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

Abstract. Methods for computing the least common subsumer (lcs) are usually restricted to rather inexpressive Description Logics (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. We will both show a theoretical result on the existence of the *least* common subsumer in this setting, and describe a practical approach (based on a method from formal concept analysis) for computing *good* common subsumers, which may, however, not be the least ones.

1 Introduction

Description Logics (DLs) [3] are a class of knowledge representation formalisms in the tradition of semantic networks and frames, which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way. DL systems provide their users with standard inference services (like subsumption and instance checking) that deduce implicit knowledge from the explicitly represented knowledge. More recently, non-standard inferences [21] were introduced to support building and maintaining large DL knowledge bases. For example, such non-standard inferences 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 $C_1, \ldots, C_n$) is the most specific concept description C expressible in the given DL language that has o as an instance (that subsumes $C_1, \ldots, C_n$). The problem of computing the lcs and (to a more limited extent) the msc has already been investigated in the literature [12, 13, 4, 5, 24, 23, 22, 2, 11].

* This work has been supported by DFG under grants GRK 334/3 and BA 1122/4-3.

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 [20] and RACER [19] are based on very expressive DLs, and there exist large knowledge bases that use this expressive power and can be processed by these systems [25, 26, 18]. 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. Given $\mathcal{L}_1(\mathcal{T})$-concept descriptions $C_1, \ldots, C_n$, we are now looking for their lcs in $\mathcal{L}_1(\mathcal{T})$, i.e., the least $\mathcal{L}_1(\mathcal{T})$-concept description that subsumes $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$.

In this paper, we consider the case where $\mathcal{L}_1$ is the DL $\mathcal{ALE}$ and $\mathcal{L}_2$ is the DL $\mathcal{ALC}$. We first show the following result: If $\mathcal{T}$ is an acyclic $\mathcal{ALC}$-TBox, then the lcs w.r.t. $\mathcal{T}$ of $\mathcal{ALE}(\mathcal{T})$-concept descriptions always exists. This result (which will be shown in Section 3) is theoretical in the sense that it does not yield a practical algorithm.

In Section 4 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 $\mathcal{L}_1$. However, the common subsumer obtained this way will usually be too general. In Section 4 we sketch a practical method 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 ignoring the TBox. As a tool, this method uses attribute exploration with background knowledge [15, 16], an algorithm developed in formal concept analysis [17] for computing concept lattices.

2 Basic definitions

In order to define concepts in a DL knowledge base, one starts with a set N_C of concept names (unary predicates) and a set N_R of role names (binary predicates), and defines more complex *concept descriptions* using the constructors provided by the concept description language of the particular system. In this paper, we consider the DL $\mathcal{ALC}$ and its sublanguages $\mathcal{ALE}$ and $\mathcal{EL}$, which allow for concept descriptions built from the indicated subsets of the constructors shown

Table 1. Syntax and semantics of concept descriptions and definitions.

Name of constructor	Syntax	Semantics	$\mathcal{ALC}$	$\mathcal{ALE}$	$\mathcal{EL}$
top-concept	$\top$	$\Delta^\mathcal{I}$	x	x	x
bottom-concept	$\bot$	$\emptyset$	x	x	
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$	x		
atomic negation	$\neg A$	$\Delta^\mathcal{I} \setminus A^\mathcal{I}$	x	x	
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$	x	x	x
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$	x		
value restriction	$\forall r.C$	$\{x \in \Delta^\mathcal{I} \mid \forall y : (x,y) \in r^\mathcal{I} \rightarrow y \in C^\mathcal{I}\}$	x	x	
existential restriction	$\exists r.C$	$\{x \in \Delta^\mathcal{I} \mid \exists y : (x,y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$	x	x	x
concept definition	$A \equiv C$	$A^\mathcal{I} = C^\mathcal{I}$	x	x	x

in Table 1. In this table, r stands for a role name, A for a concept name, and C, D for arbitrary concept descriptions. A *concept definition* (as shown in the last row of Table 1) assigns a concept name A to a complex description C. A finite set of such definitions is called a *TBox* iff it is acyclic (i.e., no definition refers, directly or indirectly, to the name it defines) and unambiguous (i.e., each name has at most one definition). The concept names occurring on the left-hand side of a concept definition are called *defined* concepts, and the others *primitive*.

The semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$. The domain $\Delta^\mathcal{I}$ of $\mathcal{I}$ is a non-empty set and the interpretation function $\cdot^\mathcal{I}$ maps each concept name $A \in N_C$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and each role name $r \in N_R$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The extension of $\cdot^\mathcal{I}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1. The interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies all its concept definitions, i.e., $A^\mathcal{I} = C^\mathcal{I}$ holds for all $A \equiv C$ in $\mathcal{T}$.

One of the most important traditional inference services provided by DL systems is computing subconcept/superconcept relationships (so-called *subsumption* relationships). The concept description C_2 *subsumes* the concept description C_1 *w.r.t. the TBox* $\mathcal{T}$ ($C_1 \sqsubseteq_\mathcal{T} C_2$) iff $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$ for all models $\mathcal{I}$ of $\mathcal{T}$. Two concept descriptions C_1, C_2 are called *equivalent* iff they subsume each other w.r.t. the empty TBox.

We are now ready to define the new non-standard inference introduced in this paper. Let $\mathcal{L}_1, \mathcal{L}_2$ be DLs such that $\mathcal{L}_1$ is a sub-DL of $\mathcal{L}_2$, i.e., $\mathcal{L}_1$ allows for less constructors. For a given $\mathcal{L}_2$-TBox $\mathcal{T}$, we call $\mathcal{L}_1(\mathcal{T})$*-concept descriptions* those $\mathcal{L}_1$-concept descriptions that may contain concepts defined in $\mathcal{T}$.

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 trivial example.

Example 1. Assume that $\mathcal{L}_1$ is the DL $\mathcal{ALE}$ and $\mathcal{L}_2$ is $\mathcal{ALC}$. Consider the $\mathcal{ALC}$-TBox $\mathcal{T} := \{A \equiv P \sqcup Q\}$, and assume that we want to compute the lcs of the $\mathcal{ALE}(\mathcal{T})$-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 $\mathcal{ALE}$ would be the top-concept $\top$.

3 An exact theoretical result

In this section, we assume that $\mathcal{L}_1$ is $\mathcal{ALE}$ 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.

Theorem 1. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox. The lcs of $\mathcal{ALE}(\mathcal{T})$-concept descriptions w.r.t. $\mathcal{T}$ always exists and can effectively be computed.*

At first sight, one might think that this result can be shown using results on the approximation of $\mathcal{ALC}$ by $\mathcal{ALE}$ [10]. In fact, given an $\mathcal{ALC}$-TBox $\mathcal{T}$ and $\mathcal{ALE}(\mathcal{T})$-concept descriptions $C_1, \ldots, C_n$, one can first *unfold* $C_1, \ldots, C_n$ into concept descriptions $C'_1, \ldots, C'_n$ by iteratively replacing defined concepts by their definitions until they contain no defined concepts. These descriptions are $\mathcal{ALC}$-concept descriptions since they may contain constructors of $\mathcal{ALC}$ that are not allowed in $\mathcal{ALE}$. One can then build the $\mathcal{ALC}$-concept description $C := C'_1 \sqcup \ldots \sqcup C'_n$, and finally approximate C from above by an $\mathcal{ALE}$-concept description E. By construction, E is a common subsumer of $C_1, \ldots, C_n$. However, E does not contain concept names defined in $\mathcal{T}$, and thus it is not necessarily the *least* $\mathcal{ALE}(\mathcal{T})$-concept descriptions subsuming $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ (see Example 1 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 1, the concept description E obtained using the approach based on approximation sketched above is $\top$, and this concept cannot be rewritten using the TBox $\mathcal{T} := \{A \equiv P \sqcup Q\}$.

To show the theorem, we first need to show two lemmas. Given an $\mathcal{ALC}$- or $\mathcal{ALE}(\mathcal{T})$-concept description C, its role depth is the maximal nesting of value restrictions and existential restrictions. For example, the role depth of $\exists r.\forall r.A$ is 2, and the role depth of $\exists r.\forall r.A \sqcup \exists r.\exists r.\exists r.B$ is 3.

Lemma 1. *For a given bound k on the role depth, there is only a finite number of inequivalent $\mathcal{ALE}$-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.[1]

Given this lemma, a first attempt to show Theorem 1 could be the following. Let $C_1, \ldots, C_n$ be $\mathcal{ALE}(\mathcal{T})$-concept descriptions, and assume that the role depths of the $\mathcal{ALC}$-concept description $C'_1, \ldots, C'_n$ obtained by unfolding the C_i w.r.t. $\mathcal{T}$ are bounded by k. If we could show that this implies that the role depth of any common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ is also bounded by k, then we could obtain the least common subsumer by simply building the (up to equivalence) finite conjunction of all common subsumers of $C_1, \ldots, C_n$ in $\mathcal{ALE}(\mathcal{T})$. However, due to the fact that $\mathcal{ALC}$ and $\mathcal{ALE}$ can express inconsistency, this simple approach does not work. In fact, $\perp$ has role depth 0, but is subsumed by any concept description. Given this counterexample, the next conjecture could be that it is enough to prevent this pathological case, i.e., assume that at least one of the concept descriptions $C_1, \ldots, C_n$ is consistent, i.e., not subsumed by $\perp$ w.r.t. $\mathcal{T}$. For the DL $\mathcal{EL}$ in place of $\mathcal{ALE}$, this modification of the simple approach sketched above really works (see [9] for details). However, due to the presence of value restrictions it does not work for $\mathcal{ALE}$. For example, $\forall r.\perp$ is subsumed by $\forall r.F$ for arbitrary $\mathcal{ALE}(\mathcal{T})$-concept descriptions F, and thus the role depth of common subsumers cannot be bounded. However, we can show that common subsumers having a large role depth are too general anyway.

Lemma 2. *Let $C_1, \ldots, C_n$ be $\mathcal{ALE}(\mathcal{T})$-concept descriptions, and assume that the role depths of the $\mathcal{ALC}$-concept description $C'_1, \ldots, C'_n$ obtained by unfolding the C_i w.r.t. $\mathcal{T}$ are bounded by k. If the $\mathcal{ALE}(\mathcal{T})$-concept description D is a common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$, then there is an $\mathcal{ALE}(\mathcal{T})$-concept description $D' \sqsubseteq_{\mathcal{T}} D$ of role depth at most $k+1$ that is also a common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$.*

Theorem 1 is now an immediate consequence of Lemma 1 and 2. In fact, to compute the lcs of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$, it is enough to compute the (up to equivalence) finite set of all $\mathcal{ALE}(\mathcal{T})$-concept descriptions of role depth $k+1$, check which of them are common subsumers of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$, and then build the conjunction E of these common subsumers. Lemma 1 ensures that the conjunction is finite. By definition, E is a common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$, and Lemma 2 ensures that for any common subsumer D of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$, there is a conjunct D' in E such that $D' \sqsubseteq_{\mathcal{T}} D$, and thus $E \sqsubseteq_{\mathcal{T}} D$.

Due to the space constraints, we can only *sketch* the *proof of Lemma 2.* Assume that D is an $\mathcal{ALE}(\mathcal{T})$-concept description of role depth $> k+1$ that is a common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$. Then there are quantifiers $Q_1, \ldots, Q_k$, $Q \in \{\forall, \exists\}$, roles $r_1, \ldots, r_k, r$, and an $\mathcal{ALE}(\mathcal{T})$-concept description F containing a value or existential restriction such that $D \sqsubseteq_{\mathcal{T}} Q_1 r_1. \cdots Q_k r_k.Qr.F$.

Case 1: $C_i \sqsubseteq_{\mathcal{T}} Q_1 r_1. \cdots Q_k r_k.\perp$ for all $i, 1 \leq i \leq n$. Then $D \sqcap Q_1 r_1. \cdots Q_k r_k.\perp$ is a common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ that is subsumed by D, and can

[1] This is a well-known result, which holds even for full first-order predicate logic formulae of bounded quantifier depth over a finite vocabulary.

be normalized into an equivalent concept description that is smaller than D (basically, $Qr.F$ can be replaced by $\bot$). Thus, we can apply induction to obtain the result of the lemma.

Case 2: There is an $m, 1 \leq m \leq n$ such that $C_m \not\sqsubseteq_{\mathcal{T}} Q_1 r_1. \cdots Q_k r_k.\bot$. Using the fact that C'_m has role depth at most k, we can show[2] that this implies $C_m \not\sqsubseteq_{\mathcal{T}} Q_1 r_1. \cdots Q_k r_k.\exists r.\top$. Thus, $C_m \sqsubseteq_{\mathcal{T}} D \sqsubseteq_{\mathcal{T}} Q_1 r_1. \cdots Q_k r_k.Qr.F$ shows that $Q = \forall$.

If F subsumes $\top$ w.r.t. $\mathcal{T}$, then we can replace $\forall r.F$ in D by $\top$, and thus obtain an equivalent smaller description. Otherwise, we can use the fact that $C_i \sqsubseteq_{\mathcal{T}} Q_1 r_1. \cdots Q_k r_k.\forall r.F$ and that the role depth of C'_i is at most k to show that $C_i \sqsubseteq_{\mathcal{T}} Q_1 r_1. \cdots Q_k r_k.\forall r.\bot$ holds for all $i, 1 \leq i \leq n$. But then $D \sqcap Q_1 r_1. \cdots Q_k r_k.\forall r.\bot$ is a common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ that is subsumed by D, and can be normalized into an equivalent concept description that is smaller than D. Again, we can apply induction to obtain the result of the lemma.

4 A practical approximative approach

The brute-force algorithm for computing the lcs in $\mathcal{ALE}(\mathcal{T})$ w.r.t. a background $\mathcal{ALC}$-TBox described in the previous section is not useful in practice since the number of concept descriptions that must be considered is very large (super-exponential in the role depth). In the bottom-up construction of DL knowledge bases, it is not really necessary to take the *least* common subsumer,[3] 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 $\mathcal{ALE}$-concept descriptions (without background terminology) can be computed.

The lcs of $\mathcal{ALE}$-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 $\mathsf{lcs}_{\mathcal{ALE}}(C, D)$ of two $\mathcal{ALE}$-concept descriptions C, D (see [5] for more details).

First, the input descriptions C, D are normalized by applying the following rules modulo associativity and commutativity of conjunction:

$$\begin{array}{rll} \forall r.E \sqcap \forall r.F \longrightarrow \forall r.(E \sqcap F), & & \forall r.E \sqcap \exists r.F \longrightarrow \forall r.E \sqcap \exists r.(E \sqcap F), \\ \forall r.\top \longrightarrow \top, & & E \sqcap \top \longrightarrow E, \\ A \sqcap \neg A \longrightarrow \bot & \text{for each } A \in N_C, & \\ \exists r.\bot \longrightarrow \bot, & & E \sqcap \bot \longrightarrow \bot. \end{array}$$

Due to the second rule, this normalization may lead to an exponential blow-up of the concept descriptions. In the following, we assume that the input descriptions C, D are normalized.

[2] By looking a the behavior of a tableau-based subsumption algorithm for $\mathcal{ALC}$.
[3] Using it may even result in over-fitting.

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

Now, let C, D be normalized $\mathcal{ALE}$-concept descriptions. If C (D) is equivalent to $\bot$, then $\mathsf{lcs}_{\mathcal{ALE}}(C,D) = D$ ($\mathsf{lcs}_{\mathcal{ALE}}(C,D) = C$). Otherwise, we have

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

Here, the empty conjunction stands for the top-concept $\top$. The recursive calls of $\mathsf{lcs}_{\mathcal{ALE}}$ are well-founded since the role depth decreases with each call.

A good common subsumer in $\mathcal{ALE}$ w.r.t. a background TBox Let $\mathcal{T}$ be a background TBox in some DL $\mathcal{L}_2$ extending $\mathcal{ALE}$ such that subsumption in $\mathcal{L}_2$ w.r.t. TBoxes is decidable.[4] Let C, D be normalized $\mathcal{ALE}(\mathcal{T})$-concept descriptions. If we ignore the TBox, then we can simply apply the above algorithm for $\mathcal{ALE}$-concept descriptions without background terminology to compute a common subsumer. However, in this context, taking

$$
\bigsqcap_{A\in\mathsf{names}(C)\cap\mathsf{names}(D)} A \ \sqcap \bigsqcap_{\neg B\in\overline{\mathsf{names}}(C)\cap\overline{\mathsf{names}}(D)} \neg B
$$

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 and negated concept names that subsumes (w.r.t. $\mathcal{T}$) both

$$
\bigsqcap_{A\in\mathsf{names}(C)} A \sqcap \bigsqcap_{\neg B\in\overline{\mathsf{names}}(C)} \neg B \quad \text{and} \quad \bigsqcap_{A'\in\mathsf{names}(D)} A' \sqcap \bigsqcap_{\neg B'\in\overline{\mathsf{names}}(D)} \neg B'.
$$

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 $\mathcal{ALE}(\mathcal{T})$-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

[4] Note that the restriction to TBoxes consisting of acyclic and unambiguous concept definitions is not really necessary here. We can also treat sets of general concept inclusions (GCIs) in this way.

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 $\mathcal{T}$:

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

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

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

By ignoring the TBox, we obtain the $\mathcal{ALE}(\mathcal{T})$-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

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

Computing the subsumption lattice of conjunctions of (negated) concept names w.r.t. a TBox 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 (negated) concept names that subsumes two such conjunctions w.r.t. $\mathcal{T}$. Since in our application scenario (bottom-up construction of DL knowledge bases w.r.t. a given background terminology), the TBox $\mathcal{T}$ is assumed to be fixed, it makes sense to precompute this information. Obviously, a naive approach that calls the subsumption algorithm for each pair of conjunctions of (negated) concept names is too expensive for TBoxes of a realistic size. Instead, we propose to use methods from formal concept analysis (FCA) [17] for this purpose. In FCA, the knowledge about an application domain is given by means of a formal context.

Definition 2. *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 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 [14, 17] 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[5] of conjunctions of (negated) concept names (some of which may be defined concepts in 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.

For the case of conjunctions of concept names (without negated names), 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 3. *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 2. *(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. $\mathcal{T}$.*

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

(2) Any decision procedure for subsumption w.r.t. TBoxes in $\mathcal{L}_2$ functions as an expert for the context $\mathcal{K}_{\mathcal{T}}$.

This result can easily be extended to the case of conjunctions of concept names *and negated* concept names. In fact, one can simply extend the TBox $\mathcal{T}$ by a definition for each negated concept name, and then apply the approach to this extended TBox. To be more precise, if $\{A_1, \ldots, A_n\}$ is the set of concept names occurring in $\mathcal{T}$, then we introduce new concept names $\overline{A}_1, \ldots, \overline{A}_n$, and extend $\mathcal{T}$ to a TBox $\widehat{\mathcal{T}}$ by adding the definitions $\overline{A}_1 \equiv \neg A_1, \ldots, \overline{A}_n \equiv \neg A_n$.[6]

Corollary 1. *The concept lattice of the context $\mathcal{K}_{\widehat{\mathcal{T}}}$ is isomorphic to the subsumption hierarchy of all conjunctions of concept names and negated concept names occurring in $\mathcal{T}$.*

The experimental results reported in [8] show that this approach for computing the subsumption lattice of all conjunctions of concept names gives a huge increase of efficiency compared to the semi-naive approach, which introduces a new definition for each (of the exponentially many) such conjunctions, and then applies the usual algorithm for computing the subsumption hierarchy. Nevertheless, these results also show that the approach can only be applied if the number of concept names is relatively small (less than 30).[7] For this reason, we propose to use an improved algorithm for computing concept lattices [15, 16], which can employ additional background knowledge that is readily available in our context, but not used by the basic attribute exploration algorithm.

An improved approach using attribute exploration with background knowledge When starting the exploration process, all the basic attribute exploration algorithm knows about the context is the set of its attributes. It acquires all the necessary knowledge about the context by asking the expert (which in our setting means: by calling the subsumption algorithm for $\mathcal{L}_2$). However, in our application we already have some knowledge about relationships between attributes:

1. Since $\mathcal{T}$ is assumed to be an existing terminology, we can usually assume that the subsumption hierarchy between the concept names occurring in $\mathcal{T}$ has already been computed. If $A_i \sqsubseteq_{\mathcal{T}} A_j$ holds, then we know on the FCA side that in the context $\mathcal{K}_{\widehat{\mathcal{T}}}$ all objects satisfying attribute A_i also satisfy attribute A_j.
2. Since $A_i \sqsubseteq_{\mathcal{T}} A_j$ implies $\neg A_j \sqsubseteq_{\mathcal{T}} \neg A_i$, we also know that all objects satisfying attribute $\overline{A}_j$ also satisfy attribute $\overline{A}_i$.

[6] For $\widehat{\mathcal{T}}$ to be an $\mathcal{L}_2$-TBox, we must assume that $\mathcal{L}_2$ allows for full negation.

[7] It should be noted, however, that these experiments were done almost 10 years ago on a rather slow computer, using randomly generated TBoxes and the semantic context.

3. Finally, we know that no object can simultaneously satisfy A_i and $\overline{A}_i$ and every object satisfies either A_i or $\overline{A}_i$.[8]

Attribute exploration with background knowledge [15, 16] is able to use such additional information on the context to speed up the exploration process and to obtain a smaller representation of the concept lattice.

Depending on the TBox, there may exist other such relationships between attributes that can be deduced, but it should be noted that deducing them makes sense only if this can be done without too much effort: otherwise, the efficiency gained during the exploration might be outweighed by the effort of obtaining the background knowledge.

First experimental results First experiments with prototypical implementations of attribute exploration (with and without background knowledge) and of a DL "expert" based on RACER [18] yield mixed results. First, the runtime of attribute exploration (both with and without background knowledge) strongly depends on the specific shape of the TBox, not just its size. On the one hand, the TBox used as an example in this section (which has 9 concept names, and thus leads to a context with 18 attributes, and 2^{18} different conjunctions of them) resulted in runtimes of almost 50 minutes, both with and without background knowledge. One reason for this bad behavior could be that there are almost no relationships between the concepts (with background knowledge, only one additional implication is generated). On the other hand, handcrafted TBoxes with more concepts, but also more relationships between them, could be handled within several seconds.

Second, while the use of background knowledge decreases the number of calls to the expert significantly, it does not decrease the overall runtime, and in some cases even increases it. The main reason for this unexpected behavior appears to be that the examples used until now are so small that an optimized implementation like RACER needs almost no time to answer subsumption questions. In addition, our implementation of the reasoner for the background knowledge (which is used during attribute exploration with background knowledge) is still unoptimized, and thus the overhead of using the background knowledge is large.

5 Related and future work

In a preliminary version of this paper [9], we have considered computing the lcs in $\mathcal{EL}$ w.r.t. a background $\mathcal{ALC}$-terminology. We have shown that the lcs w.r.t. acyclic TBoxes always exists in this setting, and have also sketched a practical approach for computing an approximation of the lcs. The present version of the paper improves on this by considering the considerably more expressive DL $\mathcal{ALE}$ in place of $\mathcal{EL}$ (which makes the proof of Theorem 1 much harder), by

[8] If we encode both facts in the background knowledge, then the background knowledge mentioned in point 2. is redundant. However, first tests indicate that it may nevertheless be advantageous to add it explicitly.

extending the approach for computing the subsumption lattice of all conjunctions of concept names to conjunctions of concept names and negated concept names, and by using attribute exploration with background knowledge.

It should be noted that formal concept analysis and attribute exploration have 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 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.

On the experimental side, the main topic for future research is, on the one hand, to compare the behavior of attribute exploration with background knowledge to the one without on more and larger knowledge bases. We will also evaluate the trade-off between the cost of extracting more background knowledge and the performance gain this additional knowledge yields during attribute exploration. On the other hand, we will analyze how good the "good" common subsumers computed by our approximative approach really are. On the theoretical side, we will try to find exact algorithms for computing the *least* common subsumer that are better than the brute-force algorithm sketched in the proof of Theorem 1.

References

1. F. 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, eds., *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, 1995.
2. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, eds., *Proc. of the 18th Int. Joint Conf. on AI.*, Morgan Kaufm., 2003.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
4. F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of the 22nd German Annual Conf. on AI. (KI'98),LNCS*, Springer, 1998.
5. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on AI. (IJCAI'99)*, 1999.
6. F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Repr. and Reasoning (KR'2000)*, 2000.
7. F. Baader and R. Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In B. Ganter and G.

Mineau, eds., *Conceptual Structures: Logical, Linguistic, and Computational Issues – Proc. of the 8th Int. Conf. on Conceptual Structures (ICCS2000)*, Springer, 2000.
8. F. Baader and B. Sertkaya. Applying formal concept analysis to description logics. In P. Eklund, ed., *Proc. of the 2nd Int. Conf. on Formal Concept Analysis (ICFCA 2004)*, *LNCS*, Sydney, Australia, 2004. Springer.
9. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. In *Proc. of the 2004 Int. Workshop on Description Logics (DL2004)*, 2004.
10. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, eds., *Proc. of the 8th Int. Conf. on Principles of Knowledge Repr. and Reasoning (KR2002)*, San Francisco, CA, 2002. Morgan Kaufm.
11. 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, eds., *Proc. of the 10th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'03)*, *LNAI*. Springer, 2003.
12. W. Cohen and H. Hirsh. Learning the CLASSIC description logics: Theoretical and experimental results. In J. Doyle, E. Sandewall, P. Torasso, eds., *Proc. of the 4th Int. Conf. on Principles of Knowledge Repr. and Reasoning (KR'94)*, 1994.
13. M. Frazier and L. Pitt. CLASSIC learning. *Machine Learning*, 25:151–193, 1996.
14. B. Ganter. Finding all closed sets: A general approach. *Order*, 8:283–290, 1991.
15. B. Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999.
16. B. Ganter and R. Krauße. Pseudo models and propositional Horn inference. Technical Report MATH-AL-15-1999, Inst. f. Algebra, TU Dresden, Germany, 1999.
17. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin, 1999.
18. V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on AI. (IJCAI 2001)*, 2001.
19. V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.
20. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Repr. and Reasoning (KR'98)*, 1998.
21. R. Küsters. *Non-standard Inferences in Description Logics*, *LNAI*. Springer, 2001.
22. R. Küsters and A. Borgida. What's in an attribute? Consequences for the least common subsumer. *J. of AI. Research*, 14:167–203, 2001.
23. R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. In F. Baader, G. Brewka, and T. Eiter, eds., *Proc. of the Joint German/Austrian Conf. on AI. (KI 2001)*, *LNAI*, 2001. Springer.
24. R. Küsters and R. Molitor. Computing least common subsumers in $\mathcal{ALEN}$. In *Proc. of the 17th Int. Joint Conf. on AI. (IJCAI 2001)*, 2001.
25. A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, 1997. AAAI Press.
26. S. Schultz and U. Hahn. Knowledge engineering by large-scale knowledge reuse—experience from the medical domain. In A. G. Cohn, F. Giunchiglia, and B. Selman, eds., *Proc. of the 7th Int. Conf. on Principles of Knowledge Repr. and Reasoning (KR'2000)*, Morgan Kaufm., 2000.