Approximation and Difference in Description Logics[*]

Sebastian Brandt
Theoretical Computer Science,
RWTH Aachen

Ralf Küsters
Theoretical Computer Science,
University of Kiel

Anni-Yasmin Turhan
Theoretical Computer Science,
RWTH Aachen

Abstract

Approximation is a new inference service in Description Logics first mentioned by Baader, Küsters, and Molitor. Approximating a concept, defined in one Description Logic, means to translate this concept to another concept, defined in a second typically less expressive Description Logic, such that both concepts are as closely related as possible with respect to subsumption. The present paper provides the first in-depth investigation of this inference task. We prove that approximations from the Description Logic $\mathcal{ALC}$ to $\mathcal{ALE}$ always exist and propose an algorithm computing them.

As a measure for the accuracy of the approximation, we introduce a syntax-oriented difference operator, which yields a concept that contains all aspects of the approximated concept that are not present in the approximation. It is also argued that a purely semantical difference operator, as introduced by Teege, is less suited for this purpose. Finally, for the logics under consideration, we propose an algorithm computing the difference.

1 Introduction

Approximation in Description Logics (DLs) was first mentioned by Baader, Küsters, and Molitor [2] as a possible new inference problem. The present paper is the first to investigate this problem in depth. Informally, approximation is defined as follows: given a concept C defined in a DL $\mathcal{L}_s$ ("s" for source) find a concept D, the upper/lower approximation of C, in a DL $\mathcal{L}_d$ ("d" for destination) such that i) D subsumes/is subsumed by C, and ii) D is a minimal/maximal concept in $\mathcal{L}_d$ (w.r.t. subsumption) with this property. Throughout this paper we will mainly focus on upper approximations. There are a number of different applications of this inference problem, from which we will briefly mention two here (see [7] for others, such as the translation of knowledge-bases and knowledge-base vivification, an application already mentioned in [5, 9]).

Translation of knowledge-bases
Approximation can be used to (automatically) translate a knowledge-base written in an expressive DL into a another (semantically closely related) knowledge-base in a less expressive DL. The translation may become necessary to port knowledge-bases between different knowledge representation systems or to integrate different knowledge-bases.

Non-standard inferences for expressive DLs. Non-standard inferences in DLs, such as computing the least common subsumer (lcs), matching and unification of concepts, have been introduced to support the construction and maintenance of DL knowledge-bases (see [12] for an overview). However, up to now they are mostly restricted to quite inexpressive DLs, for example to those that do not allow for concept disjunction. Approximation can be used to overcome this problem to some extent. For example, the existing matching algorithms can be lifted up to handle more expressive DLs as follows: instead of directly matching concept patterns (defined in a small DL) against concepts (defined in a DL that can not be handled by existing matching algorithms), one can first approximate the concept (in the small DL) and then match against its approximation. Even though some information may be lost, e.g., the matcher is more general than the cor-

[*] This work has been partially supported by the Deutsche Forschungsgemeinschaft, DFG Project BA 1122/4-1.

rect one, the accuracy of the result may still suffice.

Another example, which was in fact the main motivation for us to investigate approximation in the first place, is the *computation of commonalities between concepts*. This inference service is used in our chemical process engineering application [14] to support the bottom-up construction of knowledge-bases [1, 6]. Typically, the lcs is employed to accomplish this task. Formally, the lcs of two concepts, say C_1 and C_2, defined in some DL $\mathcal{L}$, is the most specific concept (w.r.t. subsumption) in $\mathcal{L}$ that subsumes both concepts. In case $\mathcal{L}$ allows for concept disjunction, the lcs is just the disjunction of C_1 and C_2 ($C_1 \sqcup C_2$). Thus, the problem is that a user inspecting this concept does not learn anything about the actual commonalities between C_1 and C_2. By using approximation, however, one can make the commonalities explicit by first approximating C_1 and C_2 in a sublanguage of $\mathcal{L}$ which does not allow to express concept disjunction, and then computing the lcs of the approximations in this sublanguage.

Supporting frame-based user interfaces of DL systems. In the interaction with DL systems, users with little knowledge representation expertise may have difficulties to understand and make use of the full expressive power of the underlying DLs. To overcome this problem, some knowledge representation systems have been equipped with a simplified frame-based user interface built on top of a more powerful DL system. Examples for such systems are the TAMBIS system [3] and the ontology editor OilEd [4] built on top of the FaCT DL system [11]. On many occasions, these systems have to present concept descriptions to the user for editing, inspection, or as a solution of inference problems. Such concept descriptions, however, need not always fit into the restricted representation of the frame-based user interface or might overwhelm an inexperienced user. In such cases, approximation might be helpful as a means to represent concept descriptions in a simplified fashion suited to the user interface and the users level of expertise.

The main technical result of this paper (Section 3) is to show that concept descriptions defined in the standard DL $\mathcal{ALC}$, which allows for concept conjunction and disjunction, value and existential restrictions, and full negation, can be approximated (from above) in the DL $\mathcal{ALE}$, a DL that does not allow for concept disjunction and full negation.

Once one has given an (upper) approximation D of C a natural question regards the loss of information, i.e., what aspects of C are not captured by D. Therefore we propose a difference operator, which given C (in $\mathcal{L}_s$) and D (in $\mathcal{L}_d$) yields a concept E (the difference of C and D) in $\mathcal{L}_s$ such that E conjoint with D is equivalent to C ($E \sqcap D \equiv C$). In other words, E contains the information that is missing in the approximation D of C. Such an operator has already been defined by Teege [16]. He requires that E is the most general concept description in $\mathcal{L}_s$ w.r.t. subsumption that satisfies the above equivalence. However, as we will see, such a purely semantical definition of difference yields very unintuitive concepts. We therefore propose a new syntax-based definition, which better captures the intuition behind difference. Roughly speaking, the difference E between C and D will be obtained by syntactically removing those parts of C that are already present in D. In Section 4, we provide a formal definition and give an algorithm for computing the difference between an $\mathcal{ALC}$- and an $\mathcal{ALE}$-concept description.

In Section 5 we present some experiences with our prototypical implementations of the algorithms presented here and conclude with some remarks on future work. All details and full proofs of our results can be found in our technical report [7].

2 Description Logics

Concept descriptions are inductively defined with the help of a set of concept *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. The available constructors determine the expressive power of the DL in question. In this paper, we consider concept descriptions built from the constructors shown in Table 1. In the DL $\mathcal{ALE}$, concept descriptions are formed using the constructors top-concept ($\top$), concept conjunction ($C \sqcap D$), existential restriction ($\exists r.C$), value restriction ($\forall r.C$), primitive negation ($\neg A$), and the bottom-concept ($\bot$). The DL $\mathcal{ALC}$ additionally provides us with concept disjunction ($C \sqcup D$) and full negation ($\neg C$). Note that in $\mathcal{ALC}$ every concept description can be negated whereas in $\mathcal{ALE}$ negation is only allowed in front of concept names. For a DL $\mathcal{L}$, such as $\mathcal{ALE}$ and $\mathcal{ALC}$, a concept description formed with the constructors allowed in $\mathcal{L}$ is called *$\mathcal{L}$-concept description* in the following.

As usual, the semantics of a concept description is defined in terms of an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$. The domain Δ 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 $A^{\mathcal{I}} \subseteq \Delta$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta \times \Delta$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is defined inductively, as shown in the second column of Table 1.

One of the most important traditional inference services provided by DL systems is computing the sub-

Syntax	Semantics	$\mathcal{ALE}$	$\mathcal{ALC}$
$\top$	Δ	x	x
$C \sqcap D$	$C^I \cap D^I$	x	x
$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$	x	x
$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$	x	x
$\neg A$, $A \in N_C$	$\Delta \setminus A^I$	x	x
$\bot$	$\emptyset$	x	x
$C \sqcup D$	$C^I \cup D^I$		x
$\neg C$	$\Delta \setminus C^I$		x

Table 1: Syntax and semantics of concept descriptions.

sumption hierarchy. The concept description C is *subsumed* by the description D ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ holds for all interpretations $\mathcal{I}$; C and D are *equivalent* ($C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$; C is strictly subsumed by D ($C \sqsubset D$) iff $C \sqsubseteq D$ and $C \not\equiv D$. Subsumption and equivalence in $\mathcal{ALC}$ are PSPACE-complete [15] and NP-complete in $\mathcal{ALE}$ [10].

In order to approximate $\mathcal{ALC}$-concept descriptions by $\mathcal{ALE}$-concept descriptions, we will need to compute the least common subsumer in $\mathcal{ALE}$.

Definition 1 *Given $\mathcal{L}$-concept descriptions $C_1, \ldots, C_n$,for some description logic $\mathcal{L}$, the $\mathcal{L}$-concept description C is the* least common subsumer *(lcs) of $C_1, \ldots, C_n$ ($C = \mathsf{lcs}(C_1, \ldots, C_n)$ for short) iff (i) $C_i \sqsubseteq C$ for all $1 \leq i \leq n$, and (ii) C is the least concept description with this property, i.e., if C' satisfies $C_i \sqsubseteq C'$ for all $1 \leq i \leq n$, then $C \sqsubseteq C'$.*

Depending on the DL under consideration, the lcs of two or more concept descriptions need not always exist, but if it exists, then, by definition, it is unique up to equivalence. For instance, in $\mathcal{ALC}$ the lcs trivially exists since $lcs(C, D) \equiv C \sqcup D$. For $\mathcal{ALE}$, which does not allow for concept disjunction, the existence is not obvious. However, as shown in [1], the lcs of two or more $\mathcal{ALE}$-concept descriptions always exists, its size may grow exponentially in the size of the input descriptions, and it can be computed in exponential time.

3 Computing Approximations

In this section, we show how $\mathcal{ALC}$-concept descriptions can be approximated (from above) by $\mathcal{ALE}$-concept descriptions. Let us first define the notion of approximation formally.

Definition 2 *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two DLs, and let C be an $\mathcal{L}_1$- and D be an $\mathcal{L}_2$-concept description. Then, D is called* upper (lower) $\mathcal{L}_2$-approximation *of C ($D = \mathsf{approx}_{\mathcal{L}_2}(C)$ for short) if (i) $C \sqsubseteq D$ ($D \sqsubseteq C$), and (ii) D is minimal (maximal) with this property, i.e., $C \sqsubseteq D'$ and $D' \sqsubseteq D$ ($D \sqsubseteq D'$) implies $D' \equiv D$ for all $\mathcal{L}_2$-concept descriptions D'.*

Note that approximations need not exist in general. Consider for example the DLs $\mathcal{L}_1 = \{\sqcap\}$ and $\mathcal{L}_2 = \{\sqcup\}$, i.e., the DLs that only allow for concept conjunction and concept disjunction, respectively. Let A and B denote concept names. Then, there does not exist an upper $\mathcal{L}_1$-approximation of the $\mathcal{L}_2$-concept description $A \sqcup B$. Conversely, there does not exist a lower $\mathcal{L}_2$-approximation of the $\mathcal{L}_1$-concept description $A \sqcap B$. Also note that approximations need not be uniquely determined. For example, both A and B are lower $\mathcal{L}_1$-approximations of $A \sqcup B$ with $\mathcal{L}_1$ defined as above.

In this paper, we restrict our investigations to upper approximations. Therefore, whenever we speak of approximations in the following, we mean upper approximations. Moreover, we concentrate on upper $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions. Since $\mathcal{ALE}$ allows for concept conjunction it immediately follows that if upper $\mathcal{ALE}$-approximations exist, they are uniquely determined up to equivalence: If D_1 and D_2 are two upper $\mathcal{ALE}$-approximations of the same $\mathcal{ALC}$-concept, then so is $D_1 \sqcap D_2$. But then, by definition of upper approximation, $D_1 \sqcap D_2 \sqsubseteq D_1$ and $D_1 \sqcap D_2 \sqsubseteq D_2$ implies $D_1 \sqcap D_2 \equiv D_1 \equiv D_2$.

3.1 The naïve Approximation Approach

Now, let us turn to the question of how upper $\mathcal{ALE}$-approximations can be computed from $\mathcal{ALC}$-concept descriptions. We first present a naïve approach to this problem and show that it fails. This will then motivate the definition of the (correct) approximation algorithm.

The naïve approach. It is easy to see that, given an $\mathcal{ALC}$-concept description $C = E \sqcup F$ with $\mathcal{ALE}$-concept descriptions E and F, the $\mathcal{ALE}$-approximation of C is

$\mathsf{lcs}(E, F)$. Having observed this, one might think that every $\mathcal{ALC}$-concept description C can be approximated by simply replacing every concept disjunction in C by the lcs operator and evaluating the lcs operators from inside out. However, the $\mathcal{ALC}$-concept description

$$C_{\mathsf{ex},1} = (\forall r.B \sqcup (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A,$$

with concept names A and B, illustrates that this is not the case: The obtained approximation would be $\mathsf{lcs}(\forall r.B, (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A \equiv \top \sqcap \exists r.A \equiv \exists r.A$. However, as one can easily check, $C_{\mathsf{ex},1} \sqsubseteq \exists r.(A \sqcap B) \sqsubset \exists r.A$. In fact, $\exists r.(A \sqcap B)$ is the correct upper $\mathcal{ALE}$-approximation of $C_{\mathsf{ex},1}$.

As it turns out, we will have to turn the concept descriptions into a certain normal form before substituting disjunctions by the lcs. Roughly speaking, the normal forms are obtained by distributing concept conjunctions over concept disjunctions. In the example, this yields the concept description $(\forall r.B \sqcap \exists r.A) \sqcup (\exists r.B \sqcap \forall r.A \sqcap \exists r.A)$ and replacing the disjunction by the lcs yields $\mathsf{lcs}(\forall r.B \sqcap \exists r.A, \exists r.B \sqcap \forall r.A \sqcap \exists r.A) \equiv \exists r.(A \sqcap B)$, which is the correct result.

Still, the following example illustrates that normalizing concepts in this way does not suffice in the general case. The description

$$C_{\mathsf{ex},2} = \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$$

is already in normal form, but substituting the concept disjunction with the lcs yields $\exists r.A \sqcap \exists r.B \sqcap \forall r.\mathsf{lcs}(\neg A, \neg B) \equiv \exists r.A \sqcap \exists r.B \sqcap \forall r.\top \equiv \exists r.A \sqcap \exists r.B$. However, the $\mathcal{ALE}$-approximation of $C_{\mathsf{ex},2}$ is $\exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)$. The reason is that we need to propagate value restrictions on existential restrictions in order to obtain the correct approximations of the existential restrictions.

In what follows, we will first introduce the normal forms and then present the approximation algorithm, which works on these normal forms and does the propagation on-the-fly.

3.2 $\mathcal{ALC}$-Normalform

For the sake of simplicity, we assume that the set N_R of role names is the singleton $\{r\}$. However, all definitions and results can easily be generalized to arbitrary sets of role names. We also assume that each conjunction in an $\mathcal{ALE}$-concept description contains at most one value restriction of the form $\forall r.C'$ (this is w.l.o.g. due to the equivalence $\forall r.E \sqcap \forall r.F \equiv \forall r.(E \sqcap F)$). Some notation is needed to access the different parts of an $\mathcal{ALE}$-concept description C (and an $\mathcal{ALC}$-concept description where disjunction only occurs within value or existential restrictions): $\mathsf{prim}(C)$ denotes the set of all (negated) concept names and the bottom concept occurring on the top-level conjunction of C; if there exists a value restriction of the form $\forall r.C'$ on the top-level conjunction of C, then $\mathsf{val}(C) := C'$; otherwise, $\mathsf{val}(C) := \top$; $\mathsf{ex}(C) := \{C' \mid$ there exists $\exists r.C'$ on the top-level conjunction of $C\}$.

Definition 3 *An $\mathcal{ALC}$-concept description C is in* $\mathcal{ALC}$-normal form *iff*

1. *if $C \equiv \bot$, then $C = \bot$; if $C \equiv \top$, then $C = \top$;*
2. *otherwise, C is of the form $C = C_1 \sqcup \cdots \sqcup C_n$ with*

$$C_i = \bigsqcap_{A \in \mathsf{prim}(C_i)} A \sqcap \bigsqcap_{C' \in \mathsf{ex}(C_i)} \exists r.C' \sqcap \forall r.\mathsf{val}(C_i),$$

$C_i \not\equiv \bot$, and $\mathsf{val}(C_i)$ and every concept description in $\mathsf{ex}(C_i)$ is in $\mathcal{ALC}$-normal form, for all $i = 1, \ldots, n$.

Obviously, every $\mathcal{ALC}$-concept description can be turned into an equivalent concept description in $\mathcal{ALC}$-normal form. Unfortunately, this may take exponential time, as the example $(A_1 \sqcup A_2) \sqcap \cdots \sqcap (A_{2n-1} \sqcup A_{2n})$ shows, whose $\mathcal{ALC}$-normal form is of size exponential in n.

3.3 Computing Approximations

Our approximation algorithm is based on the following structural characterization of subsumption between an $\mathcal{ALC}$-concept description, say C, in $\mathcal{ALC}$-normal form and an $\mathcal{ALE}$-concept description, say D. The idea is that D is compared to every disjunct C_i in C. This comparison in turn is very similar to the structural characterization of subsumption between $\mathcal{ALE}$-concept descriptions [1].

Theorem 4 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form (as specified in Definition 3) and D an $\mathcal{ALE}$-concept description. Then, $C \sqsubseteq D$ iff $C \equiv \bot$, or $D \equiv \top$, or for all $i = 1, \ldots, n$ it holds that*

1. *$\mathsf{prim}(D) \subseteq \mathsf{prim}(C_i)$, and*
2. *for all $D' \in \mathsf{ex}(D)$ there exists $C' \in \mathsf{ex}(C_i)$ such that $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$, and*
3. *$\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$.*

The approximation algorithm, denoted by $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$, is depicted in Figure 1. Given C, it finds an $\mathcal{ALE}$-concept description, which is as specific as possible and satisfies the conditions of

```
Input: $\mathcal{ALC}$-concept description $C$
Output: upper $\mathcal{ALE}$-approximation of $C$
  1. If $C \equiv \bot$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \bot$; if $C \equiv \top$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \top$
  2. Otherwise, transform $C$ into $\mathcal{ALC}$-normal form $C_1 \sqcup \cdots \sqcup C_n$ and return
     $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) :=$
         $\bigsqcap_{A \in \bigcap_{i=1}^{n} \mathsf{prim}(C_i)} A \sqcap$
         $\bigsqcap_{(C'_1,\ldots,C'_n) \in \mathsf{ex}(C_1) \times \cdots \times \mathsf{ex}(C_n)} \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\} \sqcap$
         $\forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$
```

Figure 1: The recursive algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$.

Theorem 10. For $C \equiv \bot$ and $C \equiv \top$ this is trivial. In case $C \not\equiv \bot$ and $D := \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \not\equiv \top$, one needs to show that i) the Conditions 1, 2, and 3 of Theorem 10 are satisfied for C and D, and that ii) D is a minimal concept description with this property. Here we only give an idea of how to prove i) by structural induction on C (see [7] for the full proof). Condition 1: $\mathsf{prim}(D)$ is the intersection of the sets $\mathsf{prim}(C_i)$, thus Condition 1 holds; Condition 2: An element in $\mathsf{ex}(D)$ is of the form $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_j \sqcap \mathsf{val}(C_j)) \mid 1 \leq j \leq n\}$. Choosing $C' = C'_i$ yields $C'_i \sqcap \mathsf{val}(C_i) \sqsubseteq \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val}(C_i))$, thus also $C'_i \sqcap \mathsf{val}(C_i) \sqsubseteq \mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_j \sqcap \mathsf{val}(C_j)) \mid 1 \leq j \leq n\}$. The reasoning for Condition 3 is similar.

Theorem 5 *For every $\mathcal{ALC}$-concept description C the $\mathcal{ALE}$-approximation exists, is uniquely determined up to equivalence, and can be computed by* $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$*, i.e.,* $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \equiv \mathsf{approx}_{\mathcal{ALE}}(C)$.

Applying this to our examples, we obtain for $C_{\mathsf{ex},1}$ the normal form:

$$C_{\mathsf{ex},1} \equiv ((\exists r.A \sqcap \forall r.B) \sqcup (\exists r.A \sqcap \exists r.B \sqcap \forall r.A)),$$

and for $C_{\mathsf{ex},2}$

$$C_{\mathsf{ex},2} \equiv \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$$

one verifies that $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},1}) \equiv \exists r.(A \sqcap B)$ and $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},2}) \equiv \exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)$.

In [1], it has been shown that the lcs of two $\mathcal{ALE}$-concept descriptions can grow exponentially in the size of the given concept descriptions. Since $\mathsf{approx}_{\mathcal{ALE}}(E \sqcup F) \equiv \mathsf{lcs}(E, F)$ for $\mathcal{ALE}$-concept descriptions E and F, it immediately follows that the $\mathcal{ALE}$-approximation can grow exponentially as well. Moreover, one can show that $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ runs in double exponential time [7]. Whether or not there also exists an exponential time approximation algorithm is an open problem.

Corollary 6 *The $\mathcal{ALE}$-approximation of $\mathcal{ALC}$-concept descriptions may grow exponentially and there is a double-exponential time algorithm computing it.*

4 The difference operator

In the previous section we have seen how to compute the $\mathcal{ALE}$-approximation of a given $\mathcal{ALC}$-concept description. For such a pair C, D of approximated and approximating concept, a very natural question regards the loss of information, i.e., what aspects of C are not captured by D.

An answer to such questions requires a notion of the "difference" between concept descriptions. For instance, a comparison between the example concept $C_{\mathsf{ex},2}$ and its approximation $\exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)$ should reveal that the value restriction $\forall r.(\neg A \sqcup \neg B)$ is not captured by the approximation.

A first approach for a difference operator has been proposed by Teege [16]. Here, the difference $C - D$ of two given $\mathcal{L}$-concept descriptions with $C \sqsubseteq D$ has been defined as

$$max\{E \in \mathcal{L} \mid E \sqcap D \equiv C\}$$

where the maximum is defined with respect to subsumption. Since $\mathcal{ALC}$ provides full negation, a most general concept E with $E \sqcap D \equiv C$ is always $C \sqcup \neg D$. Consequently, Teege's difference operator would return

$$(\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B))$$
$$\sqcup \neg(\exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A))$$

as the difference between $C_{\mathsf{ex},2}$ and its approximation, which obviously does not help a human user to ascertain the information lost by the approximation.

The example illustrates that it may be promising to look for a syntactic minimum instead of a semantic maximum in order to find a compact representation of the difference of two concepts.

4.1 Subdescription Ordering

In [12, 2], a so-called *subdescription ordering* on $\mathcal{ALE}$-concept descriptions has been proposed to deal with syntactical redundancies. In order to extend this to our case we need to introduce an analogous ordering on $\mathcal{ALC}$-concepts. The idea is to obtain a subdescription of some $\mathcal{ALC}$-concept description C by means of two kinds of modifications. Firstly, by making inconsistencies explicit; and secondly, by removing disjuncts and conjuncts, and by replacing some existential or value restrictions by their respective subdescriptions. Formally, this leads to the following definition.

Definition 7 *Let C, D be $\mathcal{ALC}$-concept descriptions in $\mathcal{ALC}$-normal form. Let $C = C_1 \sqcup \cdots \sqcup C_n$. Then, $D \preceq_d C$ iff $D = \bot$ or D is obtained from C by performing some of the following steps:*

1. *Remove some disjuncts C_i for $1 \leq i \leq n$,*
2. *for every remaining C_i:*
   (a) *remove some conjuncts $A \in \mathsf{prim}(C_i)$,*
   (b) *remove some conjuncts $\exists r.C_i'$ with $C_i' \in \mathsf{ex}(C_i)$,*
   (c) *remove the conjunct $\forall r.\mathsf{val}(C_i)$,*
   (d) *for every remaining $C_i' \in \mathsf{ex}(C_i) \cup \{\mathsf{val}(C_i)\}$: replace C_i' by C_i'' with $C_i'' \preceq_d C_i'$*

If everything is removed from C, the resulting concept is $\top$. As an example, consider the equivalent concept descriptions $C := \exists r.A \sqcap \forall r.\neg B$ and $D := (\exists r.(A \sqcup B) \sqcap \forall r.\neg B) \sqcup (\exists r.\neg A \sqcap \forall r.A)$. By removing the last disjunct from D and the last disjunct in the remaining existential restriction we find $C \prec_d D$.

Based on the subdescription ordering, we can provide the new definition of the difference operator.

Definition 8 *Let C be an $\mathcal{ALC}$-concept description and D an $\mathcal{ALE}$-concept description. The* difference *$C - D$ of C and D is defined as a minimal (w.r.t. $\preceq_d$) $\mathcal{ALC}$-concept description E with $E \sqcap D \equiv C \sqcap D$.*

Intuitively, the idea is to remove all subdescriptions from C which are either redundant or already present in D. It should be noted that in case of $C \sqsubseteq D$, and thus, $C \sqcap D \equiv C$, the only difference to Teege's difference operator is that the minimum w.r.t. $\preceq_d$ is used instead of the maximum w.r.t. $\sqsubseteq$. Finally, note that the difference between C and D is not a priori uniquely determined. By abuse of language and notation, we will still refer to *the* difference $C - D$ (see also Theorem 9). Coming back to the example at the beginning of the section, the difference (according to Definition 8) between $C_{\mathsf{ex},2}$ and its approximation is $\forall r.(\neg A \sqcup \neg B)$, as desired.

4.2 Computing Differences

Having defined our difference operator, we need to devise an algorithm to actually compute the difference $C - D$. In [12], an algorithm has been proposed to compute the difference $C - D$ of $\mathcal{ALE}$-concept descriptions C and D. Extending this algorithm to the case of $\mathcal{ALC}$-concept descriptions C yields our definition of the algorithm $\mathsf{c\text{-}diff}$ as depicted in Figure 2.

If C is a disjunction of subconcepts C_i then the difference between C and D is obtained by firstly computing the differences between the disjuncts and D and then eliminating the semantically redundant resulting disjuncts. In general, the following three properties can be shown for every computation of $\mathsf{c\text{-}diff}(C, D)$.

Theorem 9 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form and D be an $\mathcal{ALE}$-concept description. Then,*

1. *$\mathsf{c\text{-}diff}(C, D) \sqcap D \equiv C \sqcap D$.*
2. *If C is an $\mathcal{ALE}$-concept description, then $C - D$ is uniquely determined modulo associativity and commutativity of concept conjunction, and $C - D$ and $\mathsf{c\text{-}diff}(C, D)$ coincide.*
3. *Given an oracle for subsumption, $\mathsf{c\text{-}diff}(C, D)$ runs in polynomial time in the size of C and D.*

Thus, in case C is an $\mathcal{ALE}$-concept description, $\mathsf{c\text{-}diff}$ exactly computes the difference $C - D$ (Theorem 9, 2.). If C is an $\mathcal{ALC}$-concept description, we know that $\mathsf{c\text{-}diff}$ does not remove too much from C (Theorem 9, 1.). However, $\mathsf{c\text{-}diff}$ might not have computed the exact difference $C - D$. Thus, $\mathsf{c\text{-}diff}$ is a heuristic algorithm for computing the difference between an $\mathcal{ALC}$- and an $\mathcal{ALE}$-concept description. Nevertheless, the following examples illustrate that $\mathsf{c\text{-}diff}$ works quite satisfactorily even in the general case. This is also supported by our experiences with a prototype implementation of $\mathsf{c\text{-}diff}$ in the chemical process engineering application.

Consider the example concept $C_{\mathsf{ex},1}$ and its $\mathcal{ALE}$-approximation $\exists r.(A \sqcap B)$. In order to compute the difference $\mathsf{c\text{-}diff}(C_{\mathsf{ex},1}, \exists r.(A \sqcap B))$, $C_{\mathsf{ex},1}$ firstly

```
Input: $\mathcal{ALC}$-concept description $C = C_1 \sqcup \cdots \sqcup C_n$ in $\mathcal{ALC}$-normal form,
       $\mathcal{ALE}$-concept description $D$
Output: $\mathsf{c\text{-}diff}(C, D)$
  • If $C \sqcap D \equiv \bot$, then $\mathsf{c\text{-}diff}(C, D) := \bot$;
  • If $n > 1$, then let $\mathsf{c\text{-}diff}(C, D) := \bigsqcup_{i=1}^{n} \mathsf{c\text{-}diff}(C_i, D)$ and iteratively remove $\mathsf{c\text{-}diff}(C_j, D)$ from the disjunction
    in case $\mathsf{c\text{-}diff}(C_j, D) \sqsubseteq \bigsqcup_{i \neq j} \mathsf{c\text{-}diff}(C_i, D)$;
  • If $n = 1$, then $\mathsf{c\text{-}diff}(C, D) :=$
        $$\bigsqcap_{A \in \mathsf{prim}(C) \setminus \mathsf{prim}(D)} A \;\sqcap\; \forall r.\mathsf{c\text{-}diff}(\mathsf{val}(C), \mathsf{val}(D)) \;\sqcap\; \bigsqcap_{E \in \mathcal{E}'_r} \exists r.E$$
    where the value restriction is omitted in case $\mathsf{c\text{-}diff}(\mathsf{val}(C), \mathsf{val}(D)) \equiv \top$ and $\mathcal{E}'_r$ is computed as follows:
    Let $\mathcal{E}_r = \{C'_1, \ldots, C'_n\} := \mathsf{ex}(C)$.
    For $i = 1$ to $n$ do begin
      If (i) there exists $C' \in \mathcal{E}_r \setminus \{C'_i\}$ with $\mathsf{val}(D) \sqcap \mathsf{val}(C) \sqcap C' \sqsubseteq C'_i$, or
         (ii) there exists $D' \in \mathsf{ex}(D)$ with $\mathsf{val}(D) \sqcap \mathsf{val}(C) \sqcap D' \sqsubseteq C'_i$
      then $\mathcal{E}_r := \mathcal{E}_r \setminus \{C'_i\}$
    end
    $\mathcal{E}'_r := \{E^* \mid E \in \mathcal{E}_r\}$ where $E^* := \mathsf{c\text{-}diff}(E, \mathsf{val}(C) \sqcap \mathsf{val}(D))$, if $\mathsf{val}(C)$ is an $\mathcal{ALE}$-concept description, and
    $E^* := \mathsf{c\text{-}diff}(E, \mathsf{val}(D))$ otherwise.
```

Figure 2: The algorithm $\mathsf{c\text{-}diff}(C, D)$.

has to be transformed into $\mathcal{ALC}$-normal form, yielding $(\forall r.B \sqcap \exists r.A) \sqcup (\exists r.B \sqcap \forall r.A \sqcap \exists r.A)$. We now have to compute $\mathsf{c\text{-}diff}(\forall r.B \sqcap \exists r.A, \exists r.(A \sqcap B))$ and $\mathsf{c\text{-}diff}(\exists r.B \sqcap \forall r.A \sqcap \exists r.A, \exists r.(A \sqcap B))$. For the first expression, Condition 3(b) causes $\exists r.A$ to be removed. As no other existential restriction is left, the first expression evaluates to $\forall r.B$. The second expression similarly yields $\forall r.A$. We finally obtain $\forall r.A \sqcup \forall r.B$, which is exactly $C_{\mathsf{ex},1} - \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},1})$. Analogously, one can verify that $\mathsf{c\text{-}diff}(C_{\mathsf{ex},2}, \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},2})) = C_{\mathsf{ex},2} - \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},2}) = \forall r.(\neg A \sqcup \neg B)$.

5 Prototypical Implementations

We have evaluated a first prototypical implementation of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ realized in Lisp and using the FaCT system [11] as an underlying subsumption tester. Our implementation of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ utilizes the optimized lcs implementation described in [17]. In contrast to the $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ algorithm specified in Figure 1 our implementation reduces the number of lcs calls in advance. For many concept descriptions in $\mathcal{ALC}$-normal form it is likely that disjuncts share the same existential restrictions due to the normalization. These existential restrictions cause unnecessary lcs calls when approximating the existential restrictions. Some of the combinations from the Cartesian product of the existential restrictions yield argument sets for the lcs that are supersets of other combinations. These supersets yield more general and therefore redundant lcs concept descriptions. For example computing the approximation of the concept description $((A \sqcup \exists r.A) \sqcap (\exists r.B \sqcap \exists r.C)$ induces the lcs calls: $\mathsf{lcs}(A, B), \mathsf{lcs}(A, C), \mathsf{lcs}(B, B), \mathsf{lcs}(B, C), \mathsf{lcs}(C, B), \mathsf{lcs}(C, C)$ in a naive realization. However, only the trivial combinations $\mathsf{lcs}(B, B)$ and $\mathsf{lcs}(C, C)$ add existential restrictions to the approximation which are not subsumed by the other combinations. Therefore, in this case, the existential restrictions can be obtained without using the lcs at all. So, in order to obtain the correct approximation in general, it suffices to compute the lcs only of those combinations that do not have a superset among the combinations. This method is employed in our implementation, we compute first the minima (w.r.t. subset) of the set of combinations and then apply the lcs to the remaining combinations.

We applied $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ to $\mathcal{ALC}$-concepts from a TBox derived from our application in chemical process engineering. This application TBox contains 120 concepts and 40 roles. Surprisingly, for our unfolded input concepts with concept sizes up to 740, it turned out that the approximations were always smaller than their unfolded input concepts. The approximations had an average concept size of 81 and they had just a third of the size of the unfolded input concepts on the average. Each of the test concept descriptions was approximated within less than 3 seconds runtime. Unfortunately, our implementation ran out of memory computing approximations of some randomly generated $\mathcal{ALC}$-concepts of similar size, but consisting of big disjunctions with more than 6 disjuncts.

So, our prototypical implementation of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ indicates that, despite the high theoretical complexity, the approximation inference might be practicable for cases relevant in applications. Further optimizations are of course necessary. Standard optimization techniques as lazy unfolding are very likely to highly improve the performance for run-times as well as for sizes of returned concepts.

We have implemented a prototype for the $\mathsf{c\text{-}diff}$ algorithm in Lisp. For a first evaluation we applied the $\mathsf{c\text{-}diff}$ implementation to test concepts derived from our process engineering TBox. More precisely, we applied $\mathsf{c\text{-}diff}$ to the same $\mathcal{ALC}$-concept descriptions used for the evaluation of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ together with their approximations generated by our $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ implementation. For these test cases the $\mathsf{c\text{-}diff}$ implementation returned concept descriptions with an average size of 170 and a maximum size of 630. Thus, it turned out that the concept size of the difference between original concept description and its approximation is bigger than the approximation itself in many cases. Computing the difference took 2 seconds on the average and each difference was computed within 6.5 seconds runtime. Unlike $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ this prototypical implementation behaved also well on randomly generated concept descriptions. But for practical applications of this non-standard inference powerful optimizations are still necessary. Moreover, the output concept descriptions need to be smaller and more compact in order to be readable and comprehensible for a human user.

6 Conclusion and Future Work

We have investigated approximation as a new inference problem for DLs. As a main technical result, a double-exponential time algorithm computing upper approximations of $\mathcal{ALC}$-concepts in $\mathcal{ALE}$ has been devised. We have also introduced a syntax-based difference operator to measure the accuracy of approximations and an efficient heuristic algorithm, which uses subsumption testing as an oracle, to actually compute the difference of concepts.

Our first evaluation of the implementations of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ and $\mathsf{c\text{-}diff}$ indicates that there is need for further optimization. Even more important, since the concepts returned by both algorithms are quite big and hard to read and comprehend by a human user, it is necessary to rewrite the concepts using the concept definitions from the underlying $\mathcal{ALC}$-TBox to obtain smaller concepts. To this purpose, one needs to extend the existing rewriting approach for $\mathcal{ALE}$ [2] to $\mathcal{ALC}$.

Another direction for future work is of course to extend our results to more expressive DLs. Since there already exists an lcs algorithm for $\mathcal{ALEN}$ [13], i.e., $\mathcal{ALE}$ plus number restrictions, the approximation algorithm presented here can be extended to the approximation of $\mathcal{ALCN}$-concept descriptions by concept descriptions in $\mathcal{ALEN}$ (or sublanguages thereof) as shown in [8]. It remains as future work to adapt the difference operator to this pair of description logics.

References

[1] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.

[2] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 297–308, San Francisco, 2000. Morgan Kaufmann.

[3] P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: Transparent access to multiple bioinformatics information sources. In J. Glasgow, T. Littlejohn, F. Major, R. Lathrop, D. Sankoff, and C. Sensen, editors, *6th Int. Conf. on Intelligent Systems for Molecular Biology*, pages 25–34, Montreal, Canada, 1998. AAAI Press, Menlo Park.

[4] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. Oiled: a reason-able ontology editor for the semantic web. In F. Baader, G. Brewka, and Th. Eiter, editors, *Proceedings of the Joint German/Austrian Conference on AI (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 396–408, Vienna, Austria, 2001. Springer–Verlag.

[5] A. Borgida and D. W. Etherington. Hierarchical knowledge bases and efficient disjunctive reasoning. In H. J. Levesque, R. J. Brachman and R. Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 33–43, Toronto, Canada, May 1989. Morgan Kaufmann.

[6] S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (KIDLWS'01)*, number 44 in CEUR-WS, Vienna, Austria, September 2001. RWTH Aachen. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-44/.

[7] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. LTCS-Report 01-06, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2001. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[8] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximating $\mathcal{ALCN}$-Concept Descriptions. See http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html. To appear.

[9] W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 754–760, San Jose, CA, July 1992. MIT Press.

[10] F. M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. Spaccamela. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53(2–3):309–327, 1992.

[11] I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 636–645. Morgan Kaufmann, San Francisco, California, 1998.

[12] R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 2001.

[13] R. Küsters and R. Molitor. Computing least common subsumers in $\mathcal{ALEN}$. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 219–224. Morgan Kaufman, 2001.

[14] U. Sattler. *Terminological knowledge representation systems in a process engineering application.* PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 1998.

[15] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[16] G. Teege. Making the difference: A subtraction operation for description logics. In P. Torasso J. Doyle, E. Sandewall, editor, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 540–550, Bonn, FRG, May 1994. Morgan Kaufmann.

[17] A.-Y. Turhan and R. Molitor. Using lazy unfolding for the computation of least common subsumers. In *Proceedings of the International Workshop in Description Logics 2001 (DL2001)*, Stanford, USA, August 2001.

7 Appendix

Theorem 10 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form (as specified in Definition 3) and D an $\mathcal{ALE}$-concept description. Then, $C \sqsubseteq D$ iff $C \equiv \bot$, or $D \equiv \top$, or for all $i = 1, \dots, n$ it holds that*

1. $\mathsf{prim}(D) \subseteq \mathsf{prim}(C_i)$, *and*
2. *for all $D' \in \mathsf{ex}(D)$ there exists $C' \in \mathsf{ex}(C_i)$ such that $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$, and*
3. $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$.

PROOF. ($\Rightarrow$) Assume $\bot \sqsubset C \sqsubseteq D \sqsubset \top$.

- Assume $\mathsf{prim}(D) \not\subseteq \mathsf{prim}(C_i)$ for one i. Then there exists an $A \in \mathsf{prim}(D) \setminus \mathsf{prim}(C_i)$. By definition of the $\mathcal{ALC}$-normal form, C_i is consistent. We may therefore consider a canonical interpretation I of C_i. By definition, the individual $d_{C_i} \in \Delta^I$ for C_i does not occur in A^I, since $A \notin \mathsf{prim}(C_i)$. Thus, $d \notin D^I$ and therefore $C \not\sqsubseteq D$, in contradiction to our assumption.

- Assume for one $D' \in \mathsf{ex}(D)$ that one i exists such that for all $C' \in \mathsf{ex}(C_i)$ it holds that $C' \sqcap \mathsf{val}(C_i) \not\sqsubseteq D'$. Since C_i is consistent, every $C' \in \mathsf{ex}(C_i)$ has a tree model $I_{C'}$ where $d_{C'} \in (C' \sqcap \mathsf{val}(C_i))^{I_{C'}} \setminus (D')^{I_{C'}}$. Without loss of generality, we may assume disjoint domains, i.e., $\Delta^{I_{C'}} \cap \Delta^{I_{C''}} = \emptyset$ for two different $C', C'' \in \mathsf{ex}(C_i)$. We may now construct a new model I over the domain $\Delta^I = \{d\} \uplus \bigcup_{C' \in \mathsf{ex}(C_i)} \Delta^{I_{C'}}$ with the following properties: (1) For the role r, define $r^I := \{(d, d_{C'}) \mid C' \in \mathsf{ex}(C_i)\} \cup \bigcup_{C' \in \mathsf{ex}(C_i)} r^{I_{C'}}$. (2) For every (negated) atomic concept $A \in N_C \cup \{\neg A \mid A \in N_C\}$, define the interpretation of A as $A^I := \{d \mid A \in \mathsf{prim}(C_i)\} \cup \bigcup_{C' \in \mathsf{ex}(C_i)} A^{I_{C'}}$. Note that the first expression only states that $d \in A^I$ iff $A \in \mathsf{prim}(C_i)$.

  It is easy to see that $d \in C^I$. On the other hand $d \notin D^I$, because $(D')^{I_{C'}}$ was excluded explicitly from every $I_{C'}$. Consequently, we have $d \notin (\exists r.D)^I$.

- Assume $\mathsf{val}(C_i) \not\sqsubseteq \mathsf{val}(D)$ for one i. Thus, $\mathsf{val}(C_i)$ has a tree model I_{val} such that $d_{val} \in \mathsf{val}(C_i)^{I_{val}} \setminus \mathsf{val}(D)^{I_{val}}$. We can now extend the model I introduced for the previous case by adding d_{val} as an r-successor of d. Again, assume $\Delta^I \cap \Delta^{I_{val}} = \emptyset$. Then, define I' as follows: $\Delta^{I'} := \Delta^I \cup \Delta^{I_{val}}$. (1) For the role r, we define $r^{I'} := \{(d, d_{val})\} \cup r^I \cup r^{I_{val}}$. (2) For every (negated) atomic concept A, $A^{I'}$ is simply the union of the previous models, i.e., $A^{I'} := A^I \cup A^{I_{val}}$. As a result, we still have $d \in (C_i^{I'})$ for all i and thus $d \in C^{I'}$ but on the other hand $d \notin D^{I'}$.

($\Leftarrow$) If $C \equiv \bot$ or $D \equiv \top$ then it is easy to see that the claim holds. Otherwise, let $i \in \{1, \dots, n\}$. It is sufficient to show that $C_i \sqsubseteq D$. Let $x \in C_i^I$ for any interpretation I of C_i. Show: $x \in D^I$.

- By assumption, $x \in A^I$ for every $A \in \mathsf{prim}(C_i)$. The inclusion $\mathsf{prim}(D) \subseteq \mathsf{prim}(C_i)$ thus implies $x \in A^I$ for every $A \in \mathsf{prim}(D)$.

- Consider an arbitrary $D' \in \mathsf{ex}(D)$. By assumption, we know that there is an $C' \in \mathsf{ex}(C_i)$ with $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$. Since $x \in (\exists r.C' \sqcap \forall r.\mathsf{val}(C_i))^I$, this implies $x \in (\exists r.D')^I$.

- As $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$ and $x \in (\mathsf{val}(C_i))^I$, it holds that $x \in (\mathsf{val}(D))^I$.

The definition of conjunction yields $D^I = \bigcap_{A \in \mathsf{prim}(D)} A^I \cap \bigcap_{D' \in \mathsf{ex}(D)} (\exists r.D')^I \cap (\mathsf{val}(D))^I$, concluding the argument. ∎

Theorem 5

PROOF. Without loss of generality, we may assume C in $\mathcal{ALC}$-normal form since (1) the algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ starts by computing the $\mathcal{ALC}$-normal form of its input and (2) $\top$ and $\bot$ are represented uniquely in $\mathcal{ALC}$-normal form.

1. Show $C \sqsubseteq \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$. To this end, we show by induction over the structure of C that the conditions for subsumption from Theorem 10 hold.

If $C \in \{\bot, \top\}$ then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) = C$ which trivially satisfies the subsumption conditions. Otherwise, we may assume as induction hypothesis that the claim holds for the subterms of C occurring in existential and value restrictions. For C we therefore find that:

- By definition of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$, the set $\mathsf{prim}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$ of primitive concepts equals $\bigcap_{i=1}^n \mathsf{prim}(C_i)$ which is always a subset of $\subseteq \mathsf{prim}(C)$.

- Show: for $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$ and for all i there exists some existential restriction $C' \in \mathsf{ex}(C_i)$ such that $C' \sqcap \mathsf{val}(C_i)$ is subsumed by $\mathsf{lcs}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$.

  Pick $C' = C_i'$. By induction hypothesis it holds that $C' \sqcap \mathsf{val}(C_i)$ is subsumed by the approximation $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}(C_i))$. The definition of the lcs now guarantees that $C' \sqcap \mathsf{val}(C_i)$ is also subsumed by $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$.

- Show: $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$ for every i. By induction hypothesis we already know that the value restriction $\mathsf{val}(C_i)$ is subsumed by the approximation $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i))$ for every i. Consequently, for the lcs we find that $\mathsf{val}(C_i)$ is subsumed by $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$ for every i.

2. Show $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \sqsubseteq C$. Without loss of generality, let D be in $\mathcal{ALE}$-normal form. Proof by induction over the structure of C.

If $C \in \{\bot, \top\}$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) = C$ which is the least concept subsuming C.

Otherwise, we may assume that the claim holds for the subterms of C occurring in existential and value restrictions. If $D = \top$, then trivially $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \sqsubseteq D$. Otherwise, the subsumption $C \sqsubseteq D$ induces the following facts:

- $\mathsf{prim}(D) \subseteq \mathsf{prim}(C_i)$ for every i. As the set $\mathsf{prim}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$ of primitive concepts is defined as the intersection of every $\mathsf{prim}(C_i)$, this implies $\mathsf{prim}(D) \subseteq \mathsf{prim}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$.
- For all $D' \in \mathsf{ex}(D)$ and for all i there is an existential restriction $C' \in \mathsf{ex}(C_i)$ with $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$. The induction hypothesis now guarantees that $C' \sqcap \mathsf{val}(C_i)$ is subsumed by $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C' \sqcap \mathsf{val}(C_i)) \sqsubseteq D$ for every i. Consequently, for the lcs it holds that $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\} \sqsubseteq D$.
- For all i we have $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$. By induction hypothesis we know that the value restriction $\mathsf{val}(C_i)$ is subsumed by the approximation $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \sqsubseteq \mathsf{val}(D)$. Hence, we similarly find $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\} \sqsubseteq \mathsf{val}(D)$.

■

Corollary 11 *The algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ is a double-exponential time algorithm, i.e., for a given $\mathcal{ALC}$-concept description the computation of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$ takes at most double exponential time in the size of C.*

PROOF. The algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ expects its input in $\mathcal{ALC}$-normal form. Nevertheless, instead of transforming C into normal form before applying $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ we may also do the necessary transformation on the fly for every role level currently visited. Let $|C| = n$. The computation of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$ starts by transforming C into $D := C_1 \sqcup \cdots \sqcup C_m$—such that every C_i has no disjunction on the topmost role level—but does not modify the lower role levels. The concept D can thus have exponentially many ($2^{p(n)}$ for some polynomial p) disjuncts on the topmost level each of which is limited in size by n.

According to the recursive structure of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ the following expressions must be computed:

1. the conjunction $\bigsqcap_{A \in \bigcap_i \mathsf{prim}(C_i)} A$ of primitive concepts;
2. an existential restriction $\exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq m\}$ for every tuple $(C'_1, \ldots, C'_m)$ with $C'_i \in \mathsf{ex}(C_i)$;
3. one value restriction $\forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \leq i \leq m\}$.

Obviously, Step 1 can be computed in polynomial time in the size of D and thus in exponential time in n.

As D has exponentially many disjuncts C_i with a linear number of existential restrictions C'_i, the number of existential restrictions to be computed in Step 2 is double exponential in n. For every such existential restriction an lcs of a set of exponential cardinality must be computed. Each element of such a set is of the form $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val}(C_i))$. Hence, $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ is recursively invoked on a concept description of size bounded by the size of C and with a role depth decreased by one. Thus, the computation tree of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ (with the lcs's not evaluated for the time being), is of size double exponential in the size of C. In other words, if the lcs is not evaluated, $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ runs in double exponential time. We need to show that evaluating the lcs's occurring in the computation tree, does not increase the complexity.

We start to evaluate the lcs's from the bottom to the top of the computation tree for $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$. Every lcs operation in the tree has an exponential number of arguments and every argument is of size double exponential in $|C|$. Moreover, one can easily show that every argument is not only in $\mathcal{ALE}$-normal form, but also has the following properties:

- It contains no subexpression of the form $P \sqcap \neg P$ (where $P \in N_C$), $E \sqcap \bot$, $E \sqcap \top$, $\exists r.\bot$, or $\forall r.\top$.
- For every subexpression of the form $\exists r.E \sqcap \forall r.F$ it holds that $E \sqsubseteq F$, i.e., the value restriction has been propagated into the existential restriction already.

This holds because according to the definition, all concepts returned by $\text{c-approx}_{\mathcal{ALE}}$ are have these properties. As shown in [1], the size of the lcs can therefore be bounded by the product of the sizes of the arguments. Thus, evaluating the lcs's on the bottom level yields concept descriptions of size at most double exponential. This evaluation process is iterated on every level of the computation tree for $\text{c-approx}_{\mathcal{ALE}}(C)$ where lcs's occur. Since the depth of this tree is bounded by $|C|$ (more precisely, by the role depth of C), the whole evaluation can be carried out in double-exponential time. ■

Theorem 9

Proof. 1. Proof by ind. over the structure of C.

- $C \in \text{prim}(C)$: Then the conjunction $\text{c-diff}(C, D) \sqcap D$ is equivalent to $P_{C\setminus D} \sqcap D$, where $P_{C\setminus D}$ is the conjunction over all primitive concepts in $\text{prim}(C) \setminus \text{prim}(D)$. Let $P_{C\cap D}$ denote the concjunction over all primitive concepts in $\text{prim}(C) \sqcap \text{prim}(D)$. As $D \sqsubseteq P_{C\cap D}$, the term $\text{c-diff}(C, D) \sqcap D$ is still equivalent to the conjunction $P_{C\setminus D} \sqcap P_{C\cap D} \sqcap D$. This expression, however, is equivalent to $C \sqcap D$.

- $C = C_1 \sqcup C_2$: Without loss of generality, we may assume exactly two disjuncts on the top-level of C. By definition, even after removing redundant disjuncts, $\text{c-diff}((C_1 \sqcup C_2), D)$ is equivalent to $\text{c-diff}(C_1, D) \sqcup \text{c-diff}(C_2, D)$. Hence, the conjunction $\text{c-diff}((C_1 \sqcup C_2), D) \sqcap D$ can be simplified to $(\text{c-diff}(C_1, D) \sqcap D) \sqcup (\text{c-diff}(C_2, D) \sqcap D)$. According to the induction hypothesis, this is equivalent to $(C_1 \sqcap D) \sqcup (C_2 \sqcap D)$ which simplifies to $(C_1 \sqcup C_2) \sqcap D$.

- No disjunction on the top-level of C: Show $\text{c-diff}(C, D) \sqcap D \equiv C \sqcap D$. According to the characterization of subsumption (Theorem 10), three conditions must hold for equivalence:

  (1.) The set $\text{prim}(\text{c-diff}(C, D) \sqcap D)$ of primitive concepts is equal to the intersection $\text{prim}(\text{c-diff}(C, D)) \cap \text{prim}(D)$ which by definition is the intersection of $(\text{prim}(C) \setminus \text{prim}(D))$ and $\text{prim}(D)$. This is equal to $\text{prim}(C) \cap \text{prim}(D)$, the set of primitive concepts in the conjunction $C \sqcap D$.

  (2.) Show ($\sqsubseteq$). Consider an existential restriction $F' \in \text{ex}(C \sqcap D)$. We have to find an existential restriction $E' \in \text{ex}(\text{c-diff}(C, D) \sqcap D)$ with $E' \sqcap \text{val}(\text{c-diff}(C, D) \sqcap D) \sqsubseteq F'$. From the previous case we know that $\text{val}(\text{c-diff}(C, D) \sqcap D)$ is equivalent to $\text{val}(C \sqcap D)$. Since $\text{ex}(C \sqcap D)$ is equal to the union $\text{ex}(\text{c-diff}(C, D)) \cup \text{ex}(D)$ we may distinguish two cases.

  If $F' \in \text{ex}(D)$ then we can select $E' := F'$, because it also occurs in the set $\text{ex}(\text{c-diff}(C, D) \sqcap D)$ which is the conjunction of the concept descriptions in $\text{ex}(\text{c-diff}(C, D)) \cup \text{ex}(D)$. We thus find that the conjunction $E' \sqcap \text{val}(\text{c-diff}(C, D) \sqcap D)$ is subsumed by F'.

  If $F' \in \text{ex}(C) \setminus \text{ex}(D)$, then Conditions (i) and (ii) in the definition of the algorithm $\text{c-diff}(C, D)$ guarantee that there exists an existential restriction $\tilde{E}' \in \text{ex}(\text{c-diff}(C, D))$ with the following properties. If $\text{val}(C)$ is an $\mathcal{ALE}$-concept description then $\tilde{E}'$ is of the form $\text{c-diff}(E', (\text{val}(D) \sqcap \text{val}(C)))$ and the conjunction $E' \sqcap \text{val}(D) \sqcap \text{val}(C)$ is subsumed by F'. According to the induction hypothesis, $\text{c-diff}(E', (\text{val}(D) \sqcap \text{val}(C))) \sqcap \text{val}(D) \sqcap \text{val}(C)$ is equivalent to $E' \sqcap \text{val}(D) \sqcap \text{val}(C)$. Consequently, we find that $\tilde{E}' \sqcap \text{val}(C) \sqcap \text{val}(D)$ is subsumed by F'. It is easy to see that $\text{val}(C) \sqcap \text{val}(D)$ is equivalent to $\text{val}(C \sqcap D)$ which again is equivalent to $\text{val}(\text{c-diff}(C, D) \sqcap D)$ as we know from above. Hence, we have found an existential restriction $\tilde{E}'$ such that the conjunction $\tilde{E}' \sqcap \text{val}(\text{c-diff}(C, D) \sqcap D)$ is subsumed by F'. If D is no $\mathcal{ALE}$-concept description then $\tilde{E}'$ is of the form $E' \sqcap \text{val}(D)$. This case is analogous to the previous one.

  Show ($\sqsupseteq$). In analogy to the case ($\sqsubseteq$), consider some existential restriction $E' \in \text{ex}(\text{c-diff}(C, D) \sqcap D)$. We have to find an existential restriction $F' \in \text{ex}(C \sqcap D)$ such that $F' \sqcap \text{val}(C \sqcap D)$ is subsumed by E'. Again, we have two cases to discriminate.

  If $E' \in \text{ex}(D)$, then we can again select $F' := E'$ which also occurs in the set $\text{ex}(C \sqcap D)$ of existential restrictions.

  If $E' \in \text{ex}(\text{c-diff}(C, D)) \setminus \text{ex}(D)$, then Condition (ii) guarantees that an existential restriction $F' \in \text{ex}(D) \subseteq \text{ex}(C \sqcap D)$ exists such that the conjuntion $F' \sqcap \text{val}(C) \sqcap \text{val}(D)$ is subsumed by E'. As seen above, $\text{val}(C) \sqcap \text{val}(D)$ is equivalent to $\text{val}(C \sqcap D)$ which concludes the argument.

  (3.) By induction hypothesis, the conjunction $\text{c-diff}(\text{val}(C), \text{val}(D)) \sqcap \text{val}(D)$ is equivalent to $\text{val}(C) \sqcap \text{val}(D)$. By definition, $\text{val}(C \sqcap D)$ is equivalent to $\text{val}(C) \sqcap \text{val}(D)$, which concludes this case.

2. This result was already shown in [12].

3. The proof can be found in our technical report, see [7]. ■