An Approach for Optimized Approximation

Sebastian Brandt and Anni-Yasmin Turhan*

Theoretical Computer Science, TU Dresden, Germany

Email: {brandt, turhan}@tcs.inf.tu-dresden.de

Abstract

Approximation is a new inference service investigated in [4]. An approximation of an $\mathcal{ALC}$-concept by an $\mathcal{ALE}$-concept can be computed in double exponential time. Consequently, one needs powerful optimization techniques for approximating an entire unfoldable TBox. Addressing this issue we identify a special form of $\mathcal{ALC}$-concepts that can be divided into parts s.t. each part can be approximated independently.

1 Motivation

This paper presents preliminary results on optimization techniques for the computation of approximations. Approximation is a new non-standard inference service in Description Logics (DLs) introduced in [4]. Approximating a concept, defined in one DL, means to translate this concept to another concept, defined in a second, typically less expressive DL, such that both concepts are as closely related as possible with respect to subsumption. Like other non-standard inferences such as the least common subsumer (lcs) or matching, approximation has been introduced to support the construction and maintenance of DL knowledge-bases (see [9, 5]). Approximation has a number of different applications some of which we will mention here, see [4] for others.

Computation of commonalities of concepts. Given a set of concepts the problem is to extract the commonalities of the input concepts. Typically, the lcs is employed for this task. In case a DL $\mathcal{L}$ provides 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 commonalities between C_1 and C_2. By using approximation, however, one can make the commonalities explicit to some extent by first approximating C_1 and C_2 in a sublanguage of $\mathcal{L}$ which does not provide disjunction, and then computing the lcs of the approximations in $\mathcal{L}$.

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

Translation of knowledge-bases. Approximation can be used to (automatically) translate a knowledge-base written in an expressive DL into 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.

We investigate the case of translating an $\mathcal{ALC}$-TBox into an $\mathcal{ALE}$-TBox by computing the approximation of each concept defined in the $\mathcal{ALC}$-TBox. In [4], a first in-depth investigation of the approximation inference has been presented. In particular, a double-exponential time algorithm has been devised to approximate $\mathcal{ALC}$-concepts by $\mathcal{ALE}$-concepts. Consequently, approximating an entire TBox requires substantial optimizations. We address this problem by identifying a form of $\mathcal{ALC}$-concept descriptions whose conjuncts can be approximated independently. This approach speeds-up the computation of a single approximation. Moreover, it also allows to re-use an obtained approximation in subsequent approximations by simply inserting the approximation of a subconcept in the current approximation. Therefore the splitting of concepts in independent parts is a prerequisite for applying caching techniques to approximation. The full proofs of the results presented here can be found in our technical report [6].

2 Preliminaries

Concept descriptions are inductively defined based on a set of concept *constructors* starting with a set N_C of *concept names* and a set N_R of *role names*. In this paper, we consider concept descriptions built from the constructors shown in Table 1 where C and D denote arbitrary concepts, A a concept name, and r a role. Note that in $\mathcal{ALC}$ every concept description can be negated whereas in $\mathcal{ALE}$ negation is only allowed in front of concept names. In the following a concept description formed with the constructors allowed in a DL $\mathcal{L}$ is called *$\mathcal{L}$-concept description*.

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 Table 1.

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 $\forall r.E \sqcap \forall r.F \equiv \forall r.(E \sqcap F)$).

A *TBox* is a finite set of concept definitions of the form $A \doteq C$, where $A \in N_C$ and C is a concept description. In addition, we require that TBoxes

Syntax	Semantics	$\mathcal{ALE}$	$\mathcal{ALC}$
$\top$	Δ	x	x
$\bot$	$\emptyset$	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
$\neg C$	$\Delta \setminus C^I$		x
$C \sqcup D$	$C^I \cup D^I$		x

Table 1: Syntax and semantics of concept descriptions.

are *unfoldable*, i.e., they are acyclic and do not contain multiple definitions (see, e.g., [10]). Concept names occurring on the left-hand side of a definition are called *defined concepts*. All other concept names are called *primitive concepts*. In TBoxes of the DL $\mathcal{ALE}$, negation may only be applied to primitive concepts. An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies all its concept definitions, i.e., $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all definitions $A \doteq C$ in $\mathcal{T}$.

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 $D \sqsubseteq C$. Subsumption and equivalence in $\mathcal{ALC}$ is PSPACE-complete [11] and NP-complete in $\mathcal{ALE}$ [7].

2.1 $\mathcal{ALE}$-Approximation for $\mathcal{ALC}$

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

Definition 1 *Given $\mathcal{L}$-concept descriptions $C_1, \ldots, C_n$ with $n \geq 2$ 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 $C \sqsubseteq C'$.*

As already mentioned, in $\mathcal{ALC}$ the lcs trivially exists since $lcs(C, D) \equiv C \sqcup D$. For $\mathcal{ALE}$ the existence is not obvious. It was shown in [2] that the lcs of two or more $\mathcal{ALE}$-concept descriptions always exists, that its size may grow exponentially in the size of the input descriptions, and that it can be computed in exponential time.

Intuitively, to approximate an $\mathcal{ALC}$-concept description from "above" means to compute an $\mathcal{ALE}$-concept description that is more general than the input concept description but minimal w.r.t. subsumption.

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 an* upper $\mathcal{L}_2$-approximation *of C ($D = approx_{\mathcal{L}_2}(C)$ for short) iff (i) $C \sqsubseteq D$, and (ii) D is minimal with this property, i.e., $C \sqsubseteq D'$ and $D' \sqsubseteq D$ implies $D' \equiv D$ for all $\mathcal{L}_2$-concept descriptions D'.*

Although defined in [4] lower approximations are not yet further investigated. In this paper, we restrict our investigations to upper $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions. Therefore, whenever we speak of approximations, we mean upper $\mathcal{ALE}$-approximations. Thus, having defined approximation we turn now to how to actually compute them.

2.2 The Approximation Algorithm

Before a defined concept from a TBox can be approximated it has to be *unfolded* w.r.t. the underlying TBox to make the information captured in the concept definitions explicit. To this end, every defined concept is replaced by the concept description on the right-hand side of its concept definition until no defined concept occurs in the concept description. It is well known that this process can cause an exponential blow-up of the concept description, see [10]. To recapitulate the approximation algorithm presented in [4], we need to introduce the $\mathcal{ALC}$-normal form.

For an unfolded concept description C the *role-depth* $rd(C)$ is inductively defined as follows:

$$\begin{aligned} rd(N) &:= 0 && \text{, where } N \in N_C \cup \{\bot, \top\} \\ rd(\neg C) &:= rd(C) && \\ rd(C_1\ \rho\ C_2) &:= \max\{rd(C_1), rd(C_2)\} && \text{, where } \rho \in \{\sqcap, \sqcup\} \\ rd(Qr.C) &:= 1 + rd(C) && \text{, where } Q \in \{\exists, \forall\} \end{aligned}$$

A *role-level* of a concept C is the set of all concept descriptions occurring on the same role-depth in C. The topmost role-level of a concept description is called its *top-level*.

We call a concept description *top-level $\sqcup$-free* if it is in negation normal form (NNF), i.e., negation is pushed inwards until in front of a concept name, and does not contain any disjunction on top-level. Some notation is needed to access the different parts of an $\mathcal{ALE}$-concept description or a top-level $\sqcup$-free $\mathcal{ALC}$-concept description C:

- $\mathsf{prim}(C)$ denotes the set of all (negated) concept names and the bottom concept occurring on the top-level of C;
- $\mathsf{val_r}(C) := C_1 \sqcap \cdots \sqcap C_n$, if there exist value restrictions of the form $\forall r.C_1, \ldots, \forall r.C_n$ on the top-level of C; otherwise, $\mathsf{val_r}(C) := \top$;

Input: unfolded $\mathcal{ALC}$-concept description C
Output: $\mathcal{ALE}$-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 $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) :=$

$$\bigsqcap_{A \in \bigcap_{i=1}^{n} \mathsf{prim}(C_i)} A \quad \sqcap \quad \forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val_r}(C_i)) \mid 1 \le i \le n\} \ \sqcap$$
$$\bigsqcap_{(C'_1,\ldots,C'_n) \in \mathsf{ex_r}(C_1) \times \cdots \times \mathsf{ex_r}(C_n)} \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val_r}(C_i)) \mid 1 \le i \le n\}$$

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

- $\mathsf{ex_r}(C) := \{C' \mid$ there exists $\exists r.C'$ on the top-level of $C\}$.

Equipped with these we can define the $\mathcal{ALC}$-normal form in which conjuncts are distributed over the disjuncts. An arbitrary $\mathcal{ALC}$-concept description is transformed into a concept description with at most one disjunction on top-level of every concept of each role-level.

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*

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

   $C_i \not\equiv \bot$, and $\mathsf{val_r}(C_i)$ and every concept description in $\mathsf{ex_r}(C_i)$ 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. Every disjunct of a concept in $\mathcal{ALC}$-normal form is top-level $\sqcup$-free. Unfortunately, the normalization may take exponential time. For instance, the normal form of $(A_1 \sqcup A_2) \sqcap \cdots \sqcap (A_{2n-1} \sqcup A_{2n})$ is of size exponential in n.

The approximation algorithm displayed in Figure 1 checks if the input is a concept equivalent to $\top$ or $\bot$—in this case the approximation is trivial—otherwise it proceeds recursively on the $\mathcal{ALC}$-normal form of the input and extracts the commonalities of all disjuncts. Unfortunately, the algorithm needs double-exponential time for arbitrary $\mathcal{ALC}$-concepts in the worst case. Despite its high complexity, our prototypical implementation of the algorithm showed a quite promising performance in respect to run-time and resulting concept sizes, for details see [4].

3 Optimizing $\mathcal{ALE}$-Approximations

A TBox can be translated by computing the approximation of the concept description on the right-hand side of every concept definition in the TBox. Each defined concept has to be unfolded and transformed into $\mathcal{ALC}$-normal form before the approximation algorithm can be applied. Unfortunately, both of these steps cause an exponential growth of the concept description.

For standard reasoning tasks [1, 8] and also for the computation of the lcs [3] the first source of complexity can often be alleviated by *lazy unfolding*. Here the idea is to replace a defined concept in a concept description only if examination of that part of the description is necessary. Lazy unfolding unfolds all defined concepts appearing on the top-level of the concept description under consideration while defined concepts on deeper role-levels remain unchanged as long as possible.

When computing the lcs the main benefit of lazy unfolding is that in some cases defined concepts can be used directly in the lcs concept description. If, for example a defined concept C appears in all input concept descriptions on the same role-level, the concept definition of C does not need to be processed, but C can be inserted into the lcs directly, see [3] for details. In the case of approximation, however, this effect of lazy unfolding can not be utilized even if a defined concept is obviously common to all disjuncts. For example, in $(A \sqcap C) \sqcup (C \sqcap (\neg B))$ the concept name C cannot be used directly as a name in the approximation because the $\mathcal{ALC}$-concept description C stands for must be approximated. Thus unfolding a concept completely cannot be avoided for approximation.

The double-exponential time complexity of the approximation algorithm, however, suggests another approach to optimization. Instead of approximating an input concept C as a whole a significant amount of time could be saved by splitting C into its conjuncts and approximating them separately. If, for instance, C consists of two conjuncts of size n then the approximation of C takes some $a^{b^{2n}}$ steps while the conjunct-wise approach would just take $2a^{b^n}$. Unfortunately, splitting an arbitrary input concept at conjunctions leads to incorrect approximations, as examples show [4]. In the following section we will therefore introduce a class of so-called nice $\mathcal{ALC}$-concepts for which the conjunct-wise approximation still produces the correct result.

3.1 Nice Concepts

In the following we assume that all concept descriptions are unfolded and in NNF. For an $\mathcal{ALC}$-concept description C and $i \in \mathbb{N}$ the *quantor set* $Q_r(C, i)$ denotes the set of quantors used on the role-level i of C (referring to role r). Hence, for $0 \leq i \leq rd(C)$ the quantor set $Q_r(C, i)$ is a nonempty subset of $\{\forall, \exists\}$. Similarly, the *name set* $N_r(C, i)$ denotes the atomic concepts used on a specific role-level. Formally, Q and N are defined as follows.

Definition 4 *Let $C := \sqcup_{i=1}^{k} C_i$ be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form. For $d \in \mathbb{N}$, the sets $Q_r(C, d)$ and $N_r(C, d)$ are inductively defined by:*

- $Q_r(C, 0) := \{\exists \mid \bigcup_{i=1}^{k} \mathsf{ex_r}(C_i) \neq \emptyset\} \cup \{\forall \mid \sqcap_{i=1}^{k} \mathsf{val_r}(C_i) \not\equiv \top\}$

  $N_r(C, 0) := \bigcup_{i=1}^{k} \mathsf{prim}(C_i)$

- $Q_r(C, d+1) := \bigcup_{i=1}^{k} \bigcup_{C' \in \mathsf{ex_r}(C_i)} Q_r(C', d) \cup \bigcup_{i=1}^{k} Q_r(\mathsf{val_r}(C_i), d)$

  $N_r(C, d+1) := \bigcup_{i=1}^{k} \bigcup_{C' \in \mathsf{ex_r}(C_i)} N_r(C', d) \cup \bigcup_{i=1}^{k} N_r(\mathsf{val_r}(C_i), d).$

For a concept description C not in $\mathcal{ALC}$-normal form, Q and N are defined in terms of the $\mathcal{ALC}$-normal form of C. For example the unfolded concept $C = (\exists r.(A \sqcap B) \sqcap \forall r.(D \sqcup (\exists r.\neg E)))$ has the quantor sets $Q_r(C, 0) = \{\forall, \exists\}$, $Q_r(C, 1) = \{\exists\}$ and $Q_r(C, i) = \emptyset$ for $i \geq 2$. For the name sets, we have $N(C, 0) = \emptyset$, $N(C, 1) = \{A, B, D\}$, and $N(C, 2) = \{\neg E\}$.

We are now ready to specify in detail what nice concepts are. In general, an approximation $approx_{\mathcal{ALE}}(C \sqcap D)$ cannot be split at the conjunction because of possible interactions between existential and value restrictions on the one hand and inconsistencies induced by negation on the other. For example, the approximation $approx_{\mathcal{ALE}}(\exists r.\top \sqcap (\forall r.A \sqcup \exists r.A))$ yields $\exists r.A$ while the split version $approx_{\mathcal{ALE}}(\exists r.\top) \sqcap approx_{\mathcal{ALE}}(\forall r.A \sqcup \exists r.A)$ only produces $\exists r.\top$. Similarly, the conjunction $A \sqcap (\neg A \sqcup B)$ cannot be approximated separately.

We now call those concepts *nice* for which this simplified strategy still produces the correct result and for which a simple syntactic discrimination rule exists. Firstly, the role quantors occurring in nice concepts are restricted to one type per role-level. Hence, on every role-level of a nice concept either no $\forall$-restrictions or no $\exists$-restrictions occur. Secondly, a concept name and its negation may not occur on the same role-level. Consider Figure 2 for an illustration of these rules. Formally, we can define nice concepts by means of the syntactical operators from Definition 4.

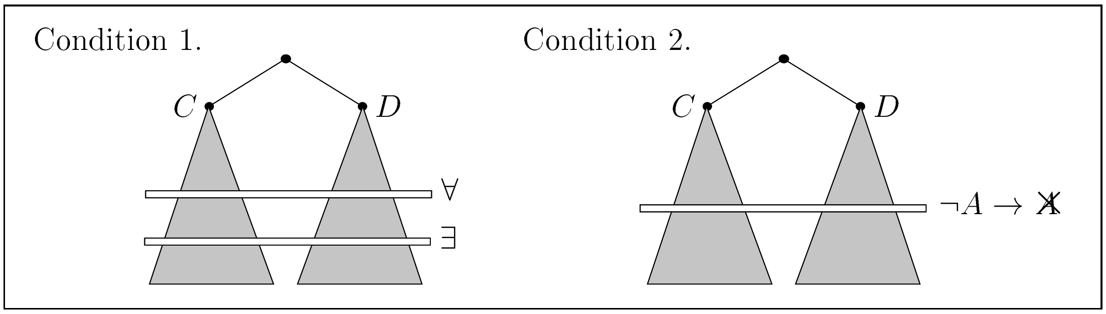

Figure 2: What nice concepts look like

Definition 5 *Let C be an $\mathcal{ALC}$-concept description in NNF. Then C is* nice *iff for every $d \in \mathbb{N}$ it holds that*

1. $|Q_r(C,d)| \leq 1$ *and*
2. $N_r(C,d)$ *does not contain a concept name and its negation.*

It remains to be shown that nice concepts as defined above in fact have the desired property. In preparation for this we firstly present a simple set-theoretic result which later on will allow us to reduce the number of existential restrictions computed in an approximation of certain nice concepts.

The distribution of a conjunction over a disjunction in the $\mathcal{ALC}$-normalization produces conjunctions of a very regular structure. As an example, consider the concept $E := (C_1 \sqcup C_2) \sqcap (D_1 \sqcup D_2)$ with $C_i := \exists r.C_i'$ and $D_j := \exists r.D_j'$. Assuming that all existential restrictions are $\mathcal{ALE}$-concepts, the normalization returns $\bigsqcup_{i,j}(C_i \sqcap D_j)$. The approximation algorithm then computes the lcs over every combination of existential restrictions from the four disjuncts. Nevertheless, every existential restriction in the result $approx_{\mathcal{ALE}}(E)$ either subsumes $\exists r.\mathsf{lcs}\{C_1', C_2'\}$ or $\exists r.\mathsf{lcs}\{D_1', D_2'\}$ because it corresponds to the lcs of a superset of one of the above sets. The following lemma shows that this subset-superset property can be generalized.

Lemma 6 *Let $m, n \in \mathbb{N}$. For every $i \in \{1,\ldots,m\}$ and $j \in \{1,\ldots,n\}$, let A_i and B_j be arbitrary finite sets, let $U_{ij} := A_i \cup B_j$, and let $u_{ij} \in U_{ij}$. Denote by U the set of all u_{ij}, i.e., $U := \{u_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$. Then one of the following claims holds: either, for every i there exist elements $a_i \in A_i$ with $\{a_i \mid 1 \leq i \leq m\} \subseteq U$; or, for every j there exist $b_j \in B_j$ with $\{b_j \mid 1 \leq j \leq n\} \subseteq U$.*

For all $j \in \{1,\ldots,m\}$ and all $j \in \{1,\ldots,n\}$ consider arbitrary $u_{ij} \in U_{ij}$. Assume that the second claim for the sets $B_1,\ldots,B_n$ does not hold. Then there is one j' with $B_{j'} \cap U = \emptyset$, otherwise $b_{j'}$ could be chosen from this intersection to satisfy the claim. Since $u_{ij'} \in A_i \cup B_{j'}$ for all i it follows that $u_{ij'} \in A_i$ for all i, satisfying the first claim for $A_1,\ldots,A_m$. ∎

The choice of sets in the unions U_{ij} in the above lemma corresponds to tuples in the product $\{A_1, \ldots, A_m\} \times \{B_1, \ldots, B_n\}$. The claim can be generalized to n-ary products where every union corresponds to a tuple from $\{S_{11}, \ldots, S_{1k_1}\} \times \cdots \times \{S_{n1}, \ldots, S_{nk_n}\}$. A proof of this generalized version can be found in the technical report [6]. In the following lemma the above result is applied to the actual computation of the lcs.

Lemma 7 *For $1 \leq i \leq 2$, let C_i and D_i be $\mathcal{ALE}$-concept descriptions such that $C_1 \sqcap C_2 \sqcap D_1 \sqcap D_2$ is a nice concept. Then it holds that $\mathsf{lcs}\{C_i \sqcap D_j \mid i, j \in \{1, 2\}\} \equiv \mathsf{lcs}\{C_1, C_2\} \sqcap \mathsf{lcs}\{D_1, D_2\}$.*

The above claim can again be generalized to larger conjunctions. Let $1 \leq i \leq n$ and $1 \leq j \leq k_i$ and let C_{ij} be $\mathcal{ALE}$-concepts whose overall conjunction is nice. For every tuple $\bar{t} \in \{1, \ldots, k_1\} \times \cdots \times \{1, \ldots, k_n\} =: T$ denote by $C_{\bar{t}}$ the conjunction $\bigsqcap_{i=1}^{n} C_{i\bar{t}(i)}$. Then the least common subsumer $\mathsf{lcs}\{C_{\bar{t}} \mid \bar{t} \in T\}$ is equivalent to the conjunction $\bigsqcap_{i=1}^{n} \mathsf{lcs}\{C_{ij} \mid 1 \leq j \leq k_i\}$. The proof is analogous to the one shown above.

We are now ready to show that approximating nice concepts, as defined in Definition 5, can be simplified to a conjunction of approximations. For the sake of simplicity we restrict our attention to binary conjunctions. The proof for n-ary conjunctions is analogous.

Theorem 8 *Let $C \sqcap D$ be a nice $\mathcal{ALC}$-concept description. Then $approx_{\mathcal{ALE}}(C \sqcap D) \equiv approx_{\mathcal{ALE}}(C) \sqcap approx_{\mathcal{ALE}}(D)$.*

For the full proof refer to [6]. The claim is proved by induction over the sum of the nesting depths of $\sqcap$ and $\sqcup$ on every role-level in C and D. For the induction step a case distinction is made depending on whether C or D are conjunctions or disjunctions. If at least one concept description is a disjunction the approximation is defined as the lcs of all $\mathcal{ALC}$-normalized and approximated disjuncts (if one of the concepts is a conjunction, it firstly has to be distributed over the disjunction). The main idea then is to use Lemma 7 to transform single lcs calls of a certain form into a conjunction of lcs calls which eventually leads to the conjunction of the approximations of C and D.

Due to Theorem 8 it is now possible to split the computation of approximations into independent parts. Although this does of course not change the complexity class of the approximation algorithm it is still a significant benefit for practical applications. The improved approximation algorithm is displayed in Figure 3. The algorithm requires the unfolded input concept to be in NNF. In the first step the $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ function checks whether the approximation is trivial. If it is not the next step is to check whether the concept is nice. For nice concepts the $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}$ function is invoked. For all other concepts the $\mathcal{ALC}$-normal form is computed lazily, i.e., the conjunctions are distributed over

Input: unfolded $\mathcal{ALC}$-concept description C already in NNF
Output: upper $\mathcal{ALE}$-approximation of C

$\mathsf{c\text{-}approx}_{\mathcal{ALE}}$

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. If $\mathsf{nice\text{-}concept\text{-}p}(C)$ then return $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(C)$
3. Otherwise, transform the top-level of C into $\mathcal{ALC}$-normal form $C_1 \sqcup \cdots \sqcup C_n$ and return
   $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) :=$

$$\prod_{A \in \bigcap_{i=1}^{n} \mathsf{prim}(C_i)} A \quad \sqcap \quad \forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val_r}(C_i)) \mid 1 \leq i \leq n\} \ \sqcap$$
$$\prod_{(C'_1,\ldots,C'_n) \in \mathsf{ex_r}(C_1) \times \cdots \times \mathsf{ex_r}(C_n)} \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val_r}(C_i)) \mid 1 \leq i \leq n\}$$

$\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}$

1. If $C \equiv \bot$, then $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(C) := \bot$;
   if $C \equiv \top$, then $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(C) := \top$
2. If $C = C_1 \sqcap \cdots \sqcap C_n$, then return
   $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(C) := \prod_{i=1}^{n} \mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(C_i)$
3. Otherwise, return
   $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(C) :=$

$$\prod_{A \in \bigcap_{i=1}^{n} \mathsf{prim}(C_i)} A \quad \sqcap \quad \forall r.\mathsf{lcs}\{\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(\mathsf{val_r}(C_i)) \mid 1 \leq i \leq n\} \ \sqcap$$
$$\prod_{(C'_1,\ldots,C'_n) \in \mathsf{ex_r}(C_1) \times \cdots \times \mathsf{ex_r}(C_n)} \exists r.\mathsf{lcs}\{\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val_r}(C_i)) \mid 1 \leq i \leq n\}$$

Figure 3: The improved algorithm $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ and $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}$.

the disjunctions only for the current top-level. Then the $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ algorithm proceeds as before. The $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}$ function for nice concepts works similar. Having treated the trivial cases, the second step is to test if the concept is a conjunction. In that case the approximation is obtained by splitting the concept conjunct-wise and making a recursive call for each conjunct. For all other nice concepts the approximation is computed as in $\mathsf{c\text{-}approx}_{\mathcal{ALE}}$, besides the recursive calls refer to $\mathsf{c\text{-}nice\text{-}approx}_{\mathcal{ALE}}$.

Observe that the test conditions for nice concepts can be checked in linear time once the concept description is unfolded and in NNF. Unfolding and transforming the concept description into NNF always have to be performed to apply $\text{c-approx}_{\mathcal{ALE}}$, so that testing whether a concept is nice is hardly any extra effort when approximating a concept.

3.2 Approximating Nice Concepts in TBoxes

If an $\mathcal{ALC}$-TBox is to be translated into an $\mathcal{ALE}$-TBox, the concept description on the right-hand side of each concept definition has to be replaced by its approximation. For practical applications it is not feasible to perform such a translation in a naive way. The idea for optimizing this procedure is to re-use the approximation of a defined concept when approximating concept descriptions that in turn make use of this defined concept. More precisely, if we have already obtained the approximation of C and want to compute the approximation of, e.g., $(D \sqcap \exists r.C)$, we would like to be able to insert the concept description $\mathsf{approx}(C)$ directly into the right place in the concept description of $\mathsf{approx}(D \sqcap \exists r.C)$. Unfortunately, this approach does not work for arbitrary $\mathcal{ALC}$-concept descriptions due to possible interactions between different parts of the concept description. Nice concepts, however, are defined to rule out this kind of interaction. Hence, besides speeding-up the computation of a single approximation, the property of being a nice concept also is a prerequisite for caching and the re-use of already computed approximations. For example, if the defined concepts C_1, C_2, C_3 from the following TBox (with A, B and D as primitive concepts)

$$\begin{aligned} \mathcal{T} \ = \{ \ \ & C_1 = (\exists r.\neg A) \sqcup (\exists r.B), \\ & C_2 = \exists r.(\forall r.D \sqcup \neg E) \sqcap C_1 \sqcap \neg B, \\ & C_3 = \neg\,(\forall r.\exists r.(\neg D \sqcap A) \sqcup \neg C_1 \sqcup \neg C_2) \ \ \} \end{aligned}$$

are to be approximated and C_1 is approximated first, then this concept description can be re-used in subsequent approximations. If unfolded and transformed into NNF the concepts C_2 and C_3 are nice concepts. Hence, the approximation of C_2 is the conjunction of $\mathsf{approx}(\exists r.(\forall r.D \sqcup \neg E))$ and $\mathsf{approx}(C_1)$ and $\mathsf{approx}(B)$, where the already computed approximation of C_1 can be inserted directly. For C_3 we can re-use both approximations of C_1 and C_2 directly and only have to compute the approximation of $\exists r.\forall r.(D \sqcup \neg A)$. Thus, the cost for approximating the entire TBox is reduced heavily.

4 Conclusion and Future Work

In this paper we have presented some first steps towards optimizing the computation of approximations. The main idea is to identify concepts that can be decomposed into parts which then can be approximated independently. These so-called nice concepts are structured in such a way that the top-level conjuncts cannot interact with one another. Therefore, each conjunct can be approximated separately. Detecting nice concepts and approximating each of their conjuncts independently should be especially powerful in the context of translating entire $\mathcal{ALC}$-TBoxes into $\mathcal{ALE}$-TBoxes because it enables the direct re-use of already computed approximations and caching. Unfortunately, the conditions for nice concepts are very strict.

It is an open problem whether the rather strict conditions for nice concepts can be relaxed. To determine if independent approximation of nice concepts is a real benefit for practical applications, requires an implementation of modular approximation. Moreover, it is unknown if nice concepts occur frequently in application TBoxes.

Another open problem is whether the given conditions for nice concepts can be extended to the case where $\mathcal{ALCN}$-concept descriptions are approximated by $\mathcal{ALEN}$-concept descriptions.

References

[1] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.

[2] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In T. Dean, editor, *Proc. of IJCAI-99*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann.

[3] F. Baader and A.-Y. Turhan. On the problem of computing small representations of least common subsumers. In *Proceedings of the German Conference on Artificial Intelligence, 25th German Conference on Artificial Intelligence (KI 2002)*, Lecture Notes in Artificial Intelligence. Springer–Verlag, 2002. To appear.

[4] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, D. McGuiness, and M.-A. Williams, editors, *Proc. of KR-02*, San Francisco, CA, 2002. Morgan Kaufmann Publishers.

[5] 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/.

[6] S. Brandt and A.-Y. Turhan. An approach for optimizing $\mathcal{ALE}$-approximation of $\mathcal{ALC}$-concepts. LTCS-Report 02-03, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2002. See http://lat.inf.tu-dresden.de/research/reports.html.

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

[8] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

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

[10] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

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