RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

Approximation and Difference in Description Logics

Sebastian Brandt Ralf Küsters Anni-Yasmin Turhan

LTCS-Report 01-06

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

Approximation and Difference in Description Logics

Sebastian Brandt[†], Ralf Küsters[‡], and Anni-Yasmin Turhan[†]

[†] LuFG Theoretische Informatik,
RWTH Aachen

[‡] Institut für Informatik und Praktische Mathematik,
Christian-Albrechts-Universität zu Kiel

November 2001

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 description 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.

Contents

1 Introduction

Approximation in Description Logics (DLs) was first mentioned by Baader, Küsters, and Molitor [2] as an interesting 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:

- *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 [11, 6] for an overview). However, up to now they are mostly restricted to quite inexpressive DLs, for example to DLs that do not allow for concept disjunction. Approximation can be used to overcome this problem to some extent:

  - *Matching for expressive DLs*
    For example, the existing matching algorithms can be lifted 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 cannot 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 correct one, the accuracy of the result may still suffice.
  - *Finding commonalities of concepts with disjunction*
    Another example, which in fact was our main motivation for looking at approximation in the first place, is to combine approximation and the lcs computation: 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. Intuitively, the lcs yields the commonalities between C_1 and C_2. However, 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, a user inspecting this concept does not learn anything about the actual commonalities. 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. Again, due to the approximation step some information may be lost, nevertheless, since commonalities are made explicit, the resulting concept might even be more interesting to a user than just the disjunction of C_1 and C_2.

- *Support for 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. As an approach to 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 this approach are the TAMBIS system [3] and the ontology editor OilEd [4] built on top of the FACT DL system [10]. 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 user's level of expertise. Furthermore, the aspects not captured by the frame-based representation could be computed for further inspection by a difference operator as proposed in this work (see below).

- *Knowledge-base vivification*
  Concept disjunction in many cases increases the computational complexity of inference problems, which in particular was a problem in the early DL systems. The idea of knowledge-base vivification is to replace disjunction by the lcs of its disjuncts [5, 7]. We will show in Section 3 that such a direct substitution does not always yield the best concept expressible in the smaller DL. Hence, a more general approach to knowledge-base vivification is to approximate the original concepts defined in a DL with disjunction in a corresponding DL where disjunction cannot be expressed. The problem with both approaches is that they might lead to an exponential blow-up of the concepts, making reasoning in the smaller DL expensive too. In fact, given the highly optimized up-to-date DL systems, such as FACT [10] and RACER [9], the use of knowledge-base vivification is questionable nowadays.

The main technical result of this paper (Section 4) is to show that concept descriptions defined in the standard DL $\mathcal{ALC}$, which provides concept conjunction and disjunction, value and existential restrictions, and full negation, can be approximated (from above) in the DL $\mathcal{ALE}$, a DL without 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 and D yields a concept description E (the difference

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.

of C and D) in $\mathcal{L}_s$ such that E conjoint with D is equivalent to C, i.e., $E \sqcap D \equiv C$. 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 [14]. 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 allows for 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 5, we provide a formal definition and give an algorithm for computing the difference between $\mathcal{ALC}$- and $\mathcal{ALE}$-concept descriptions.

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 ($\neg C$) (see Table 1). 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* $\mathcal{I} = (\Delta, \cdot^I)$. The domain Δ of $\mathcal{I}$ is a non-empty set and the interpretation function $\cdot^I$ maps each concept name $A \in N_C$ to a set $A^I \subseteq \Delta$ and each role name $r \in N_R$ to a binary relation $r^I \subseteq \Delta \times \Delta$. The extension of $\cdot^I$ 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 subsumption 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 $\mathcal{ALC}$ is PSPACE-complete [13] and NP-complete in $\mathcal{ALE}$ [8].

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*

1. *$C_i \sqsubseteq C$ for all $1 \leq i \leq n$, and*
2. *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.

Some notation is needed to access the different parts of an $\mathcal{ALE}$-concept description or an $\mathcal{ALC}$-concept description where disjunction only occurs within value or existential restrictions. Given such a concept C:

- $prim(C)$ denotes the set of all (negated) concept names and the bottom concept occurring on the top-level conjunction of C.
- $\mathsf{val}_r(C) := D$, where D is a conjunction of all C' occurring in value restrictions of the form $\forall r.C'$ on the top-level of C. If there is no value restriction on top-level of C $\mathsf{val}_r(C) := \top$.
- $\mathsf{ex}_r(C) := \{C' \mid \text{there exists } \exists r.C' \text{ on the top-level of } C\}$.

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.

3 Approximation—a trivial task?

As introduced in Section 1, the upper approximation of a given $\mathcal{ALC}$-concept description C is a minimal $\mathcal{ALE}$-concept description D (w.r.t. subsumption) subsuming C. Formally, this leads to the following definition.

Definition 2 *Let C be an $\mathcal{ALC}$-concept description. An $\mathcal{ALE}$-concept description D is an* upper $\mathcal{ALE}$-approximation *(for short approx$_{\mathcal{ALE}}$) of C, iff*

1. $C \sqsubseteq D$, *and*
2. $C \sqsubseteq D'$ *and* $D' \sqsubseteq D$ *implies* $D' \equiv D$ *for every* $\mathcal{ALE}$*-concept description* 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 (and we will show that they always do), they are uniquely determined up to equivalence: If D_1 and D_2 are two upper $\mathcal{ALE}$-approximations, 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 that $D_1 \sqcap D_2 \equiv D_1 \equiv D_2$. This means that an upper $\mathcal{ALE}$-approximation D of C is the *most specific* concept in $\mathcal{ALE}$ subsuming C, i.e., $C \sqsubseteq D$, and ii) $D \sqsubseteq D'$ for every $\mathcal{ALE}$-concept description D' with $C \sqsubseteq D'$.

Obviously, the crucial point in approximating an $\mathcal{ALC}$-concept description C in $\mathcal{ALE}$ is to deal with the disjunctions occurring in C. In the very simple case of only one disjunction on the top-level of C, i.e., C is the disjunction $C_1 \sqcup C_2$ of two $\mathcal{ALE}$-concept descriptions, it is easy to see that the most specific (and thus, minimal) $\mathcal{ALE}$-concept description subsuming C is just the least common subsumer of C_1 and C_2, i.e., $\mathsf{lcs}(C_1, C_2)$. Hence, the disjunction is approximated by the lcs of the disjuncts.

It seems natural to generalize this approach to disjunctions occurring at other positions in the syntax tree of C. Thus, an approximation algorithm would traverse the syntax tree of C in a bottom-up fashion and substitute a disjunction by the lcs whenever one is found. This idea is formalized in the following definition of our first straightforward attempt to an approximation algorithm:

Definition 3 (substitute disjunctions by the lcs) *The pseudo-approximation* $\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C)$ *of an* $\mathcal{ALC}$ *concept description* C *by an* $\mathcal{ALE}$ *concept description is defined by:*

$$
\begin{aligned}
\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C) &:= C \quad \text{, if } C \equiv \bot \text{ or } C \equiv \top \text{ or } C \in \mathit{prim}(C) \\
\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_1 \sqcap \cdots \sqcap C_n) &:= \mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_1) \sqcap \cdots \sqcap \mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_n) \\
\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_1 \sqcup \cdots \sqcup C_n) &:= \mathsf{lcs}\{\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_1), \ldots, \mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_n)\} \\
\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(\exists r.C') &:= \exists r.\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C') \\
\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(\forall r.C') &:= \forall r.\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C')
\end{aligned}
$$

Unfortunately, this naive approach does not always compute the most specific $\mathcal{ALE}$-concept description subsuming C, as the following example illustrates.

Example 4 [$\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}} \not\equiv approx_{\mathcal{ALE}}$]
For atomic concepts A and B, consider $C_{\mathsf{ex},1} := (\forall r.B \sqcup (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A$.

$$\begin{aligned}\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_{\mathsf{ex},1}) &\equiv \mathsf{lcs}(\forall r.B, \exists r.B \sqcap \forall r.A) \sqcap \exists r.A \\ &\equiv \forall r.\top \sqcap \exists r.A \\ &\equiv \exists r.A\end{aligned}$$

It is easy to verify that $C_{\mathsf{ex},1} \sqsubseteq \exists r.(A \sqcap B) \sqsubset \mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}(C_{\mathsf{ex},1})$. Thus, the algorithm $\mathsf{c\text{-}approx}^{\mathrm{triv}}_{\mathcal{ALE}}$ did not find an optimal solution.

It appears that substituting the lcs locally without taking care of other aspects of $C_{\mathsf{ex},1}$ does not suffice. Another straightforward way to deal with disjunctions therefore seems to compute a set of copies of the original $\mathcal{ALC}$-concept description C where in every copy each disjunction is replaced by only one of its disjuncts. The least common subsumer of all these copies might be a candidate for the approximation. The following definition formalizes this idea.

Definition 5 (split disjunctions) *Let C be an $\mathcal{ALC}$-concept description. Then the pseudo-approximation* $\mathsf{c\text{-}approx}^{\mathrm{split}}_{\mathcal{ALE}}(C)$ *is defined as* $\mathsf{c\text{-}approx}^{\mathrm{split}}_{\mathcal{ALE}}(C) := \mathsf{lcs}\,(\mathsf{split}(C))$, *where* $\mathsf{split}(C)$ *is inductively defined as follows:*

$$\begin{aligned}\mathsf{split}(C) &:= \{C\} \quad \text{, if } C \in prim(C) \cup \{\bot, \top\} \\ \mathsf{split}(C_1 \sqcap \cdots \sqcap C_n) &:= \{D_1 \sqcap \cdots \sqcap D_n \mid D_i \in \mathsf{split}(C_i), 1 \le i \le n\} \\ \mathsf{split}(C_1 \sqcup \cdots \sqcup C_n) &:= \mathsf{split}(C_1) \cup \cdots \cup \mathsf{split}(C_n) \\ \mathsf{split}(\forall r.C') &:= \{\forall r.D \mid D \in \mathsf{split}(C')\} \\ \mathsf{split}(\exists r.C') &:= \{\exists r.D \mid D \in \mathsf{split}(C')\}\end{aligned}$$

One can verify that the above algorithm works correctly for Example 4. The function split transforms the input concept description into the a set consisting of $\forall r.A \sqcap \exists r.B$ and $\exists r.A \sqcap \forall r.B \sqcap \exists r.B$ the lcs of which yields $\exists r.(A \sqcap B)$. Nevertheless, other examples exist where the resulting concept does not even subsume the input.

Example 6 [$\mathsf{c\text{-}approx}^{\mathrm{split}}_{\mathcal{ALE}} \not\equiv approx_{\mathcal{ALE}}$]
For atomic concepts A and B, let $C_{\mathsf{ex},2} := \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$. Applying the

algorithm $\mathsf{c\text{-}approx}^{\mathrm{split}}_{\mathcal{ALE}}$ to $C_{\mathsf{ex},2}$ yields the following result.

$$\begin{aligned}
\mathsf{c\text{-}approx}^{\mathrm{split}}_{\mathcal{ALE}}(C_{\mathsf{ex},2}) &\equiv \mathsf{lcs}\ (\mathsf{split}(C_{\mathsf{ex},2})) \\
&\equiv \mathsf{lcs}\{\exists r.A \sqcap \exists r.B \sqcap \forall r.\neg A\ ,\ \exists r.A \sqcap \exists r.B \sqcap \forall r.\neg B\} \\
&\equiv \exists r.\mathsf{lcs}\{A \sqcap \neg A, A \sqcap \neg B\} \sqcap \\
&\quad \exists r.\mathsf{lcs}\{A \sqcap \neg A, B \sqcap \neg B\} \sqcap \\
&\quad \exists r.\mathsf{lcs}\{B \sqcap \neg A, A \sqcap \neg B\} \sqcap \\
&\quad \exists r.\mathsf{lcs}\{B \sqcap \neg A, B \sqcap \neg B\} \sqcap \\
&\quad \forall r.\mathsf{lcs}\{\neg A, \neg B\} \\
&\equiv \exists r.(A \sqcap \neg B)\ \sqcap\ \exists r.\bot\ \sqcap\ \ldots \\
&\equiv \bot
\end{aligned}$$

The examples given so far suggest that two properties are important for a successful approximation algorithm. Firstly, $\mathcal{ALC}$-concept descriptions must be normalized before approximation, i.e., conjunctions must be distributed over disjuncts; and secondly, value restrictions must be propagated to existential restrictions. Thus, expressions of the form of $A \sqcap (B \sqcup C)$ are turned into $(A \sqcap B) \sqcup (A \sqcap C)$, and $\exists r.A \sqcap \forall r.B$ is transformed into $\exists r.(A \sqcap B) \sqcap \forall r.B$. In the following section, we will see how to employ these ideas in the actual approximation algorithm.

4 Approximation

In the previous section we have seen that normal forms of concept descriptions are desirable for our approach to approximation. Consequently, the following subsection will be concerned with a normal form for $\mathcal{ALC}$-concept descriptions. In Subsection 4.2, this normal form is used for a structural characterization of subsumption for the asymmetric case of an $\mathcal{ALE}$-concept description subsuming an $\mathcal{ALC}$-concept description. With these preliminaries, the upper approximation of $\mathcal{ALC}$-concept descriptions is introduced formally in Subsection 4.3.

4.1 Normal forms

For $\mathcal{ALE}$-concept descriptions, a concept-oriented normal form has been introduced in [11]. The idea is to restrict (w.l.o.g.) concept descriptions to only one value restriction per role on top-level and in every existential restriction and in value restriction.

Definition 7 *Let D be an $\mathcal{ALE}$-concept description. D is in* $\mathcal{ALE}$-normal form, *iff $D = \bot$, $D = \top$, or D is of the form*

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

where $\mathsf{val}_r(D)$ and every concept in $\mathsf{ex}_r(D)$ again are in $\mathcal{ALE}$-normal form.

Note that $prim(D)$ also contains negated atomic concepts. For instance, the $\mathcal{ALE}$-normal form of $A \sqcap \forall r.A \sqcap \exists r.B \sqcap \forall r.\neg B$ yields $A \sqcap \exists r.B \sqcap \forall r.(A \sqcap \neg B)$. Transforming an arbitrary $\mathcal{ALE}$-concept description C into $\mathcal{ALE}$-normal form does not increase its size because it suffices to conjoin value restrictions into one whenever more than one value restriction occurs in a conjunct at some position in C.

Nevertheless, inconsistencies need not be explicit in the $\mathcal{ALE}$-normal form and value restrictions are not propagated to existential restrictions. We define a so-called *propagated* $\mathcal{ALE}$-normal form to satisfy these conditions also.

Definition 8 *Let D be an $\mathcal{ALE}$-concept description. Then D is in* propagated $\mathcal{ALE}$-normal form*, iff none of the following normalization rules can be applied at any position in C.*

$$\begin{aligned}
P \sqcap \neg P &\longrightarrow \bot \quad \text{, where } P \in N_C \\
E \sqcap \bot &\longrightarrow \bot \\
\exists r.\bot &\longrightarrow \bot \\
\forall r.\top &\longrightarrow \top \\
E \sqcap \top &\longrightarrow E \\
\forall r.C \sqcap \forall r.D &\longrightarrow \forall r.(C \sqcap D) \\
\exists r.C \sqcap \forall r.D &\longrightarrow \exists r.(C \sqcap D) \sqcap \forall r.D
\end{aligned}$$

It can be shown that the propagated $\mathcal{ALE}$-normal form is in fact a specialization of the $\mathcal{ALE}$-normal form which could be generated by means of the last but one transformation rule alone. Nevertheless, transforming a concept description into propagated $\mathcal{ALE}$-normal form can result in exponentially larger concept descriptions because the last transformation rule (which performs the propagation) leads to an increase in size by copying subtrees of the syntax tree and may be applied very often.

In order to extend the (ordinary) $\mathcal{ALE}$-normal form to $\mathcal{ALC}$-concept descriptions, we have to deal with full negation and disjunction. Nevertheless, negated complex, i.e., non-atomic, concepts can be avoided by the negation normal form, where negation only occurs in front of concept names. Additionally, we require the bottom concept to be represented uniquely and every disjunction on every role level to be in disjunctive normal form.

Definition 9 *Let C be an $\mathcal{ALC}$-concept description. C is in* $\mathcal{ALC}$-normal form*, iff $C = \bot$, $C = \top$, or C is of the form*

$$\begin{aligned}
C &= C_1 \sqcup \cdots \sqcup C_n \\
C_i &= \bigsqcap_{A \in prim(C_i)} A \sqcap \bigsqcap_{C' \in \mathsf{ex}_r(C_i)} \exists r.C' \sqcap \forall r.\mathsf{val}_r(C_i) \quad \forall i,
\end{aligned}$$

where (1) $C_i \not\equiv \bot$ for all i and (2) $\mathsf{val}_r(C_i)$ and every concept in $\mathsf{ex}_r(C_i)$ again are in $\mathcal{ALC}$-normal form.

It is not difficult to verify that every $\mathcal{ALC}$-concept description can be transformed into $\mathcal{ALC}$-normal form. Note that, if C is in $\mathcal{ALC}$-normal form, C is also in $\mathcal{ALC}$ negation normal form. Consider the following simple example:

Example 10 For atomic concepts A, B, C, let $C := \neg\forall r.(A \sqcap B) \sqcap (B \sqcup \exists r.\neg A)$. The negation normal form of C yields $\exists r.(\neg A \sqcup \neg B) \sqcap (B \sqcup \exists r.\neg A)$. By distributing conjuncts over the disjunction we obtain $(\exists r.(\neg A \sqcup \neg B) \sqcap B) \sqcup (\exists r.(\neg A \sqcup \neg B) \sqcap \exists r.\neg A)$.

It should be noted that $\mathcal{ALC}$-normal form of a concept C can be exponentially larger than C itself. For instance, computing the disjunctive normal form of $(A_1 \sqcup B_1) \sqcap \cdots \sqcap (A_n \sqcup B_n)$ produces a concept description of exponential size in n.

4.2 Characterization of subsumption

The normal forms introduced in the previous section allow us to give a structural characterization of subsumption between an $\mathcal{ALC}$-concept description C and an $\mathcal{ALE}$-concept description D. Recall that a disjunction $C_1 \sqcup C_2$ is subsumed by D if and only if both C_i are subsumed by D. Following this idea, our characterization of subsumption reduces subsumption for a disjunction to subsumption of the respective disjuncts on every role level. Formally, we obtain the following theorem.

Theorem 11 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form and D an $\mathcal{ALE}$-concept description in $\mathcal{ALE}$-normal form. Then, $C \sqsubseteq D$ iff*
1. $C \equiv \bot$ or $D \equiv \top$, or
2. for every $i \in \{1, \ldots, n\}$ it holds that

- *$prim(D) \subseteq prim(C_i)$, and*
- *$\forall D' \in \mathsf{ex}_r(D)\ \exists C' \in \mathsf{ex}_r(C_i) : C' \sqcap \mathsf{val}_r(C_i) \sqsubseteq D'$, and*
- *$\mathsf{val}_r(C_i) \sqsubseteq \mathsf{val}_r(D)$.*

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

- Assume $prim(D) \not\subseteq prim(C_i)$ for one i. Then there exists an $A \in prim(D) \setminus prim(C_i)$. 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 $A \notin prim(C_i)$. Thus, $d \notin D^I$ and therefore $C \not\sqsubseteq D$, in contradiction to our assumption.

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

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 $\mathsf{val}_r(C_i) \not\sqsubseteq \mathsf{val}_r(D)$ for one i. Thus, $\mathsf{val}_r(C_i)$ has a tree model I_{val} such that $d_{val} \in \mathsf{val}_r(C_i)^{I_{val}} \setminus \mathsf{val}_r(D)^{I_{val}}$. 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 $d \notin D^{I'}$.

($\Leftarrow$) 1. Trivial. 2. Let $i \in \{1, \ldots, n\}$. 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 $A \in prim(C_i)$. The inclusion $prim(D) \subseteq prim(C_i)$ thus implies $x \in A^I$ for every $A \in prim(D)$.
- Consider an arbitrary $D' \in \mathsf{ex}_r(D)$. By assumption, we know that there is an $C' \in \mathsf{ex}_r(C_i)$ with $C' \sqcap \mathsf{val}_r(C_i) \sqsubseteq D'$. Since $x \in (\exists r.C' \sqcap \forall r.\mathsf{val}_r(C_i))^I$, this implies $x \in (\exists r.D')^I$.
- As $\mathsf{val}_r(C_i) \sqsubseteq \mathsf{val}_r(D)$ and $x \in (\mathsf{val}_r(C_i))^I$, it holds that $x \in (\mathsf{val}_r(D))^I$.

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

It was argued in Section 3 that the normalization of concept descriptions is crucial for a successful approach to approximation. In the following Section we will see how these are employed in the formal definition of the approximation.

4.3 Upper approximation

A definition of the upper approximation is already given by Definition 2. In this subsection, a corresponding algorithm is proposed to actually compute the upper approximation of a given $\mathcal{ALC}$-concept description.

Definition 12 *Approximation algorithm of $\mathcal{ALC}$ by $\mathcal{ALE}$.*
Input: $\mathcal{ALC}$-concept description C.
Output: $\mathcal{ALE}$-approximation D of C.
1. $C \equiv \{\bot, \top\}$. Then, $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \bot$ *or* $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \top$ *respectively.*
2. Otherwise, transform C into $\mathcal{ALC}$-normal form and return $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$ *as*

$$\bigsqcap_{A \in \bigcap_i prim(C_i)} A$$
$$\sqcap \bigsqcap_{(C_1', \ldots, C_n') \in \mathsf{ex}_r(C_1) \times \cdots \times \mathsf{ex}_r(C_n)} \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i)) \mid 1 \leq i \leq n\}$$
$$\sqcap \forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}_r(C_i)) \mid 1 \leq i \leq n\}.$$

Obviously, the algorithm always makes sure that the input is transformed into $\mathcal{ALC}$-normal form. By computing $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i))$ instead of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i')$ for every existential restriction in the resulting concept description it is also ensured that value restrictions are propagated to existential restrictions. It should be noted that the argument $C_i' \sqcap \mathsf{val}_r(C_i)$ is not necessarily in $\mathcal{ALC}$-normal form even if C was transformed into normal form before.

To see how the above algorithm works, let us return to the examples discussed in Section 3.

Example 13 Consider the concept description $C_{\mathsf{ex},1} = (\forall r.B \sqcup (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A$ from Example 4. Applying $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ to $C_{\mathsf{ex},1}$ would firstly transform the input into $\mathcal{ALC}$-normal form, yielding $(\forall r.B \sqcap \exists r.A) \sqcup (\exists r.B \sqcap \forall r.A \sqcap \exists r.A)$. According to the definition of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$, we therefore have:

$$\begin{aligned}
\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},1}) &= \mathsf{c\text{-}approx}_{\mathcal{ALE}}((\forall r.B \sqcap \exists r.A) \sqcup (\exists r.B \sqcap \forall r.A \sqcap \exists r.A)) \\
&\equiv \exists r.\mathsf{lcs}\{A \sqcap B, B \sqcap A\} \sqcap \\
&\quad \exists r.\mathsf{lcs}\{A \sqcap B, A \sqcap A\} \sqcap \\
&\quad \forall r.\mathsf{lcs}\{B, A\} \\
&\equiv \exists r.(B \sqcap A) \ \sqcap \ \exists r.A \ \sqcap \forall r.\top \\
&\equiv \exists r.(B \sqcap A)
\end{aligned}$$

The concept description from Example 6, $C_{\mathsf{ex},2} = \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$, is already in $\mathcal{ALC}$-normal form. Thus, applying $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ yields:

$$\begin{aligned}
\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_{\mathsf{ex},2}) &= \mathsf{c\text{-}approx}_{\mathcal{ALE}}(\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)) \\
&\equiv \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(A \sqcap (\neg A \sqcup \neg B))\} \sqcap \\
&\quad \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(B \sqcap (\neg A \sqcup \neg B))\} \sqcap \\
&\quad \forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\neg A \sqcup \neg B)\} \\
&\equiv \exists r.\mathsf{c\text{-}approx}_{\mathcal{ALE}}((A \sqcap \neg A) \sqcup (A \sqcap \neg B)) \sqcap \\
&\quad \exists r.\mathsf{c\text{-}approx}_{\mathcal{ALE}}((B \sqcap \neg A) \sqcup (B \sqcap \neg B)) \sqcap \\
&\quad \forall r.\top \\
&\equiv \exists r.(A \sqcap \neg B) \ \sqcap \ \exists r.(B \sqcap \neg A)
\end{aligned}$$

The above example shows that $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ correctly approximates the example concepts from Section 3. The following theorem proves that the algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ always finds the correct approximation:

Theorem 14 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form. Then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$ is the upper $\mathcal{ALE}$-approximation of C, i.e.,*

1. $C \sqsubseteq \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$, *and*
2. $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \sqsubseteq D$ *for every $\mathcal{ALE}$-concept description D with $C \sqsubseteq D$.*

PROOF. 1. Show $C \sqsubseteq \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$. To this end, show by induction over the structure of C that the conditions for subsumption from Theorem 11 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 $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$, we have $prim(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)) = \bigcap_{i=1}^{n} prim(C_i) \subseteq prim(C)$.
- Show: for $\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i)) \mid 1 \leq i \leq n\}$ and for all i there exists some $C' \in \mathsf{ex}_r(C_i)$ with $C' \sqcap \mathsf{val}_r(C_i) \sqsubseteq \mathsf{lcs}(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i)) \mid 1 \leq i \leq n\}$.

  Pick $C' = C_i'$. By induction hypothesis it holds that $C' \sqcap \mathsf{val}_r(C_i) \sqsubseteq \mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i))$. The definition of the lcs now guarantees $C' \sqcap \mathsf{val}_r(C_i) \sqcap \mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i)) \mid 1 \leq i \leq n\}$.
- Show: $\mathsf{val}_r(C) \sqsubseteq \mathsf{val}_r(\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C))$. By induction hypothesis we already know that $\mathsf{val}_r(C_i) \sqsubseteq \mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}_r(C_i))$ for every i. Consequently, for the lcs we find $\mathsf{val}_r(C_i) \sqsubseteq \mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}_r(C_i)) \mid 1 \leq i \leq n\}$.

2. 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:

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

∎

As the transformation into $\mathcal{ALC}$-normal form preserves equivalence we may extend the above result in the following way:

Corollary 15 *1. The above result also holds for $\mathcal{ALC}$-concept descriptions which are not in $\mathcal{ALC}$-normal form.*

2. The size of $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C)$ *can be exponential in the size of* C*, where* C *is in* $\mathcal{ALC}$*-normal form.*

PROOF. 1. Easy to see 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.

2. Consider two $\mathcal{ALE}$-concept descriptions C_1 and C_2 in $\mathcal{ALE}$-normal form. According to the definition, $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_1 \sqcup C_2) = \mathsf{lcs}(C_1, C_2)$. It has been shown in [1] that there exist pairs of $\mathcal{ALE}$-concept descriptions whose lcs is exponentially large in the size of the input. ∎

Having shown its correctness, the natural next question regards the computational complexity of the algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$. In the following corollary it is shown that $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ can be realized as a double-exponential time algorithm.

Corollary 16 *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 $\prod_{A \in \bigcap_i prim(C_i)} A$ of primitive concepts;
2. an existential restriction $\exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i)) \mid 1 \leq i \leq m\}$ for every tuple $(C_1', \ldots, C_m')$ with $C_i' \in \mathsf{ex}_r(C_i)$;
3. one value restriction $\forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}_r(C_i)) \mid 1 \leq i \leq m\}$.

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 $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C_i' \sqcap \mathsf{val}_r(C_i))$. 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, c-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 c-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 in propagated $\mathcal{ALE}$-normal form (Definition 8), since the concepts returned by c-approx$_{\mathcal{ALE}}$ are in this form. 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 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. ∎

We have evaluated a first prototypical implementation of c-approx$_{\mathcal{ALE}}$ realized in Lisp and using the FaCT system [10] as a subsumption tester. Our implementation of c-approx$_{\mathcal{ALE}}$ utilizes the lcs implementation described in [15]. In contrast to the c-approx$_{\mathcal{ALE}}$ algorithm specified in Definition12 our implementation reduces the number of lcs calls. 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. But 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 c-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 c-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 runtimes as well as for sizes of returned concepts.

5 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 concepts, 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 $C_{\mathsf{ex},2}$ from Example 4 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 [14]. 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, the most general concept E with $E \sqcap D \equiv C$ is always $C \sqcup \neg D$. Consequently, the difference operator proposed by Teege 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 might 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. In [11], a so-called *subdescription ordering* has been proposed to deal with syntactical redundancies. In order to extend this approach to our case the first step is 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 17 *Let C, D be an $\mathcal{ALC}$-concept descriptions in $\mathcal{ALC}$-normal form. Let $C = C_1 \sqcup \cdots \sqcup C_n$. Then, $D \preceq_d C$ iff $D \in \{\bot, \top\}$ 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 $A \in prim(C_i)$,

(b) remove some conjuncts $\exists r.C_i'$ with $C_i' \in \mathsf{ex}_r(C_i)$,

(c) remove the conjunct $\forall r.\mathsf{val}_r(C_i)$,

(d) for every remaining $C_i' \in \mathsf{ex}_r(C_i) \cup \{\mathsf{val}_r(C_i)\}$: replace C_i' by C_i'' with $C_i'' \preceq_d C_i'$.

As an example, consider the 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 removing the last disjunct in the remaining existential restriction we find that $C \prec_d D$. Note that $C \equiv D$.

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

Definition 18 *Let C be $\mathcal{ALC}$-concept descriptions in $\mathcal{ALC}$-normal form and D be $\mathcal{ALE}$-concept descriptions in $\mathcal{ALE}$-normal form. Then, the $\mathcal{ALC}$-concept description E is called the* difference of C and D, *($C - D$ for short), iff*
1. $E \sqcap D \equiv C \sqcap D$

2. For every $\mathcal{ALE}$-concept description E' with $E' \sqcap D \equiv C \sqcap D$ it holds that either $E \preceq_d E'$ or E and E' are incomparable with respect to $\preceq_d$.

Intuitively, the idea is to remove all subdescriptions from C which are either redundant in C 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. $\sqsubseteq$. In general, the difference $C - D$ is not maximal with respect to subsumption, as a simple example illustrates. For $C = A \sqcup B$ and $D = B$, we obtain $C - D = A$, although $C \sqcap (A \sqcup B) \equiv C \sqcap D$, i.e., $A \sqcup B$ is a more general solution w.r.t. subsumption.

Finally, it should be noted that a priori the difference between C and D is not uniquely determined. By abuse of language and notation, we will still refer to *the* difference $C - D$. Coming back to the example from the beginning of this section, the difference (according to Definition 18) between $C_{\mathsf{ex},2}$ and its approximation is in fact the desired value restriction $\forall r.(\neg A \sqcup \neg B)$.

Having defined our difference operator, we need to devise an algorithm to actually compute the difference $C - D$. In [11], 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 c-diff as depicted in Figure 1.

If C is a disjunction of subconcepts C_i then the difference between C and D is computed by firstly computing the differences between the disjuncts and D and then eliminating the semantically redundant resulting disjuncts. In $n = 1$, C is a conjunction of $\mathcal{ALC}$-concept descriptions (with possibly just one conjunct). In this case, redundant concept names and existential restrictions on the top-level conjunction of C are removed. Furthermore,

Input: $\mathcal{ALC}$-concept description $C = C_1 \sqcup \cdots \sqcup C_n$ in $\mathcal{ALC}$-normal form,
$\mathcal{ALE}$-concept description D
Output: $\mathsf{c\text{-}diff}(C, D)$

1. If $C \sqcap D \equiv \bot$, then $\mathsf{c\text{-}diff}(C, D) := \bot$;
2. 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 $\mathsf{c\text{-}diff}(C_j, D) \sqsubseteq \bigsqcup_{i \neq j} \mathsf{c\text{-}diff}(C_i, D)$;
3. If $n = 1$, then $\mathsf{c\text{-}diff}(C, D) :=$

$$\prod_{A \in prim(C) \setminus prim(D)} A \;\sqcap\; \forall r.\mathsf{c\text{-}diff}(\mathsf{val}_r(C), \mathsf{val}_r(D)) \;\sqcap \prod_{E \in \mathcal{E}'_r} \exists r.E$$

where the value restriction is omitted in case $\mathsf{c\text{-}diff}(\mathsf{val}_r(C), \mathsf{val}_r(D)) \equiv \top$ and $\mathcal{E}'_r$ is computed as follows:

Let $\mathcal{E}_r = \{C'_1, \ldots, C'_n\} := \mathsf{ex}_r(C)$.
For $i = 1$ to n do begin
If (i) there exists $C' \in \mathcal{E}_r \setminus \{C'_i\}$ with $\mathsf{val}_r(D) \sqcap \mathsf{val}_r(C) \sqcap C' \sqsubseteq C'_i$, or
(ii) there exists $D' \in \mathsf{ex}_r(D)$ with $\mathsf{val}_r(D) \sqcap \mathsf{val}_r(C) \sqcap D' \sqsubseteq C'_i$
then $\mathcal{E}_r := \mathcal{E}_r \setminus \{C'_i\}$
end
$\mathcal{E}'_r := \{E^* \mid E \in \mathcal{E}_r\}$ where $E^* := \mathsf{c\text{-}diff}(E, \mathsf{val}_r(C) \sqcap \mathsf{val}_r(D))$, if $\mathsf{val}_r(C)$ is an $\mathcal{ALE}$-concept description, and $E^* := \mathsf{c\text{-}diff}(E, \mathsf{val}_r(D))$ otherwise.

Figure 1: The algorithm $\mathsf{c\text{-}diff}(C, D)$.

redundancies in existential restrictions and value restrictions are removed recursively. The set $\mathcal{E}'_r$ can be computed by iteratively removing existential restrictions of C that do not satisfy Conditions 3(i) or 3(ii). Given an oracle for subsumption, this can be carried out in polynomial time.

The following lemma proves that $\mathsf{c\text{-}diff}(C, D)$ respects the first condition of the difference operator (Definition 18), i.e., it does not remove too much from the original concept description C.

Lemma 19 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form and D an $\mathcal{ALE}$-concept description in $\mathcal{ALE}$-normal form. Then,* $\mathsf{c\text{-}diff}(C, D) \sqcap D \equiv C \sqcap D$.

Proof. Proof by induction over the structure of C.

1. $C \in prim(C)$
As $prim(\mathsf{c\text{-}diff}(C, D)) = prim(C) \setminus prim(D)$, it follows that $\mathsf{c\text{-}diff}(C, D) \sqcap D$ is equivalent to $D \sqcap \prod_{A \in prim(\mathsf{c\text{-}diff}(C,D))} A$. We can safely add to this another conjunct more

general than D yielding

$$D \sqcap \bigsqcap_{A \in prim(\mathsf{c\text{-}diff}(C,D))} A \sqcap \bigsqcap_{A \in prim(C) \cap prim(D)} A.$$

The expression thus obtained is equivalent to $C \sqcap D$.

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

3. No disjunction on the top-level of C
Show $\mathsf{c\text{-}diff}(C, D) \sqcap D \equiv C \sqcap D$. According to the characterization of subsumption (Theorem 11), three conditions must hold for equivalence:

   - The set of primitive concepts $prim(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ equals $prim(\mathsf{c\text{-}diff}(C, D)) \cup prim(D)$ which by definition is $(prim(C) \setminus prim(D)) \cup prim(D)$. This is equal to $prim(C) \cup prim(D)$, the set of primitive concepts in $C \sqcap D$.
   - By induction hypothesis, $\mathsf{c\text{-}diff}(\mathsf{val}_r(C), \mathsf{val}_r(D)) \sqcap \mathsf{val}_r(D)$ is equivalent to $\mathsf{val}_r(C) \sqcap \mathsf{val}_r(D)$. By definition $\mathsf{val}_r(C \sqcap D)$ is equivalent to $\mathsf{val}_r(C) \sqcap \mathsf{val}_r(D)$ which concludes this case.
   - Show ($\sqsubseteq$). Let $F' \in \mathsf{ex}_r(C \sqcap D)$. We have to find an $E' \in \mathsf{ex}_r(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ with $E' \sqcap \mathsf{val}_r(\mathsf{c\text{-}diff}(C, D) \sqcap D) \sqsubseteq F'$. From the previous case we know that $\mathsf{val}_r(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ is equivalent to $\mathsf{val}_r(C \sqcap D)$. Since $\mathsf{ex}_r(C \sqcap D)$ is equal to the union $\mathsf{ex}_r(\mathsf{c\text{-}diff}(C, D)) \cup \mathsf{ex}_r(D)$ we may distinguish two cases.

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

     If $F' \in \mathsf{ex}_r(C) \setminus \mathsf{ex}_r(D)$, then Conditions 3(i) and 3(ii) in the definition of the algorithm $\mathsf{c\text{-}diff}(C, D)$ guarantee that there exists an existential restriction $\tilde{E}' \in \mathsf{ex}_r(\mathsf{c\text{-}diff}(C, D))$ with the following properties. If $\mathsf{val}_r(C)$ is an $\mathcal{ALE}$-concept description then $\tilde{E}'$ is of the form $\mathsf{c\text{-}diff}(E', (\mathsf{val}_r(D) \sqcap \mathsf{val}_r(C)))$ with $E' \sqcap \mathsf{val}_r(D) \sqcap \mathsf{val}_r(C) \sqsubseteq F'$. According to the induction hypothesis, $\mathsf{c\text{-}diff}(E', (\mathsf{val}_r(D) \sqcap \mathsf{val}_r(C))) \sqcap \mathsf{val}_r(D) \sqcap \mathsf{val}_r(C)$ is equivalent to $E' \sqcap \mathsf{val}_r(D) \sqcap \mathsf{val}_r(C)$. Consequently, we find that $\tilde{E}' \sqcap \mathsf{val}_r(C) \sqcap \mathsf{val}_r(D) \sqsubseteq F'$. It is easy to see that $\mathsf{val}_r(C) \sqcap \mathsf{val}_r(D)$ is equivalent to $\mathsf{val}_r(C \sqcap D)$ which again is equivalent to $\mathsf{val}_r(\mathsf{c\text{-}diff}(C, D) \sqcap D)$ as we know from above. Hence, we have found an $\tilde{E}'$ with $\tilde{E}' \sqcap \mathsf{val}_r(\mathsf{c\text{-}diff}(C, D) \sqcap D) \sqsubseteq F'$. If D is no $\mathcal{ALE}$-concept description then $\tilde{E}'$ is of the form $E' \sqcap \mathsf{val}_r(D)$. This case is analogous to the previous one.

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

If $E' \in \mathsf{ex}_r(D)$, then we can again select $F' := E'$ which also occurs in $\mathsf{ex}_r(C \sqcap D)$.

If $E' \in \mathsf{ex}_r(\mathsf{c\text{-}diff}(C,D)) \setminus \mathsf{ex}_r(D)$, then Condition 3(ii) guarantees that an $F' \in \mathsf{ex}_r(D) \subseteq \mathsf{ex}_r(C \sqcap D)$ exists such that $F' \sqcap \mathsf{val}_r(C) \sqcap \mathsf{val}_r(D) \sqsubseteq E'$. As seen above, $\mathsf{val}_r(C) \sqcap \mathsf{val}_r(D)$ is equivalent to $\mathsf{val}_r(C \sqcap D)$ which concludes the argument.

∎

We still have to examine the computational complexity of the algorithm c-diff. In the following corollary it is shown that c-diff is a polynomial time algorithm.

Corollary 20 *Given an oracle for subsumption, the algorithm* c-diff *is a polynomial time algorithm, i.e., for a given $\mathcal{ALC}$-concept description C in $\mathcal{ALC}$-normal form and an $\mathcal{ALE}$-concept description D, the computation of* $\mathsf{c\text{-}diff}(C,D)$ *takes at most polynomial time in the size of C and D.*

Proof. It is not difficult to see that the size of the output $\mathsf{c\text{-}diff}(C,D)$ never exceeds the size of C: if $n > 1$ then the difference is simply distributed to the disjuncts, and if $n = 1$ then, (1) some primitive concepts are removed, thus reducing the size of the resulting concept description, (2) the value restriction is handled recursively and (3) some existential restrictions are removed while the remaining ones are also handled recursively. Consequently, during the recursive computation of $\mathsf{c\text{-}diff}(C,D)$ the algorithm is never applied to an argument exceeding the size of the input. Neither does the algorithm introduce new existential or value restrictions during the computation of $\mathsf{c\text{-}diff}(C,D)$.

Thus, it is sufficient for our claim to show that (1) the computation of the subset $\mathcal{E}'_r$ takes only polynomial time in the size of the input and (2) there are at most polynomially many (in the size of C and D) calls to c-diff during the recursive computation of $\mathsf{c\text{-}diff}(C,D)$.

1. As the condition in Step 3 states an appropriate subset $\mathcal{E}'_r$ can be found by iteratively removing elements from the original set $\mathsf{ex}_r(C)$ and verifying Conditions 3(i) and 3(ii) in every iteration. Thus, the number of subsets to inspect is bounded by the size of C. For every subset, a polynomial number of subsumption test must be made. Given an oracle for subsumption, this task costs only polynomial time.

2. Recursive calls to c-diff are necessary for the computation of $\mathsf{c\text{-}diff}(\mathsf{val}_r(C), \mathsf{val}_r(D))$ as well as for the computation of every E^*_j. Nevertheless, there is only one value restriction $\mathsf{val}_r(C)$ in C the size of which is bounded by the size of C. As no new value restrictions are introduced, we have at most polynomially many expressions of the form $\mathsf{c\text{-}diff}(\mathsf{val}_r(C), \mathsf{val}_r(D))$ to evaluate during the execution of $\mathsf{c\text{-}diff}(C,D)$.

   As c-diff does not introduce new existential restrictions and as the size of its output never exceeds the size of its input it is easy to see that the number of existential restrictions E'_j and their size is bounded by the input. Consequently, the number of

calls to c-diff is bounded by the syntax tree of the input concept C which again is bounded by the size of C, since C was assumed in $\mathcal{ALC}$-normal form.

∎

It should be recalled though that transforming an arbitrary $\mathcal{ALC}$-concept description into $\mathcal{ALC}$-normal form can produce an exponentially larger concept description. To summarize the existing results, the following properties can be shown for every computation of the algorithm c-diff(C, D).

Theorem 21 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form and D be an $\mathcal{ALE}$-concept description. Then,* c-diff(C, D) *satisfies the following properties:*
1. c-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* c-diff(C, D) *coincide modulo associativity and commutativity, and*
3. *given an oracle for subsumption, the computation of* c-diff(C, D) *takes polynomial time in the size of C and D.*

The first property where only $\mathcal{ALE}$ is considered was already shown in [11]. The others have been shown in the above lemma and corollary.

We have implemented a prototype for the c-diff algorithm in Lisp. For a first evaluation we applied the c-diff implementation to test concepts derived from our process engineering TBox. More precisely, we applied c-diff to the same $\mathcal{ALC}$-concept descriptions used for the evaluation of c-approx$_{\mathcal{ALE}}$ together with their approximations generated by our c-approx$_{\mathcal{ALE}}$ implementation. For these test cases the c-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 c-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

The present paper has investigated a new inference problem for DLs, namely computing the approximation of concepts from one DL in another DL. For the concrete case of approximating $\mathcal{ALC}$-concepts in $\mathcal{ALE}$ the seemingly simple task of eliminating disjunctions in concepts may fail without the computation of normal forms and the propagation of value

restrictions to existential restrictions. As a main result, we have devised a correct and effective algorithm to compute upper approximations of $\mathcal{ALC}$-concepts in $\mathcal{ALE}$.

In order to ascertain the accuracy of the approximation, we have proposed a difference operator and a corresponding algorithm which effectively computes a compact representation of the subconcepts not present in the approximation. The algorithm is correct in the sense that does not overlook subconcepts missing in the approximation though it does not always return a (syntactically) minimal concept.

Our first evaluation of the prototype implementations of c-approx$_{\mathcal{ALE}}$ and c-diff indicates that the implementations behave fairly good on test cases derived from our practical application. On the other hand there is clearly a need for further optimization to employ these new non-standard inferences efficiently in practical applications. Even more important, since the concept descriptions returned by both algorithms can grow quite big and are therefore 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 $\mathcal{ALC}$.

As an algorithm for the lcs of $\mathcal{ALEN}$-concepts exists [12], a future step is to extend the present approximation technique to $\mathcal{ALCN}$, thus producing $\mathcal{ALEN}$-approximations. Furthermore, it is desirable to investigate further if optimal solutions for the difference of $\mathcal{ALC}$-concepts always exist and can be computed effectively.

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 *Lec-*

ture 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] 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.

[8] 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.

[9] V. Haarslev and R. Möller. Description of the RACER system and its applications. In *Proceedings of the International Workshop in Description Logics 2001 (DL2001)*, Stanford, USA, August 2001.

[10] I. R. 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.

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

[12] R. Küsters and R. Molitor. Computing Least Common Subsumers in ALEN. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 219–224. Morgan Kaufman, 2001.

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

[14] 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.

[15] 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.