections. It is easy to see that attribute exploration can again be used to precompute the necessary information. Basically, the only change is that now both concept names and negated concept names are attributes in the formal context.

4 Future work

The attributes of the formal contexts introduced in our approach (concept names and possibly negated concept names) are not independent of each other. For example, the name A and its negation $\neg A$ are disjoint, i.e., it is not possible for an object (other than $\perp$) of the context to satisfy both A and $\neg A$. In addition, the TBox induces subsumption relationships between the attributes (and this information may already be precomputed for the given TBox during classification). Thus, one can try to apply a modified version of attribute exploration that can use such background knowledge [14] to speed up the exploration process.

References

[1] Franz Baader. Computing a minimal representation of the subsumption lattice of all conjunctions of concepts defined in a terminology. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, pages 168–178, 1995.

[2] Franz Baader. Computing the least common subsumer in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 11th International Conference on Conceptual Structures, ICCS 2003*, volume 2746 of *Lecture Notes in Artificial Intelligence*, pages 117–130. Springer-Verlag, 2003.

[3] Franz Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 319–324. Morgan Kaufmann, 2003.

[4] Franz Baader and Ralf Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of the 22th German Annual Conf. on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1998.

[5] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, 1999.

[6] Franz Baader, Ralf Küsters, and Ralf Molitor. Rewriting concepts using terminologies. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-00)*, pages 297–308, 2000.

[7] Franz Baader and Ralf Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In B. Ganter and G. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues – Proceedings of the 8th International Conference on Conceptual Structures (ICCS2000)*, volume 1867 of *Lecture Notes In Artificial Intelligence*, pages 290–303. Springer-Verlag, 2000.

[8] Franz Baader and Baris Sertkaya. Applying formal concept analysis to description logics. In P. Eklund, editor, *Proceedings of the 2nd International Conference on Formal Concept Analysis (ICFCA 2004)*, volume 2961 of *Lecture Notes in Computer Science*, pages 261–286, Sydney, Australia, 2004. Springer-Verlag.

[9] S. Brandt, A.-Y. Turhan, and R. Küsters. Extensions of non-standard inferences for description logics with transitive roles. In M. Vardi and A. Voronkov, editors, *Proceedings of the tenth International Conference on Logic for Programming and Automated Reasoning (LPAR'03)*, LNCS. Springer, 2003.

[10] Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 203–214, San Francisco, CA, 2002. Morgan Kaufman.

[11] William W. Cohen and Haym Hirsh. Learning the CLASSIC description logics: Theoretical and experimental results. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 121–133, 1994.

[12] Michael Frazier and Leonard Pitt. CLASSIC learning. *Machine Learning*, 25:151–193, 1996.

[13] Bernhard Ganter. Finding all closed sets: A general approach. *Order*, 8:283–290, 1991.

[14] Bernhard Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999.

[15] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, 1999.

[16] Volker Haarslev and Ralf Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-01)*, 2001.

[17] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-01)*, 2001.

[18] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 636–647, 1998.

[19] Ralf Küsters and Alex Borgida. What's in an attribute? Consequences for the least common subsumer. *Journal of Artificial Intelligence Research*, 14:167–203, 2001.

[20] Ralf Küsters and Ralf Molitor. Approximating most specific concepts in description logics with existential restrictions. In Franz Baader, Gerd Brewka, and Thomas Eiter, editors, *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes In Artificial Intelligence*, pages 33–47, Vienna, Austria, 2001. Springer-Verlag.

[21] Ralf Küsters and Ralf Molitor. Computing least common subsumers in $\mathcal{ALEN}$. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-01)*, pages 219–224, 2001.

[22] Alan Rector and Ian Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, 1997. AAAI Press.

[23] Stefan Schultz and Udo Hahn. Knowledge engineering by large-scale knowledge reuse—experience from the medical domain. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-00)*, pages 601–610. Morgan Kaufmann, 2000.