Commonality-Based ABox Retrieval

Thomas Mantay*

Abstract

In commonality-based information retrieval, the commonalities of user-specified examples of desired information are used for information retrieval. As described in previous literature, such a retrieval mechanism can be built using a combination of description logic reasoning services which makes this retrieval technique an interesting research topic in the field of description logic applications. However, as will be shown in this article, the existing technique does not always yield a desirable retrieval result. Therefore, we present a theoretical framework useful for commonality-based information retrieval and other description logic applications. Part of this framework is a formal definition of the notion of ABox subsumption using the standard semantics for ABoxes and an algorithm for deciding this problem. Based on ABox subsumption, we will present an operation for determining the least common subsuming ABox of a set of ABoxes w.r.t. the expressive description logics $\mathcal{ALENR}$ and $\mathcal{ALQ}$. As a by-product, an algorithm for computing the most specific concept of individuals occurring in ABoxes w.r.t. the two mentioned description logics will be developed. We present soundness, completeness, and complexity results and show how the developed reasoning services can be used for a variant of commonality-based information retrieval which we call commonality-based ABox retrieval.

1 Introduction

Terminological knowledge representation systems based on description logics have proven to be a useful means for representing the knowledge of an application domain in a structured and formally well understood way [2]. The knowledge base as part of a terminological knowledge representation system

*Labor für Künstliche Intelligenz, Universität Hamburg, Vogt-Kölln-Straße 30, D-22527 Hamburg, mantay@informatik.uni-hamburg.de

usually consists of a terminological and an assertional component. The terminological component, also called TBox, represents the vocabulary used in the assertional component referred to as the ABox. It contains definitions of *concepts* which describe sets of individuals. Concepts are built out of atomic components and roles (representing binary relations between individuals) using the concept constructors provided by the underlying description logic language. For example, the set of *grandmothers* can be described using the atomic concepts woman and parent and the role has-child:

$$\mathsf{woman} \sqcap (\geq 1\ \mathsf{has\text{-}child}\ \mathsf{parent}).$$

The ABox is a (partial) instantiation of the vocabulary defined in the terminological component and contains assertions relating either individuals to concepts, or individuals to each other via roles. For instance, it can be stated that the individual Mary is considered to be an instance of the concept mother and that she has a child called Tom:

$$\{\mathsf{Mary} : \mathsf{mother}, (\mathsf{Mary}, \mathsf{Tom}) : \mathsf{has\text{-}child}\}.$$

A central feature of terminological knowledge representation systems based on description logics is a set of reasoning services with the ability to deduce implicit knowledge from explicitly represented knowledge in the knowledge base. For instance, the subsumption relation between two concepts can be determined. Intuitively, a concept C *subsumes* a concept D if the set of individuals represented by C is a superset of the set of individuals represented by D, i.e. if C is more general than D. Determining the *most specific concept* (MSC) describes the problem of computing the most specific concept (from the infinite space of all concepts in the description logic) of which a given individual is an instance. For two concepts C and D, the *least common subsumer* (LCS) operation determines the most specific concept (from the description logic language) which subsumes C and D. Both the MSC of an individual and the LCS of concepts depend on the underlying description logic language. Also, reasoning services concerning ABoxes have been subject to research. Algorithms for checking the *satisfiability of ABoxes* are among the most prominent ABox reasoning services intensively studied so far. Intuitively speaking, an ABox is satisfiable iff the conjunction of its assertions does not lead to a contradiction. Given an ABox indidivual and a concept, *instantiation* describes the problem of determining whether or not the individual is an instance of the concept. An instantiation algorithm can be used in order to solve the *instance retrieval* problem which describes the task of computing all instances of a given concept.

The described reasoning services can be used for commonality-based information retrieval which is a relatively new application context for terminological knowledge representation systems citeMantay-TR-99. With this kind of information retrieval, the goal is to provide a user of an information system with an example-based query mechanism. More specifically, the "commonalities" of user-specified examples of desired information are used as a retrieval filter on an underlying database. The database is modeled as an ABox and the set of database items is modeled by a subset of the set of individuals in the ABox. The conceptual abstractions are represented by the MSCs of the selected individuals and the notion of commonality is formalized by the LCS operation. Eventually, information retrieval is performed by instance retrieval using the LCS concept.

However, two problems occur when abstracting from ABox individuals to concepts by means of the MSC operation. The possible presence of ABox cycles is the reason for the MSC to not exist for all individuals. This problem can be circumvented by approximating the MSC of an individual by considering the conjunction of most specific concepts included in the TBox of which the individual is an instance. But there is another shortcoming from a more practical point of view. Due to the MSC abstraction, relevant information concerning individuals given by the user are filtered out and thus, cannot be taken into account in the information retrieval process subsequently. In this paper, we present new ABox inference services in order to circumvent this problem.

For example, consider a TV information system equipped with a commonality-based retrieval mechanism. In this context, let us assume the presence of a TBox containing the relevant vocabulary of the TV world. Furthermore, let $\mathcal{A}$ be an ABox such that

$$\begin{aligned}\{&\mathsf{Armageddon} : \mathsf{scifi\text{-}movie}, (\mathsf{Armageddon}, \mathsf{Bruce\text{-}Willis}) : \mathsf{has\text{-}actor},\\ &\mathsf{Pulp\text{-}Fiction} : \mathsf{action\text{-}movie}, (\mathsf{Pulp\text{-}Fiction}, \mathsf{Bruce\text{-}Willis}) : \mathsf{has\text{-}actor},\\ &\mathsf{Bruce\text{-}Willis} : \mathsf{actor}\} \subseteq \mathcal{A}.\end{aligned}$$

In the given subset of $\mathcal{A}$ we state that Armageddon is an instance of the concept scifi-movie and Pulp-Fiction is an instance of action-movie. In both movies, Bruce-Willis is starring as an actor. The MSCs[1] of Armageddon and Pulp-Fiction are given by

$$\begin{aligned} msc_{\mathcal{A}}(\mathsf{Armageddon}) &\equiv \mathsf{scifi\text{-}movie} \sqcap \exists\, \mathsf{has\text{-}actor.actor} \text{ and}\\ msc_{\mathcal{A}}(\mathsf{Pulp\text{-}Fiction}) &\equiv \mathsf{action\text{-}movie} \sqcap \exists\, \mathsf{has\text{-}actor.actor},\end{aligned}$$

[1]In this example, $\mathcal{ALENR}$ is assumed to be the underlying description logic. The language will be formally introduced later.

respectively. Assuming movie to be the LCS of $\mathsf{scifi\text{-}movie}$ and $\mathsf{action\text{-}movie}$, the LCS of the two MSCs,

$$C := \mathsf{movie} \sqcap \exists\, \mathsf{has\text{-}actor}.\mathsf{actor}$$

is used for instance retrieval on $\mathcal{A}$ yielding the set of movies with *any* actor. The information that $\mathsf{Bruce\text{-}Willis}$ is starring in both $\mathsf{Armageddon}$ and $\mathsf{Pulp\text{-}Fiction}$ is no longer present in C since this fact was "lost" in the MSC abstraction. However, this behavior is certainly undesired for users who prefer movies with this actor.

Therefore, in this article we present new ABox inference services for commonality-based information retrieval where assertional knowledge is integrated in the commonality computation. More specifically, after formally introducing some important definitions and notations in Section 2, we give a definition of the notion of ABox subsumption in terms of the standard semantics for ABoxes in Section 3. We also provide an algorithm for deciding ABox subsumption and prove its soundness and completeness. Based on ABox subsumption, in Section 4 we will introduce the least common subsuming ABox as an operation which determines the most specific ABox (w.r.t. ABox subsumption) which subsumes the ABoxes to which the operation is applied. As a by-product, an algorithm for computing the MSC of individuals occurring in ABoxes w.r.t. $\mathcal{ALENR}$ and $\mathcal{ALQ}$ will be developed. We conclude with a summary and an outlook on possible future research topics. Due to the new ABox inference services, we adapt the commonality-based information retrieval scenario outlined above. We assume that the database is given by a set of ABoxes where each ABox models exactly one specific information item in the database. For instance, in the TV information system the database of TV broadcasts is represented by a set of ABoxes where each ABox models exactly one broadcast. The two movies $\mathsf{Armageddon}$ and $\mathsf{Pulp\text{-}Fiction}$ could be represented by the ABoxes

$$\begin{aligned}
\mathcal{A} &:= \{\mathsf{Armageddon} : \mathsf{scifi\text{-}movie}, (\mathsf{Armageddon}, \mathsf{Bruce\text{-}Willis}) : \\
&\quad\ \mathsf{has\text{-}actor}, \mathsf{Bruce\text{-}Willis} : \mathsf{actor}\} \text{ and} \\
\mathcal{B} &:= \{\mathsf{Pulp\text{-}Fiction} : \mathsf{action\text{-}movie}, (\mathsf{Pulp\text{-}Fiction}, \mathsf{Bruce\text{-}Willis}) : \\
&\quad\ \mathsf{has\text{-}actor}, \mathsf{Bruce\text{-}Willis} : \mathsf{actor}\},
\end{aligned}$$

respectively. Now we describe how to compute the commonalities of $\mathcal{A}$ and $\mathcal{B}$ by means of the least common subsuming ABox operation. In order to simplify the task, we replace the individuals which are subject to the LCSA operation by a common "anchor" which formally is a new individual not occurring in any of the database ABoxes. In our example, the anchor is named Broadcast and replaces Armageddon in $\mathcal{A}$, Pulp-Fiction in $\mathcal{B}$, and the corresponding broadcast individuals in the other database ABoxes. Then, the least common subsuming ABox of $\mathcal{A}$ and $\mathcal{B}$ is given by

$$\{\mathsf{Broadcast} : \mathsf{movie}, (\mathsf{Broadcast}, \mathsf{Bruce\text{-}Willis}) : \mathsf{has\text{-}actor}, \mathsf{Bruce\text{-}Willis} : \mathsf{actor}\}.$$

Eventually, retrieval is performed by filtering those ABoxes from the database which are subsumed by the least common subsuming ABox. Thereby, the information that Bruce-Willis is an actor in both movies can be considered subsequently since it is still present in the ABox used for retrieval.

Whereas we considered a TV information system as an example, the retrieval mechanism can be applied to a number of applications, e.g. document retrieval and retrieval on picture databases where the content of a document/picture is modeled by an ABox. These applications are particularly interesting for commonality-based ABox retrieval due to the typical occurrance of ABox individuals. As a precondition for the described information retrieval mechanism, we extend the unique name assumption usually adopted for individuals to sets of ABoxes, i.e. individuals with different names are interpreted as different individuals, even if they occur in different ABoxes in the database. Also, we take for granted an open world assumption for ABoxes, i.e. the truth value of assertional knowledge not explicitly represented in the ABox is considered to be unknown. For instance, given the above ABox $\mathcal{A}$, we cannot conclude that Bruce-Willis is the *only* actor in Armageddon.

2 Preliminaries

In the following two sections, let $\mathcal{L}$ be a description logic which includes a constructor for full concept negation and for which there exists an algorithm for checking satisfiability of ABoxes w.r.t. $\mathcal{L}$. All concepts and roles mentioned in the sequel are concepts and roles in $\mathcal{L}$. We also assume that there exists an interpretation $\mathcal{I}$ consisting of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ for the interpretation of the concepts and roles in $\mathcal{L}$.

Definition 1 (Concept Relations) *Let C and D be concepts. Then we introduce the following concept relations:*

- C is subsumed by D *($C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$ of C and D.*

- *C is* strictly subsumed by *D ($C \sqsubset D$) iff $C^{\mathcal{I}} \subset D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$ of C and D.*

- *C is* equivalent *to D ($C \equiv D$) iff both $C \sqsubseteq D$ and $D \sqsubseteq C$ hold.*

$\square$

We will now introduce the assertional part of a knowledge base.

Definition 2 (ABox) *Assume that there exists an alphabet of symbols, called* individuals*, disjoint from the sets of concepts and roles of $\mathcal{L}$. Then we say that*

- *a* concept assertion *is a syntactic expression of the form $a : C$, and*

- *a* role assertion *is a syntactic expression of the form $(a, b) : R$,*

where a and b are individuals, C is a concept, and R is a role. An assertion *is either a concept assertion or a role assertion. A finite set of assertions is called an* ABox*. $Ind(\mathcal{A})$ denotes the set of individuals and $Roles(\mathcal{A})$ denotes the set of roles occurring in assertions of $\mathcal{A}$.*

$\square$

Given a role assertion $(a, b) : R$ in an ABox $\mathcal{A}$, we say that b is an R-successor of a in $\mathcal{A}$. If it is clear from the context which ABox is meant, we just say that b is an R-successor of a. Note that $\emptyset$ also qualifies for an ABox. Now the standard semantics for ABoxes will be defined starting with a semantical characterization of individuals.

Definition 3 (Interpretation of Individuals) *The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ for concepts of $\mathcal{L}$ is extended to individuals by mapping them to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$.*

$\square$

We can now define the semantics of ABoxes.

Definition 4 (Semantics of an ABox) *Let a and b be individuals, C a concept, and R a role. An interpretation $\mathcal{I}$* satisfies *a concept assertion $a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and it satisfies a role assertion $(a, b) : R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. We say that an interpretation $\mathcal{I}$ is a* model of an assertion *α iff $\mathcal{I}$ satisfies α. An interpretation $\mathcal{I}$ is a* model of an ABox *$\mathcal{A}$ iff $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$.*

$\square$

An (un-)satisfiable ABox is also called *(in-)consistent.* In the next section, we will define the notion of ABox subsumption and provide an algorithm for deciding the ABox subsumption problem.

3 ABox Subsumption

Based on the ABox semantics, we can define the following ABox relations.

Definition 5 (Relations Concerning ABoxes) *Let $\mathcal{A}$ and $\mathcal{B}$ be ABoxes, $a : C$ and $b : D$ concept assertions, and $(a, b) : R$ and $(c, d) : S$ role assertions. Then we define the following relations:*

- $a : C$ is subsumed by $b : D$ *($a : C \sqsubseteq b : D$) iff, for all interpretations $\mathcal{I}$, $\mathcal{I}$ is a model of $a : C$ implies that $\mathcal{I}$ is also a model of $b : D$.*
- $(a, b) : R$ is subsumed by $(c, d) : S$ *$[(a, b) : R \sqsubseteq (c, d) : S]$ iff, for all interpretations $\mathcal{I}$, $\mathcal{I}$ is a model of $(a, b) : R$ implies that $\mathcal{I}$ is also a model of $(c, d) : S$.*
- $a : C$ is equivalent to $b : D$ *($a : C \equiv b : D$) iff both $a : C \sqsubseteq b : D$ and $b : D \sqsubseteq a : C$ hold.*
- $(a, b) : R$ is equivalent to $(c, d) : S$ *$[(a, b) : R \equiv (c, d) : S]$ iff both $(a, b) : R \sqsubseteq (c, d) : S$ and $(c, d) : S \sqsubseteq (a, b) : R$ hold.*
- $\mathcal{A}$ is subsumed by $\mathcal{B}$ *($\mathcal{A} \sqsubseteq \mathcal{B}$) iff, for all interpretations $\mathcal{I}$, $\mathcal{I}$ is a model of $\mathcal{A}$ implies that $\mathcal{I}$ is also a model of $\mathcal{B}$.*
- $\mathcal{A}$ is strictly subsumed by $\mathcal{B}$ *($\mathcal{A} \sqsubset \mathcal{B}$) iff $\mathcal{A} \sqsubseteq \mathcal{B}$ holds and there exists a model of $\mathcal{B}$ which is no model of $\mathcal{A}$.*
- $\mathcal{A}$ *is* equivalent *to* $\mathcal{B}$ ($\mathcal{A} \equiv \mathcal{B}$) *iff both $\mathcal{A} \sqsubseteq \mathcal{B}$ and $\mathcal{B} \sqsubseteq \mathcal{A}$ hold.*

□

In Definition 5 we use the same symbols for ABox subsumption, strict subsumption, and equivalence as for the corresponding relations between concepts (see Definition 1). It will be clear from the context whether these relations refer to ABoxes or concepts. From the preceding definition we can immediately derive the following observations.

Proposition 1 *Let $\mathcal{A}$ and $\mathcal{B}$ be ABoxes, $a : C$ and $b : D$ concept assertions, and $(a, b) : R$ and $(c, d) : S$ role assertions. Then*

(i) $a : C \sqsubseteq b : D$ *iff* $a = b$ *and* $C \sqsubseteq D$.

(ii) $(a, b) : R \sqsubseteq (c, d) : S$ *iff* $a = c$, $b = d$, *and* R *is a subrole*[2] *of* S.

[2] We assume the subrole relationship between two roles to be appropriately defined in the interpretation of the concept language.

(iii) *If* $\mathcal{A} = \emptyset$, *then* $\mathcal{B} \sqsubseteq \mathcal{A}$.

(iv) *If* $\mathcal{A}$ *is unsatisfiable, then* $\mathcal{A} \sqsubseteq \mathcal{B}$.

(v) $\mathcal{A} \sqsubseteq \mathcal{A} \cap \mathcal{B}$.

(vi) $\mathcal{A} \cup \mathcal{B} \sqsubseteq \mathcal{A}$.

□

The claims in Proposition 1 are trivial consequences of Definition 5 and will be of use subsequently. Notice the meaning of (v) and (vi) in the proposition: Whereas the conjunction (disjunction) of concepts leads to a more specific (general) concept w.r.t. to concept subsumption, the corresponding operation for ABoxes, ABox intersection (union), leads to a more general (specific) ABox w.r.t. ABox subsumption.

We will now present a calculus for deciding ABox subsumption. The decision procedure follows the ideas for deciding the applicability of default rules on ABoxes as presented in [7]. The ABox subsumption problem can be reduced to the ABox satisfiability problem. More specifically, $\mathcal{A}$ subsumes $\mathcal{B}$ iff the "negation of each assertion β" in $\mathcal{B}$ added to the assertions in $\mathcal{A}$ leads to an unsatisfiable ABox. Intuitively, the negation of a concept assertion $a : C$ will be the concept assertion $a : \neg C$ and the negation of a role assertion can be expressed by an ABox consisting of the two concept assertions $a : \forall\, R.A$ and $b : \neg A$ where A is a new atomic concept not already present in neither $\mathcal{A}$ nor $\mathcal{B}$. Thereby, the concept assertion $a : \neg C$ is well-defined because a language constructor for full negation has been presumed to be present in the underlying description logic $\mathcal{L}$.

Lemma 1 *Let* $a : C$ *be a concept assertion,* $(a, b) : R$ *a role assertion,* A *an atomic concept,* α *an assertion in which* A *does not appear, and* $\mathcal{I}$ *an interpretation. Then*

(i) $\mathcal{I}$ *is a model of* $a : C$ *iff* $\mathcal{I}$ *is not a model of* $a : \neg C$.

(ii) *If* $\mathcal{I}$ *is a model of* $(a, b) : R$, *then* $\mathcal{I}$ *is not a model of* $\{a : \forall R.A, b : \neg A\}$.

(iii) *If* $\mathcal{A} = \{\alpha, a : \forall\, R.A, b : \neg A\}$ *is unsatisfiable, then* α *is of the form* $(a, b) : R$.

Proof. $\mathcal{I}$ is a model of $a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ iff $a^{\mathcal{I}} \notin \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} = (\neg C)^{\mathcal{I}}$ which proves (i). Now we will prove (ii) by contradiction. Suppose $\mathcal{I}$ is a model of $(a, b) : R$ and also a model of $\{a : \forall\, R.A, b : \neg A\}$. Since $\mathcal{I}$ is a model of $a : \forall\, R.A$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, it follows that $b^{\mathcal{I}} \in A^{\mathcal{I}}$. However, since $\mathcal{I}$ is also a model of $b : \neg A$, we have that $b^{\mathcal{I}} \in (\neg A)^{\mathcal{I}} \cap A^{\mathcal{I}} = \emptyset$ which leads to a contradiction. (*iii*) will also be proved by contradiction. We first observe that $\{a : \forall\, R.A, b : \neg A\} \subset \mathcal{A}$ is satisfiable. Suppose α is a concept assertion of the form $c : C$ such that A does not occur as a subexpression in C. Then, there exists a model $\mathcal{I}$ of $\{c : C, a : \forall\, R.A, b : \neg A\}$ for all $c \in \Delta^{\mathcal{I}}$, which again contradicts to our assumption. Now suppose α is a role assertion of the form $(c, d) : S$. If $c \neq a$, then there obviously exists a model $\mathcal{I}$ of $\{(c, d) : S, a : \forall\, R.A, b : \neg A\}$ for all $d \in \Delta^{\mathcal{I}}$ and roles S. Also, if $d \neq b$, there exists a model $\mathcal{I}$ of $\{(c, d) : S, a : \forall\, R.A, b : \neg A\}$ for all $c \in \Delta^{\mathcal{I}}$ and roles S. Finally, if $S \neq R$, we can easily find a model $\mathcal{I}$ of $\{(c, d) : S, a : \forall R.A, b : \neg A\}$ for all $c \in \Delta^{\mathcal{I}}$ and $d \in \Delta^{\mathcal{I}}$. This proves that $c = a$, $d = b$, and $S = R$ and α is of the form $(a, b) : R$. $\square$

Theorem 1 *Let $\mathcal{A}$ and $\mathcal{B}$ be ABoxes. Then $\mathcal{A} \sqsubseteq \mathcal{B}$ iff, for all $\beta \in \mathcal{B}$, the following conditions hold:*

(i) If β is of the form $a : C$, then $\mathcal{A} \cup \{a : \neg C\}$ is unsatisfiable.

(ii) If β is of the form $(a, b) : R$, then $\mathcal{A} \cup \{a : \forall R.A, b : \neg A\}$ is unsatisfiable.

Proof. We first prove "$\Rightarrow$" (completeness). Suppose $\mathcal{A} \sqsubseteq \mathcal{B}$. In case $\mathcal{B} = \emptyset$, nothing has to be shown. Therefore, let $a : C \in \mathcal{B}$ and $\mathcal{I}$ be a model of $\mathcal{A}$. Since $\mathcal{A} \sqsubseteq \mathcal{B}$, we have that $\mathcal{I}$ is also a model of $\mathcal{B}$ and of $a : C$ because $a : C \in \mathcal{B}$. By Lemma 1 (*i*) we know that $\mathcal{I}$ is not a model of $a : \neg C$ and hence, $\mathcal{I}$ is not a model of $\mathcal{A} \cup \{a : \neg C\}$. Now let $(a, b) : R \in \mathcal{B}$ and $\mathcal{I}$ be a model of $\mathcal{A}$. Since $\mathcal{A} \sqsubseteq \mathcal{B}$, $\mathcal{I}$ is also a model of $\mathcal{B}$ and of $(a, b) : R$ because $(a, b) : R \in \mathcal{B}$. According to Lemma 1 (*ii*), $\mathcal{I}$ is not a model of $\{a : \forall R.A, b : \neg A\}$ and hence, $\mathcal{I}$ is not a model of $\mathcal{A} \cup \{a : \forall R.A, b : \neg A\}$. Now "$\Leftarrow$" (soundness) will be proved. If $B = \emptyset$, the claim follows by Proposition 1 (*iii*). Otherwise, let $a : C \in \mathcal{B}$, $\mathcal{A} \cup \{a : \neg C\}$ be unsatisfiable, and $\mathcal{I}$ be a model of $\mathcal{A}$. If $\mathcal{A} \cup \{a : \neg C\}$ is unsatisfiable and $\mathcal{I}$ is a model of $\mathcal{A}$, it follows that $\mathcal{I}$ is not a model of $a : \neg C$. By Lemma 1 (*i*) this means that $\mathcal{I}$ is a model of $a : C$. Now let $(a, b) : R \in \mathcal{B}$ and let $\mathcal{A} \cup \{a : \forall\, R.A, b : \neg A\}$ be unsatisfiable. Then $\mathcal{I}$ is no model of $\mathcal{A} \cup \{a : \forall\, R.A, b : \neg A\}$. Since A is an atomic concept which does not appear in $\mathcal{A}$, we have that, for all $\alpha \in \mathcal{A}$, the ABoxes $\{\alpha, a : \forall\, R.A\}$ and $\{\alpha, b : \neg A\}$ are satisfiable. But this implies that there exists an $\alpha \in \mathcal{A}$ such that $\{\alpha, a : \forall\, R.A, b : \neg A\}$ is unsatisfiable. By Lemma 1 (*iii*) it follows that α is of the form $(a, b) : R$. Hence, $\mathcal{I}$ is also a model of $(a, b) : R$, which completes the proof. $\square$

The function *abox-subsumes* implements an ABox subsumption test given ABoxes $\mathcal{A}$ and $\mathcal{B}$ w.r.t. $\mathcal{L}$.

Algorithm 1 abox-subsumes($\mathcal{A}, \mathcal{B}$)

```
for all $\beta \in \mathcal{B}$ do
  if $\beta$ is of the form $a : C$ then
    $not(abox\text{-}satisfiable(\mathcal{A} \cup \{a : \neg C\}))$
  else if $\beta$ is of the form $(a, b) : R$ then
    // let $A$ be a new atomic concept occurring neither in $\mathcal{A}$ nor in $\mathcal{B}$
    $not(abox\text{-}satisfiable(\mathcal{A} \cup \{a : \forall R.A, b : \neg A\}))$
  end if
end for
```

Theorem 2 *Let $\mathcal{A}$ and $\mathcal{B}$ be ABoxes w.r.t. $\mathcal{L}$. Then* abox-subsumes($\mathcal{A}, \mathcal{B}$) *terminates and returns true iff $\mathcal{B} \sqsubseteq \mathcal{A}$.*

Proof. The termination of *abox-subsumes* follows from the termination of the function *abox-satisfiable*, which implements the ABox satisfiability test. In Algorithm 1 we check, for every concept assertion $a : C$ (role assertion $(a, b) : R$) in $\mathcal{B}$, whether the ABox $\mathcal{A} \cup \{a : \neg C\}$ ($\mathcal{A} \cup \{a : \forall R.A, b : \neg A\}$) is unsatisfiable. Hence, the claim is a consequence of Theorem 1. □

Theorem 3 *The subsumption problem for ABoxes w.r.t. $\mathcal{L}$ is at most as complex as ABox satisfiability checking for ABoxes w.r.t. $\mathcal{L}$.*

Proof. According to Theorem 1, it can be decided by $|\mathcal{B}|$ ABox satisfiability tests if $\mathcal{A}$ is subsumed by $\mathcal{B}$. Hence, the number of satisfiability tests is polynomial in the length of $\mathcal{B}$. This shows that the complexity of the ABox subsumption problem only depends on the complexity of deciding ABox satisfiability of ABoxes w.r.t. $\mathcal{L}$. □

The ABox satisfiability problem has been studied for a variety of description logic languages. For $\mathcal{ALCNR}$ [1] and $\mathcal{ALCQ}$ [4], algorithmus using exponential space in the size of the ABox were proposed. In [3], an algorithm for checking satisfiability of ABoxes w.r.t. $\mathcal{ALCNH}_{R^+}$ is presented using exponential space in the size of the ABox as well. Prominent features of the description logic $\mathcal{ALCNH}_{R^+}$ are number restrictions, role hierarchies, transitively closed roles, and generalized concept inclusions. For ABoxes w.r.t. the same description logic language augmented by inverse roles and qualified number restrictions, $\mathcal{ALCHQI}_{R^+}$, Horrocks, Sattler, and Tobies [5] showed that satisfiability checking is decidable however they do not give a lower bound complexity. Presumably, due to the existence of role hierarchies, ABox satisfiability checking for $\mathcal{ALCNH}_{R^+}$ and $\mathcal{ALCHQI}_{R^+}$ is no longer in PSPACE but in EXPTIME.

In this section, we defined the notion of ABox subsumption for ABoxes w.r.t. to a description logic $\mathcal{L}$ and provided an algorithm to decide this problem. Thereby, only two requirements are imposed on $\mathcal{L}$: A constructor for full concept negation must be present and a sound and complete algorithm for checking ABox satisfiability must be available for ABoxes w.r.t. $\mathcal{L}$. Instead of the explicit presence of a full negation operation, it suffices if $\mathcal{L}$ is a sublanguage of another description logic fulfilling the two requirements. Subsequently, we will make use of this property. We also proved soundness and completeness of the ABox subsumption algorithm and proved that, even though ABox subsumption can be reduced to ABox satisfiablity checking, the problem does not become more complex. Subsequently, the results of this section will play an important role in the definition and computation of a least common subsuming ABox operation.

4 Least Common Subsuming ABox

In this section, we are interested in the inference task of computing the LCSA of ABoxes w.r.t. a description logic $\mathcal{L}$. As will be discussed in more detail throughout this section, the algorithm for LCSA computation presented here requires the presence of an LCS algorithm for concepts of $\mathcal{L}$. Since $\mathcal{ALENR}$ and $\mathcal{ALQ}$ are two of the most expressive description logics for which an LCS operation has been presented [6], we restrict the presentation of the LCSA operation to ABoxes w.r.t. $\mathcal{ALENR}$ and $\mathcal{ALQ}$. The description logic $\mathcal{ALENR}$ ($\mathcal{ALQ}$) is a sublanguage of $\mathcal{ALCNR}$ ($\mathcal{ALCQ}$) which both include a full concept negation operation. Moreover, Buchheit, Donini, and Schaerf showed in [1] that ABox satisfiability checking for ABoxes w.r.t. $\mathcal{ALCNR}$ is a decidable problem. Hollunder and Baader proved the same result for $\mathcal{ALCQ}$ [4]. Let us first formally introduce syntax and semantics of these two description logics.

Definition 6 (Syntax of $\mathcal{ALENR}$) *Let $\mathcal{C}$ be a set of* atomic concepts *and $\mathcal{R}$ a set of* atomic roles *disjoint from $\mathcal{C}$. $\mathcal{ALENR}$ concepts are recursively defined as follows:*

- *The symbols $\top$ and $\bot$ are $\mathcal{ALENR}$ concepts (top concept, bottom concept).*
- *A and $\neg A$ are $\mathcal{ALENR}$ concepts for each $A \in \mathcal{C}$ (atomic concept, negated atomic concept).*
- *Let C and D be $\mathcal{ALENR}$ concepts, $R \in \mathcal{R}$ an atomic role, and $n \in I\!N \cup \{0\}$. Then*

  - $C \sqcap D$ *(concept conjunction),*
  - $\exists\, R.C$ *(existential role quantification),*
  - $\forall\, R.C$ *(universal role quantification),*
  - $(\geq\ n\, R)$ *($\geq$-restriction), and*
  - $(\leq\ n\, R)$ *($\leq$-restriction)*

  are also $\mathcal{ALENR}$ concepts.

- *If R and S are roles, then $R \sqcap S$ is a role in $\mathcal{ALENR}$ (role conjunction).* $\square$

Definition 7 (Syntax of $\mathcal{ALQ}$) *Let $\mathcal{C}$ be a set of* atomic concepts *and $\mathcal{R}$ a set of* roles *disjoint from $\mathcal{C}$. $\mathcal{ALQ}$ concepts are recursively defined as follows:*

- *The symbols $\top$ and $\bot$ are $\mathcal{ALQ}$ concepts (top concept, bottom concept).*
- *A and $\neg A$ are $\mathcal{ALQ}$ concepts for each $A \in \mathcal{C}$ (atomic concept, negated atomic concept).*
- *Let C and D be $\mathcal{ALQ}$ concepts, $R \in \mathcal{R}$ a role, and $n \in I\!N \cup \{0\}$. Then*
  - $C \sqcap D$ *(concept conjunction),*
  - $(\geq\ n\, R\, C)$ *(qualified $\geq$-restriction), and*
  - $(\leq\ n\, R\, C)$ *(qualified $\leq$-restriction)*

  are also $\mathcal{ALQ}$ concepts. $\square$

The languages $\mathcal{ALENR}$ and $\mathcal{ALQ}$ can be extended to $\mathcal{ALCNR}$ and $\mathcal{ALCQ}$ by adding a constructor for full concept negation: $\neg C$. The semantics of $\mathcal{ALENR}$ and $\mathcal{ALQ}$ concepts is defined in terms of an interpretation.

Definition 8 (Interpretation, Model, Satisfiability) *An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of an $\mathcal{ALENR}$ or $\mathcal{ALQ}$ concept consists of a non-empty set $\Delta^{\mathcal{I}}$ (the* domain *of $\mathcal{I}$) and an* interpretation function *$\cdot^{\mathcal{I}}$. The interpretation function maps every atomic concept A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role R to a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is recursively extended to a complex $\mathcal{ALENR}$ or $\mathcal{ALQ}$ concept as follows. Assume that $C^{\mathcal{I}}, D^{\mathcal{I}}$ and $R^{\mathcal{I}}, S^{\mathcal{I}}$ are already given and $n \in I\!N \cup \{0\}$. Then*

- $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$,
- $\bot^{\mathcal{I}} := \emptyset$,

- $(\neg A)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$,
- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$,
- $(R \sqcap S)^{\mathcal{I}} := R^{\mathcal{I}} \cap S^{\mathcal{I}}$,
- $\exists\, R.C^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \exists b : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$,
- $\forall\, R.C^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \forall b : (a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$,
- $(\geq\ n\, R)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \sharp\{b|(a,b) \in R^{\mathcal{I}}\} \geq n\}$,
- $(\leq\ n\, R)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \sharp\{b|(a,b) \in R^{\mathcal{I}}\} \leq n\}$,
- $(\geq\ n\, R\, C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \sharp\{aR^{\mathcal{I}} \cap C^{\mathcal{I}}\} \geq n\}$, *and*
- $(\leq\ n\, R\, C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} | \sharp\{aR^{\mathcal{I}} \cap C^{\mathcal{I}}\} \leq n\}$,

where $aR^{\mathcal{I}} := \{b \in \Delta^{\mathcal{I}} | (a,b) \in R^{\mathcal{I}}\}$. *An interpretation* $\mathcal{I}$ *is a* model *of an* $\mathcal{ALENR}$ *or* $\mathcal{ALQ}$ *concept* C *iff* $C^{\mathcal{I}} \neq \emptyset$. *If* C *has a model,* C *is called* satisfiable *(or* consistent*).*

$\square$

Note that the constructor $\top$ can be expressed by $(\geq\ 0\, R)$ in $\mathcal{ALENR}$ and by $(\geq\ 0\, R\, \top)$ in $\mathcal{ALQ}$. The concept $\bot$ is expressible by $A \sqcap \neg A$ in both description logic languages. The semantics can be extended to the constructor for full concept negation by defining $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$.

For some explanations of the algorithms presented subsequently, we introduce the concept depth.

Definition 9 (Depth) *Let* C *be either an* $\mathcal{ALENR}$ *or* $\mathcal{ALQ}$ *concept. Then the* depth *of* C *is recursively defined over its structure.*

- *If* $C = \exists\, R.C'$, $C = \forall\, R.C'$, $C = (\geq\ n\, R\, C')$, *or* $C = (\leq\ n\, R\, C')$, *then* $depth(C) = 1 + depth(C')$.
- *If* $C = C_1 \sqcap \cdots \sqcap C_n$, *then* $depth(C) = \max(\{depth(C_i) | 1 \leq i \leq n\})$.
- *In all other cases,* $depth(C) = 0$.

$\square$

With these preparations, we will now define the least common subsuming ABox of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ as the most specific ABox (w.r.t. ABox subsumption) which subsumes $\mathcal{A}_1, \ldots, \mathcal{A}_n$.

Definition 10 (Least Common Subsuming ABox) *Let* $\mathcal{A}_1, \ldots, \mathcal{A}_n, n \geq 1$, *be ABoxes. Then we define the* least common subsuming ABox *(LCSA) of* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *as*

$$\begin{aligned} lcsa(\mathcal{A}_1, \ldots, \mathcal{A}_n) := \{\mathcal{A} \mid\ & \mathcal{A}_1 \sqsubseteq \mathcal{A} \wedge \cdots \wedge \mathcal{A}_n \sqsubseteq \mathcal{A} \wedge \\ & \forall \mathcal{A}' : \mathcal{A}_1 \sqsubseteq \mathcal{A}' \wedge \cdots \wedge \mathcal{A}_n \sqsubseteq \mathcal{A}' \Rightarrow \mathcal{A} \sqsubseteq \mathcal{A}'\}. \ \square \end{aligned}$$

From Definition 10 it follows that *lcsa* is an associative and commutative function and $lcsa(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is equivalent to $lcsa(\mathcal{A}_1, lcsa(\mathcal{A}_2, \ldots lcsa(\mathcal{A}_{n-1}, \mathcal{A}_n) \ldots))$. Therefore, we will restrict the attention to the problem of computing the LCSA of *two* ABoxes since the LCSA of $n > 2$ ABoxes can be obtained by $n - 1$ iterated applications of the binary LCSA operation. In addition, we can derive the following consequences.

Proposition 2 *Let* $\mathcal{A}_1, \mathcal{A}_2, \mathcal{B}_1, \ldots, \mathcal{B}_n$ *be ABoxes. Then*

(i) If $\mathcal{A}_2 \sqsubseteq \mathcal{A}_1$*, then* $lcsa(\mathcal{A}_1, \mathcal{A}_2) \equiv \mathcal{A}_1$.

(ii) $\forall \mathcal{B}, \mathcal{B}' \in lcsa(\mathcal{B}_1, \ldots, \mathcal{B}_n) : \mathcal{B} \equiv \mathcal{B}'$.

Proof. (i) is an obvious consequence of Definition 10. To see (ii), suppose $\mathcal{B}, \mathcal{B}' \in lcsa(\mathcal{B}_1, \ldots, \mathcal{B}_n)$ with $\mathcal{B} \not\equiv \mathcal{B}'$. Then, according to Proposition 1 (vi), $\mathcal{B} \cup \mathcal{B}'$ is a more specific subsumer of $\mathcal{B}_1, \ldots, \mathcal{B}_n$ than both $\mathcal{B}$ and $\mathcal{B}'$ and hence, $\mathcal{B}$ (resp. $\mathcal{B}'$) cannot be an LCSA of $\mathcal{B}_1, \ldots, \mathcal{B}_n$ which leads to a contradiction. □

Proposition 2 (ii) states a uniqueness property similar to the one for the LCS. If the LCSA is not empty, all pairs of its elements are equivalent. Therefore, if it is convenient, we will consider $lcsa(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ as a function which returns an ABox rather than a set of ABoxes in the following. Sometimes, the LCSA is given by the empty set of ABoxes. In this case we will say that LCSA is undefined. We will postpone the treatment of an undefined LCSA and first deal with the problem of how to compute the LCSA in the case that it exists.

Before starting our analysis, it will be convenient to define the LCS of $C_1, \ldots, C_n$ which are all either $\mathcal{ALENR}$ or $\mathcal{ALQ}$ concepts as the most specific concept which subsumes $C_1, \ldots, C_n$.

Definition 11 (Least Common Subsumer) *Let* $C_1, \ldots, C_n, n \geq 1$, *be all either* $\mathcal{ALENR}$ *or* $\mathcal{ALQ}$ *concepts. Then we define the set of* least common subsumers *of* $C_1, \ldots, C_n$ *as:*

$$\begin{aligned} lcs(C_1, \ldots, C_n) := \{E \mid\ & C_1 \sqsubseteq E \wedge \cdots \wedge C_n \sqsubseteq E \wedge \\ & \forall E' : C_1 \sqsubseteq E' \wedge \cdots \wedge C_n \sqsubseteq E' \Rightarrow E \sqsubseteq E'\}. \ \square \end{aligned}$$

In [6], we provide an algorithm for computing the LCS of $C_1, \ldots, C_n$. It will also prove helpful to introduce an operation for the most specific role of roles $R_1, \ldots, R_n$.

Definition 12 (Most Specific Role) *Let $R_1, \ldots, R_n$ be roles in either $\mathcal{ALENR}$ or $\mathcal{ALQ}$. Then we define the* most specific superrole *(MSS) of $R_1, \ldots, R_n$ as:*

$$\begin{aligned} mss(R_1, \ldots, R_n) := \{R \;\mid\; & R_1 \sqsubseteq R \wedge \cdots \wedge R_n \sqsubseteq R \wedge \\ & \forall R' : R_1 \sqsubseteq R' \wedge \cdots \wedge R_n \sqsubseteq R' \Rightarrow R \sqsubseteq R'\}. \end{aligned} \quad \square$$

From this definition we can derive a simple method for computing the MSS of a number of roles.

Proposition 3 *Let $R_1, \ldots, R_n$ be roles in $\mathcal{ALENR}$ with $R_i := R_{i1} \sqcap \cdots \sqcap R_{im_i}$ and $\mathfrak{R}_i := \{R_{i1}, \ldots, R_{im_i}\}$ for all $i \in \{1, \ldots, n\}$. Then*

$$mss(R_1, \ldots, R_n) := \begin{cases} \sqcap_{R \in \mathfrak{R}_1 \cap \cdots \cap \mathfrak{R}_n} R & \textit{if } \mathfrak{R}_1 \cap \cdots \cap \mathfrak{R}_n \neq \emptyset, \textit{ and} \\ \textit{undefined} & \textit{otherwise.} \end{cases} \quad \square$$

Obviously, for roles $R_1, \ldots, R_n$ in $\mathcal{ALQ}$, $mss(R_1, \ldots, R_n)$ is given by R_1 iff all pairs R_i and R_j are identical and undefined otherwise. This is true because there exists no role forming constructor in $\mathcal{ALQ}$. In the sequel, we will provide an algorithm for computing the LCSA of ABoxes $\mathcal{A}$ and $\mathcal{B}$. The idea is based on the following facts. Given $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$ where α and β are both either concept assertions or role assertions, we can compute a new assertion γ such that γ is the most specific assertion (w.r.t. the subsumption definition for assertions in Definition 5) which subsumes α and β. In general, such a γ may not exist. If it exists, it is convenient to transform $\mathcal{A}$ and $\mathcal{B}$ into a form such that γ can easily be determined by pairwise comparing *compatible* assertions to one another. Thereby, two assertions are compatible to each other if both assertions are either concept assertions which involve the same individual or role assertions which involve the same individuals, and the MSS of the involved roles is defined. In such a case, we will say that the new assertion γ emerges from α and β by application of the LCSA rules. The following proposition summarizes these observations.

Proposition 4 *Let $a : C$ and $a : D$ be concept assertions and $(a, b) : R$ and $(a, b) : S$ be role assertions such that $mss(R, S)$ is defined. Then:*

(i) $a : C \sqsubseteq a : lcs(C, D)$, $a : D \sqsubseteq a : lcs(C, D)$, and, for all assertions γ with $a : C \sqsubseteq \gamma$ and $a : D \sqsubseteq \gamma$, we have that $a : lcs(C, D) \sqsubseteq \gamma$.

(ii) $(a,b) : R \sqsubseteq (a,b) : mss(R,S)$*,* $(a,b) : S \sqsubseteq (a,b) : mss(R,S)$*, and, for all assertions* γ *with* $(a,b) : R \sqsubseteq \gamma$ *and* $(a,b) : S \sqsubseteq \gamma$*, we have that* $(a,b) : mss(R,S) \sqsubseteq \gamma$*.*

□

The two observations are immediate consequences of the definition of ABox subsumption (Definition 5), the definition of LCS (Definition 11) and the definition of MSS (Definition 12).

Following the line described above, we apply the LCSA rules to every pair of compatible assertions $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$ and obtain a set of new assertions forming an ABox which is equivalent to the LCSA of $\mathcal{A}$ and $\mathcal{B}$. Let us illustrate this idea by an example.

Example 1 *Let*

$$\begin{aligned} \mathcal{A} &:= \{a : A \sqcap A', (a,b) : (R \sqcap S), (c,d) : (R \sqcap S)\} \textit{ and} \\ \mathcal{B} &:= \{a : A \sqcap A'', (a,b) : (R \sqcap S'), b : A \sqcap A'\}. \end{aligned}$$

Then, for the compatible concept assertions $a : A \sqcap A' \in \mathcal{A}$ *and* $a : A \sqcap A'' \in \mathcal{B}$*, we set* $\gamma_1 := a : lcs(A \sqcap A', A \sqcap A'')$*. Furthermore, given the compatible role assertions* $(a,b) : (R \sqcap S) \in \mathcal{A}$ *and* $(a,b) : (R \sqcap S') \in \mathcal{B}$*, we set* $\gamma_2 := (a,b) : mss(R \sqcap S, R \sqcap S')$*. All other pairs of assertions are incompatible to each other. Hence, we yield the ABox* $\{\gamma_1, \gamma_2\} \equiv \{a : A, (a,b) : R\}$ *which is equivalent to* $lcsa(\mathcal{A}, \mathcal{B})$*.*

For the ABoxes $\mathcal{A}$ and $\mathcal{B}$ considered in Example 1, it is possible to compute $lcsa(\mathcal{A}, \mathcal{B})$ in the described way. However, as the following example shows, simply applying the LCSA rules does not solve the problem in general.

Example 2 *Let*

$$\begin{aligned} \mathcal{A} &:= \{a : \exists\, R.A\} \textit{ and} \\ \mathcal{B} &:= \{(a,b) : (R \sqcap S)\}. \end{aligned}$$

Then, applying the LCSA rules to $\mathcal{A}$ *and* $\mathcal{B}$ *yields the empty ABox. However, the LCSA of* $\mathcal{A}$ *and* $\mathcal{B}$ *is equivalent to the ABox* $\{a : \exists\, R.\top\}$ *which is strictly subsumed by* $\emptyset$*. This problem can be circumvented by adding the concept assertion* $a : \exists\,(R \sqcap S).\top$ *to* $\mathcal{B}$ *yielding* $\mathcal{B}'$*. Obviously, the new concept assertion is a logical consequence of* $\mathcal{B}$ *and hence, the transformation from* $\mathcal{B}$ *to* $\mathcal{B}'$ *is semantics-preserving. Now the LCSA of* $\mathcal{A}$ *and* $\mathcal{B}$ *can be determined by applying the LCSA rules to* $\mathcal{A}$ *and* $\mathcal{B}'$ *in the usual way.*

In Example 2, the concept appearing in the added assertion $a : \exists(R \sqcap S).\top$ is the most specific concept of the individual a w.r.t. $\mathcal{B}$. Generalizing the described idea, before applying the LCSA rules, we add to $\mathcal{A}$ ($\mathcal{B}$) the most specific concept of a (b) for all individuals a (b) occurring in $\mathcal{A}$ ($\mathcal{B}$).

Definition 13 *Let $\mathcal{A}$ be an ABox, $a \in Ind(\mathcal{A})$, and C a concept. Then we say that a* is an instance of C *iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{A}$.*

The notion of instances of concepts can now be used to define the most specific concept of an ABox individual.

Definition 14 (Most Specific Concept) *Let $\mathcal{L}$ be a description logic, $\mathcal{A}$ an ABox w.r.t. $\mathcal{L}$, and $a \in Ind(\mathcal{A})$. Then we define the* most specific concept *(MSC) of a w.r.t. $\mathcal{A}$ as*

$$\begin{aligned} msc_{\mathcal{A}}(a) := \{C \in \mathcal{L} \mid\ & a \text{ is an instance of } C \wedge \\ & \forall C' \in \mathcal{L} : a \text{ is an instance of } C' \Rightarrow C \sqsubseteq C'\}. \end{aligned}$$ □

From Definition 14 we can derive two important properties of the MSC. One of them refers to so called cyclic ABoxes which will be introduced now.

Definition 15 (Path and Cycle) *Let $\mathcal{A}$ be an ABox and $n \in \mathbb{N}$. Then we introduce the following terms:*

- *A subset $\mathcal{A}' \subseteq \mathcal{A}$ is called a* path of length n *of $\mathcal{A}$ iff $\mathcal{A}'$ is of the form $\{(a_0, a_1) : R_0, \dots, (a_{n-1}, a_n) : R_{n-1}\}$ where, for $i, j \in \{0, \dots, n\}$, $a_i \neq a_j$ holds whenever $i \neq j$.*

- *$\mathcal{A}' \subseteq \mathcal{A}$ is called a* cycle of length n *of $\mathcal{A}$ iff $\mathcal{A}'$ is of the form $\{(a_0, a_1) : R_0, \dots, (a_{n-2}, a_{n-1}) : R_{n-2}, (a_{n-1}, a_0) : R_{n-1}\}$. If $\mathcal{A}$ contains a cycle, we also say that $\mathcal{A}$ is* cyclic, *otherwise $\mathcal{A}$ is* non-cyclic.

- *$\mathcal{A}$ has* maximal cycle (path) length n *iff $\mathcal{A}$ contains a cycle (path) of length n and all other cycles (paths) in $\mathcal{A}$ are of length smaller than or equal to n.*

- *By default, we set the maximal cycle (path) length of $\mathcal{A}$ to 0 in case $\mathcal{A}$ is non-cyclic (has no path).*

□

Note that, according to Definition 15, loops are also considered as cycles. For example, the ABox $\{(a,a):R\}$ is cyclic with maximal cycle length of 1. For a given ABox $\mathcal{A}$, it is easy to compute $\mathcal{A}$'s maximal cycle and path length by an algorithm. We omit the technical details here. From Definitions 14 and 15 we can derive important properties of the MSC. It is easy to see that $msc_{\mathcal{A}}(a) = \emptyset$ if $\mathcal{A}$ is unsatisfiable. Consequently, for computing the MSC of an individual in $\mathcal{A}$, we expect $\mathcal{A}$ to be at least satisfiable. The following proposition shows however that this is not the only reason for the MSC to be the empty set.

Proposition 5 *Let $\mathcal{A}$ be an ABox and $a \in Ind(\mathcal{A})$. Then,*

(i) for all individuals $a \in Ind(\mathcal{A})$ and all $C, C' \in msc_{\mathcal{A}}(a)$, C and C' are equivalent, and

(ii) $msc_{\mathcal{A}}(a)$ does not exist iff $\mathcal{A}$ is unsatisfiable or there exist $a_1, \dots, a_m$, $a_{m+1}, \dots, a_{n-1} \in Ind(\mathcal{A})$ and $R_0, \dots, R_m, R_{m+1}, \dots, R_{n-1}$ such that $\{(a,a_1):R_0, \dots, (a_m,a_{m+1}):R_m, \dots, (a_{n-1},a_m):R_{n-1}\} \subseteq \mathcal{A}$.

Proof. We prove (i) by contradiction. If $\mathcal{A}$ is unsatisfiable, then $msc_{\mathcal{A}}(a) = \emptyset$ and nothing needs to be shown. Otherwise, let $C, C' \in msc_{\mathcal{A}}(a)$ and suppose $C \not\equiv C'$. Then, a is an instance of $C \sqcap C'$ and both $C \sqcap C' \sqsubset C$ and $C \sqcap C' \sqsubset C'$ hold, which is a contradiction to $C, C' \in msc_{\mathcal{A}}(a)$. Now we prove "$\Leftarrow$" of (ii) by contradiction. If $\mathcal{A}$ is unsatisfiable, it is obvious that $msc_{\mathcal{A}}(a)$ does not exist. Otherwise, let $\{(a,a_1):R_0, \dots, (a_m,a_{m+1}):R_m, \dots, (a_{n-1},a_m):R_{n-1}\} \subseteq \mathcal{A}$ and suppose $msc_{\mathcal{A}}(a)$ exists. Obviously, we have that $msc_{\mathcal{A}}(a_m)$ exists if $msc_{\mathcal{A}}(a)$ exists. We will now show that $msc_{\mathcal{A}}(a_m)$ does not exist. In order to simplify the proof, we define $b_i := a_{m+i}$ and $T_i := R_{m+i}$, for $i \in \{0, \dots, n-m-1\}$. Assume that there exists a concept $C \in msc_{\mathcal{A}}(b_0)$. Then we recursively define a series of concepts $C_i, i = 0,1,2,\dots$, as follows:

$$C_0 := msc_{\mathcal{A}}(b_0), \text{ and} \tag{1}$$

$$C_{i+1} := msc_{\mathcal{A}}(b_{(i \mod (n-m))}) \sqcap \exists\, T_{(i \mod (n-m))}.C_i. \tag{2}$$

The concepts C_i are well-defined since, for every individual $b_{(i \mod (n-m))}$ occurring in (1) and (2), $msc_{\mathcal{A}}(b_{(i \mod (n-m))})$ is defined (because there is a path from a to $b_{(i \mod (n-m))}$ and $msc_{\mathcal{A}}(a)$ is defined according to our assumption). The sets $C_i, i = 0,1,2,\dots$, are constructed in such a way that, for all $i \in \mathbb{N}$, a is an instance of C_i and $C_{i+1} \sqsubset C_i$. Therefore, there exists an $i \in \mathbb{N}$ such that b_0 is an instance of C_i and $depth(C_i) > depth(C)$. But this implies that $C \not\sqsubseteq C_i$ in contradiction to the assumption that $C \in msc_{\mathcal{A}}(b_0)$. Now we will prove "$\Rightarrow$" by contradiction. Suppose $msc_{\mathcal{A}}(a)$ does not exist and let $\mathcal{A}$ be a satisfiable ABox such that there exists no $\{(a, a_1) : R_0, \ldots, (a_m, a_{m+1}) : R_m, \ldots, (a_{n-1}, a_m) : R_{n-1}\} \subseteq \mathcal{A}$. Consequently, there are obviously only paths emerging from a. Let p be the length of the longest of these paths and let $\mathcal{A}'$ be the preprocessing complete[3] version of $\mathcal{A}$ in which no pair of identical role assertions is present. If $\mathcal{A}$ is an ABox w.r.t. $\mathcal{ALENR}$, then, for $b \in \mathit{Ind}(\mathcal{A}')$ and $i \in \mathbb{N}$, we recursively define a function $f(b, i)$ as follows:

$$\begin{aligned} f(b,0) &:= \top \sqcap \sqcap_{b:C \in \mathcal{A}'} C \text{ and} \\ f(b,i+1) &:= f(b,0) \sqcap \sqcap_{R \in \mathit{Roles}(\mathcal{A}')} (\top \sqcap (\geq\ |\{(b,c) : R \in \mathcal{A}\}|\ R)). \end{aligned}$$

From this definition it follows that, for all $i \in \{1, \ldots, p\}$, $f(a, i)$ is well-defined (since no cycles can be reached by any path emerging from a), a is an instance of $f(a, p)$, and the concept $f(a, p)$ is the most specific concept with this property. Hence, $f(a, p)$ is equivalent to $\mathit{msc}_{\mathcal{A}}(a)$ in contradiction to our assumption. In the case that $\mathcal{A}$ is an ABox w.r.t. $\mathcal{ALQ}$, we define

$$\begin{aligned} f(b,0) &:= \top \sqcap \sqcap_{b:C \in \mathcal{A}'} C \text{ and} \\ f(b,i+1) &:= f(b,0) \sqcap \sqcap_{R \in \mathit{Roles}(\mathcal{A}')} (\top \sqcap \sqcap_{\{(b,c_1):R_1, \ldots, (b,c_q):R_q \mid R_i = R\} \subseteq \mathcal{A}'} (\geq\ q\ R \\ &\qquad \mathit{lcs}(f(c_1, i), \ldots, f(c_q, i)))). \end{aligned}$$

Again, for all $i \in \{1, \ldots, p\}$, $f(a, i)$ is well-defined, a is an instance of $f(a, p)$, and the concept $f(a, p)$ is the most specific concept with this property. Hence, $f(a, p)$ is equivalent to $\mathit{msc}_{\mathcal{A}}(a)$, which again contradicts to our assumption. $\square$

Proposition 5 (i) states a similar uniqueness criteria for the MSC as for the LCSA for ABoxes. From the second part of the proposition it follows that, for an individual a, $\mathit{msc}_{\mathcal{A}}(a)$ does not exist if either $\mathcal{A}$ is unsatisfiable or there emerges some path from a which terminates in any of $\mathcal{A}$'s cycles. These properties of the MSC lead to the following two consequences. Due to (i), we will consider the MSC of an individual a occurring in $\mathcal{A}$ to be a concept rather than a set of concepts if $\mathit{msc}_{\mathcal{A}}(a) \neq \emptyset$. On the other hand, we consider $\mathit{msc}_{\mathcal{A}}(a)$ to be undefined if $\mathit{msc}_{\mathcal{A}}(a) = \emptyset$. As another consequence, the above described idea for computing the LCSA of ABoxes does not work in all cases. We will now illustrate with an example how to circumvent this problem.

[3]In the preprocessing complete version of an ABox $\mathcal{A}$, the concepts C which appear in concept assertions of the form $a : \exists\, R.C, a : \forall\, R.C, a : (\geq\ n\ R\ C)$, and $(\leq\ n\ R\ C)$ are spread over a's R-successors, and concept and role conjunctions are decomposed. The preprocessing completion will be formally defined in Definition 19.

Example 3 *Let*

$$\begin{aligned}\mathcal{A} &:= \{(a_0, a_1) : R_0, (a_1, a_2) : R_1, (a_2, a_0) : R_2\} \textit{ and}\\ \mathcal{B}_1 &:= \{(a_0, a_1) : R_0, (a_1, a_0) : R_1\}.\end{aligned}$$

Then both $\mathcal{A}$ and $\mathcal{B}_1$ are cyclic and $msc_{\mathcal{A}}(a)$ does not exist for all $a \in \{a_0, \ldots, a_2\}$ and $msc_{\mathcal{B}_1}(a)$ does not exist for all $b \in \{a_0, a_1\}$. Despite, $lcsa(\mathcal{A}, \mathcal{B}_1)$ is given by an ABox equivalent to $\{a_1 : \exists\, R_1.\top, (a_0, a_1) : R_0\}$. For determining $lcsa(\mathcal{A}, \mathcal{B}_1)$, we can "approximate" the MSC of each individual in $Ind(\mathcal{A}) \cup Ind(\mathcal{B}_1)$ up to a depth which is equal to the sum of the lengths of the longest cycle in $\mathcal{A}$ and the one of $\mathcal{B}$. Since the maximal length of $\mathcal{A}$'s cycles is 3 and the maximal length of $\mathcal{B}$'s cycles is 2, we define:

$$\begin{aligned}\mathcal{A}' &:= \{a_0 : \exists\, R_0.(\exists\, R_1.(\exists\, R_2.(\exists\, R_0.(\exists\, R_1.\top)))),\\ &\qquad a_1 : \exists\, R_1.(\exists\, R_2.(\exists\, R_0.(\exists\, R_1.(\exists\, R_2.\top)))),\\ &\qquad a_2 : \exists\, R_2.(\exists\, R_0.(\exists\, R_1.(\exists\, R_2.(\exists\, R_0.\top))))\} \cup \mathcal{A} \textit{ and}\\ \mathcal{B}_1' &:= \{a_0 : \exists\, R_0.(\exists\, R_1.(\exists\, R_0.(\exists\, R_1.(\exists\, R_0.\top)))),\\ &\qquad a_1 : \exists\, R_1.(\exists\, R_0.(\exists\, R_1.(\exists\, R_0.(\exists\, R_1.\top))))\} \cup \mathcal{B}.\end{aligned}$$

Since the additional concept assertions in $\mathcal{A}'$ ($\mathcal{B}_1'$) are logical consequences of $\mathcal{A}$ ($\mathcal{B}_1$), the transformation is semantics-preserving and $\mathcal{A}' \equiv \mathcal{A}$ ($\mathcal{B}_1' \equiv \mathcal{B}_1$) is guaranteed. Now $lcsa(\mathcal{A}, \mathcal{B}_1)$ can be computed by applying the LCSA rules to $\mathcal{A}'$ and $\mathcal{B}_1'$ in the usual way.

Example 3 shows that, if the LCSA of two satisfiable ABoxes exists, but the MSC of an individual a does not exist, the MSC can be replaced by an MSC approximation. More specifically, given an ABox $\mathcal{A}$ ($\mathcal{B}$) with maximal cycle length n (m), maximal path length p (q), and r (s) as the maximum concept depth over all concepts occurring in concept assertions in $\mathcal{A}$ ($\mathcal{B}$), for all individuals in $\mathcal{A}$ ($\mathcal{B}$), we add $a : C$ to $\mathcal{A}$ ($\mathcal{B}$) where C is the most specific concept of depth $n + m + p + q + r + s$ of which a is an instance. However, the following example shows that there is another exception to be considered in the LCSA determination.

Example 4 (Example 3 continued) *Let $\mathcal{A}$ be defined as in Example 3 and*

$$\mathcal{B}_2 := \{a_0 : \exists\, R_0.(\exists\, R_1.(\exists\, R_2.(\exists\, R_0.\top)))\}.$$

Again, the sum of the lengths of the maximal cycle length of both ABoxes is 3 (because $\mathcal{A}$'s maximal cycle length is 3 and $\mathcal{B}_2$ is non-cyclic) and thus, we *can define*

$$\begin{aligned}\mathcal{A}' \;:=\; & \{a_0 : \exists\, R_0.(\exists\, R_1.(\exists\, R_2.\top)),\\ & a_1 : \exists\, R_1.(\exists\, R_2.(\exists\, R_0.\top)),\\ & a_2 : \exists\, R_2.(\exists\, R_0.(\exists\, R_1.\top))\} \cup \mathcal{A}\end{aligned}$$

and apply the LCSA rules to $\mathcal{A}'$ and $\mathcal{B}_2$ yielding an ABox equivalent to

$$\mathcal{C} := \{a_0 : \exists\, R_0.(\exists\, R_1.(\exists\, R_2.\top))\}.$$

The ABox $\mathcal{C}$ is a subsumer of both $\mathcal{A}$ and $\mathcal{B}_2$, but it is not the most specific one. The LCSA of $\mathcal{A}$ and $\mathcal{B}_2$ is given by

$$lcsa(\mathcal{A}, \mathcal{B}_2) \equiv \mathcal{B}_2$$

where $\mathcal{B}_2 \sqsubset \mathcal{C}$. The reason for this undesired result is due to the fact that the only concept assertion in $\mathcal{B}_2$ involves a concept of depth four which is larger than the sum of the lengths of the longest cycle in $\mathcal{A}$ and the one in $\mathcal{B}_2$, and this has an effect on determining the LCSA. If we approximate $msc_{\mathcal{A}}(a_0)$, $msc_{\mathcal{A}}(a_1)$, and $msc_{\mathcal{A}}(a_2)$ up to a concept of depth four, we get

$$\begin{aligned}\mathcal{A}'' \;:=\; & \{a_0 : \exists\, R_0.(\exists\, R_1.(\exists\, R_2.(\exists\, R_0.\top))),\\ & a_1 : \exists\, R_1.(\exists\, R_2.(\exists\, R_0.(\exists\, R_2.\top))),\\ & a_2 : \exists\, R_2.(\exists\, R_0.(\exists\, R_1.(\exists\, R_2.\top)))\} \cup \mathcal{A}.\end{aligned}$$

As in Example 3, semantics preservation is guaranteed by the transformation and thus, $\mathcal{A}'' \equiv \mathcal{A}$ holds. Now we can easily compute $lcsa(\mathcal{A}, \mathcal{B}_2)$ by applying the known LCSA rules to $\mathcal{A}''$ and $\mathcal{B}_2$ yielding an ABox equivalent to $\mathcal{B}_2$ as desired.

The examples suggest the following algorithm for computing the LCSA of two ABoxes $\mathcal{A}$ and $\mathcal{B}$. For all $a \in Ind(\mathcal{A})$ and $b \in Ind(\mathcal{B})$, we add to $\mathcal{A}$ $\mathcal{B}$ the additional concept assertions $a : msc_{\mathcal{A}}(a)$ $[b : msc_{\mathcal{B}}(b)]$ if $a : msc_{\mathcal{A}}(a)$ $[b : msc_{\mathcal{B}}(b)]$ exists. Otherwise, we add to $\mathcal{A}$ ($\mathcal{B}$) the MSC approximation of a (b) described in Example 4. As argued before, this transformation is semantics-preserving and thus, equivalence between $\mathcal{A}$ ($\mathcal{B}$) and its transformed ABox $\mathcal{A}'$ ($\mathcal{B}'$) is guaranteed. Finally, we apply the LCSA rules to $\mathcal{A}'$ and $\mathcal{B}'$ obtaining the LCSA of $\mathcal{A}$ and $\mathcal{B}$. We will now formalize our idea to which degree the MSC needs to be approximated.

Definition 16 (ABox depth) *Let $\mathcal{A}$ be an ABox, $c_{\mathcal{A}}$ be $\mathcal{A}$'s maximal cycle length, and $p_{\mathcal{A}}$ the length of the longest path in $\mathcal{A}$. Then we define the* depth *of $\mathcal{A}$ as:*

$$depth(\mathcal{A}) := c_{\mathcal{A}} + p_{\mathcal{A}} + \max(\{depth(C) | a : C \in \mathcal{A}\}). \qquad \square$$

Intuitively, the depth of an ABox $\mathcal{A}$ is given by the sum of the following numerical values: its maximal cycle length ($c_{\mathcal{A}}$), the length of the longest path occurring in $\mathcal{A}$ ($p_{\mathcal{A}}$), and the value of the largest depth over the concepts involved in $\mathcal{A}$'s concept assertions.

Definition 17 (Most Specific Concept Approximation) *Let $\mathcal{L}$ be either the description logic $\mathcal{ALENR}$ or $\mathcal{ALQ}$, $\mathcal{A}$ an ABox w.r.t. $\mathcal{L}$, $a \in Ind(\mathcal{A})$, and $d \in I\!N$. Then we define the* most specific concept approximation *(MSC approximation) of a w.r.t. $\mathcal{A}$ and d as*

$$\begin{aligned} \textit{msc-approx}_{\mathcal{A},d}(a) := \{C \in \mathcal{L} \quad | \quad & \textit{depth}(C) \leq d \wedge a \textit{ is an instance of } C \wedge \\ & \forall C' \in \mathcal{L} : \textit{depth}(C') \leq \textit{depth}(C) \wedge a \textit{ is} \\ & \textit{an instance of } C' \Rightarrow C \sqsubseteq C'\}. \end{aligned} \qquad \square$$

Intuitively, the MSC approximation of an individual a occurring in an ABox $\mathcal{A}$ is the set of most specific concepts of which a is an instance, where the depth of the concepts is less than or equal to d. Since all pairs of concepts of $\textit{msc-approx}_{\mathcal{A},d}(a)$ are equivalent as in the case of the MSC, we will consider $\textit{msc-approx}_{\mathcal{A},d}(a)$ as a concept rather than a set of concepts. Obviously, for $d = \textit{depth}(\mathcal{A})$, $\textit{msc-approx}_{\mathcal{A},d}(a)$ coincides with $\textit{msc}_{\mathcal{A}}(a)$ if $\textit{msc}_{\mathcal{A}}(a)$ exists. The following example shows that it is convenient to put an ABox into a specific form before computing the MSC or the MSC approximation of its individuals.

Example 5 *Let*

$$\mathcal{A} := \{(a, b) : R, \forall\, R.A, b : B\}.$$

Then, $\textit{msc}_{\mathcal{A}}(b)$ is equivalent to $A \sqcap B$. This result follows since b is an R-successor of a in $\mathcal{A}$ and, due to $a : \forall\, R.A$, b has the property $A \sqcap B$. If we add the concept assertion $b : A$ to $\mathcal{A}$, $\textit{msc}_{\mathcal{A}}(b)$ can easily be determined.

Example 5 shows that before applying the procedure for computing the MSC of an individual $a \in Ind(\mathcal{A})$, we have to transform $\mathcal{A}$ into a form such that relevant information regarding a is "transported" to a. We will now present a simple method for transforming an ABox into an equivalent ABox all of whose individuals have this property. The idea is to introduce preprocessing rules similar to the rules presented in [4] for obtaining a preprocessing complete ABox.

Definition 18 (Preprocessing Rules) *For an ABox $\mathcal{A}$ w.r.t. either $\mathcal{ALENR}$ or $\mathcal{ALQ}$, we define the* preprocessing rules *as follows:*

1. $\mathcal{A} \rightarrow_{\sqcap} \{a : C_1, a : C_2\} \cup \mathcal{A}$ *if*
   - *(i)* $a : C_1 \sqcap C_2 \in \mathcal{A}$ *and*
   - *(ii)* $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$,
2. $\mathcal{A} \rightarrow_{\exists} \{b : D\} \cup \mathcal{A}$ *if*
   - *(i)* $\{a : \exists\, R.C, (a, b) : R\} \subseteq \mathcal{A}$,
   - *(ii) neither* $b : C \in \mathcal{A}$ *nor there exists an* $E \in \mathcal{ALENR}$ *such that* $b : E \in \mathcal{A}$ *and* $E \equiv \neg C$,
   - *(iii)* $\mathcal{A} \cup \{b : D\}$ *is satisfiable, and*
   - *(iv)* $D = C$ *or* $D = E$ *if there exists an* $E \in \mathcal{ALENR}$ *such that* $E \equiv \neg C$,
3. $\mathcal{A} \rightarrow_{\forall} \{b : C\} \cup \mathcal{A}$
   - *(i) if* $\{a : \forall\, R.C, (a, b) : R\} \subseteq \mathcal{A}$ *and*
   - *(ii)* $b : C \notin \mathcal{A}$,
4. $\mathcal{A} \rightarrow_{\geq} \{b : D\} \cup \mathcal{A}$
   - *(i)* $\{a : (\geq\ n\, R\, C), (a, b) : R\} \subseteq \mathcal{A}$,
   - *(ii) neither* $b : C \in \mathcal{A}$ *nor there exists an* $E \in \mathcal{ALQ}$ *such that* $b : E \in \mathcal{A}$ *and* $E \equiv \neg C$,
   - *(iii)* $\mathcal{A} \cup \{b : D\}$ *is satisfiable, and*
   - *(iv)* $D = C$ *or* $D = E$ *if there exists an* $E \in \mathcal{ALQ}$ *such that* $E \equiv \neg C$,
5. $\mathcal{A} \rightarrow_{\leq} \{b : D\} \cup \mathcal{A}$
   - *(i)* $\{a : (\leq\ n\, R\, C), (a, b) : R\} \subseteq \mathcal{A}$,
   - *(ii) neither* $b : C \in \mathcal{A}$ *nor there exists an* $E \in \mathcal{ALQ}$ *such that* $b : E \in \mathcal{A}$ *and* $E \equiv \neg C$,
   - *(iii)* $\mathcal{A} \cup \{b : D\}$ *is satisfiable, and*
   - *(iv)* $D = C$ *or* $D = E$ *if there exists an* $E \in \mathcal{ALQ}$ *such that* $E \equiv \neg C$, *and*
6. $\mathcal{A} \rightarrow_{R\sqcap} \{(a, b) : R, (a, b) : S\} \cup \mathcal{A}$ *if*
   - *(i)* $(a, b) : (R \sqcap S) \in \mathcal{A}$ *and*
   - *(ii)* $\{(a, b) : R, (a, b) : S\} \not\subseteq \mathcal{A}$.

□

Definition 19 *Let $\mathcal{A}$ be an ABox and let $\mathbf{A} := \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be the set of ABoxes emerging from $\mathcal{A}$ by exhaustive applications of the preprocessing rules in Definition 18. Then we define $\mathcal{A}$'s* preprocessing completion *as*

$$\mathcal{A}' := \bigcap_{\mathcal{B} \in \mathbf{A}} \mathcal{B}.$$

□

Let us discuss the preprocessing rules and the definition of a preprocessing completion. The $\rightarrow_{\sqcap}$-rule decomposes concept assertions involving concept conjunctions in the usual way. In case, a concept assertion of the form $a : \exists\, R.C$ is present and a has an R-successor b in $\mathcal{A}$, the $\rightarrow_{\exists}$-rule non-deterministically checks whether $b : C$ and/or a concept assertion equivalent to $b : \neg C$ can be added to $\mathcal{A}$ while preserving satisfiability of $\mathcal{A}$. If only $\mathcal{A} \cup \{b : C\}$ ($\mathcal{A} \cup \{b : \neg C\}$) is satisfiable, $b : C$ (a concept assertion equivalent to $b : \neg C$) will be included in $\mathcal{A}$ reflecting the idea that the property C ($\neg C$) holds for b. Otherwise, if both $\mathcal{A} \cup \{b : C\}$ and $\mathcal{A} \cup \{b : \neg C\}$ are satisfiable, due to the intersection operation in the definition of $\mathcal{A}'$, $\mathcal{A}$'s preprocessing completion does neither include $b : C$ nor a concept assertion equivalent to $b : \neg C$ reflecting the idea that there exists at least one model of $\mathcal{A}$ in which $b : C$ holds and at least one model of $\mathcal{A}$ in which $b : \neg C$ holds. In principal, the $\rightarrow_{\geq}$- and $\rightarrow_{\leq}$-rules work in the same way. The $\rightarrow_{\forall}$-rule adds $b : C$ to $\mathcal{A}$ if $a : \forall\, R.C \in \mathcal{A}$ and b is an R-successor of a in $\mathcal{A}$ since in this case b is obviously an instance of C. Finally, the $\rightarrow_{R\sqcap}$-rule decomposes role assertions involving role conjunctions (in $\mathcal{ALENR}$).

Proposition 6 *Let $\mathcal{A}$ be an ABox and $\mathcal{A}'$ be $\mathcal{A}$'s preprocessing completion. Then*

$\mathcal{A}$ is equivalent to $\mathcal{A}'$.

Proof. Let $\mathcal{A}$ be an ABox and $\mathcal{C}$ and $\mathcal{C}'$ be ABoxes which emerge from $\mathcal{A}$ by application of one of the preprocessing rules. Furthermore, let $\mathbf{B}$ be the set of all ABoxes which emerge from $\mathcal{A}$ by exhaustive applications of the preprocessing rules. Then, $\mathcal{A} \subset \mathcal{C}$ and $\mathcal{A} \subset \mathcal{C}'$ holds, which implies $\mathcal{A} \sqsubseteq \mathcal{C}$ and $\mathcal{A} \sqsubseteq \mathcal{C}'$. By induction on the length of rule applications it can easily be verified that, for all $\mathcal{B} \in \mathbf{B}$, $\mathcal{A} \sqsubseteq \mathcal{B}$ holds which implies $\mathcal{A} \sqsubseteq \mathcal{A}'$. In order to prove $\mathcal{A}' \sqsubseteq \mathcal{A}$, we first observe that an application of the $\rightarrow_{\forall}$-, $\rightarrow_{\sqcap}$, or $\rightarrow_{R\sqcap}$-rule is semantics-preserving. If, due to an application of any of the other preprocessing rules, a concept assertion $b : C$ ($b : E$ with $E \equiv \neg C$) is added to $\mathcal{A}$ and $\mathcal{A} \cup \{b : E\}$ ($\mathcal{A} \cup \{b : C\}$) is unsatisfiable, then $b : C$

$(b : E)$ is a logical consequence of $\mathcal{A}$. Hence, for all $\mathcal{B} \in \mathbf{B}$, we have that $a : C \in \mathcal{B}$ $(a : E \in \mathcal{B})$. But this implies that $a : C \in \mathcal{A}'$ $(a : E \in \mathcal{A}')$. If both $\mathcal{A} \cup \{b : C\}$ and $\mathcal{A} \cup \{b : E\}$ are satisfiable, then there exists $\mathbf{B}' \subseteq \mathbf{B}$ such that, for all $\mathcal{B} \in \mathbf{B}'$, we have that $b : C \in \mathcal{B}'$ and there exists $\mathbf{B}'' \subseteq \mathbf{B}$ such that, for all $\mathcal{B} \in \mathbf{B}''$, $b : E \in \mathcal{B}''$ holds. But then neither $b : C \in \mathcal{A}'$ nor $b : E \in \mathcal{A}'$ since neither $b : C \in \cup_{\mathcal{B}\in\mathbf{B}}\mathcal{B}$ nor $b : E \in \cup_{\mathcal{B}\in\mathbf{B}}\mathcal{B}$. Consequently, we have that, for all $\alpha \in \mathcal{A}' \setminus \mathcal{A}$, α is a logical consequence of $\mathcal{A}$, which implies $\mathcal{A}' \sqsubseteq \mathcal{A}$, and the claim is proved.

□

Proposition 6 guarantees that the preprocessing operation is semantics-preserving. In the sequel, we will only consider the preprocessing completions of ABoxes. Such an ABox will be called a *preprocessing complete* ABox. With these preparations, we can now state an algorithm for computing the MSC, respectively, the MSC approximation.

For a preprocessing complete ABox $\mathcal{A}$ w.r.t. either $\mathcal{ALENR}$ or $\mathcal{ALQ}$ and an individual $a \in \mathit{Ind}(\mathcal{A})$, the invocation *compute-msc*$(a, \mathit{depth}(\mathcal{A}), \mathcal{A})$ computes *msc-approx*$_{\mathcal{A}}(a)$ where $\mathcal{A}$ is assumed to be in a form such that no pair of identical role assertions is present. If an ABox w.r.t. $\mathcal{ALENR}$ is present, the algorithm computes the existential and universal role quantifications and $\geq$-restrictions following from the constraints imposed on a due to the role assertions in $\mathcal{A}$. Otherwise, if $\mathcal{A}$ is an ABox w.r.t. $\mathcal{ALQ}$, the qualified $\geq$-restrictions following from the constraints imposed on a are computed. In addition, the algorithm considers the concepts occurring in concept assertions involving a.

Theorem 4 *Let $\mathcal{A} \neq \emptyset$ be a satisfiable and preprocessing complete ABox in which no pair of identical role assertions appears, $a \in \mathit{Ind}(\mathcal{A})$, and $d \in \mathbb{N}$. Then,* compute-msc$(a, d, \mathcal{A})$ *terminates and returns a concept equivalent to msc-approx$_{\mathcal{A},d}(a)$.*

Proof. Termination of the algorithm can easily be verified. The second parameter d is a fixed natural number which is not modified anywhere in the function. Moreover, *compute-msc* is only recursively invoked with $d - 1$. As a consequence, either the computation terminates with $d > 0$ or there exists an invocation with $d = 0$. But then the first **then**-clause is processed and there are no further invocations, which guarantees termination. We show the remainder of the claim by induction on d. If $d = 0$, $\mathcal{A}$ is non-cyclic and *compute-msc*$(a, d, \mathcal{A})$ returns $\top \sqcap \sqcap_{a:C\in\mathcal{A}} C$ which is equivalent to *msc*$_{\mathcal{A}}(a)$ even in the case that there are no concept assertions involving a. Now assume $d > 0$. Then, we add to $\mathbf{C}$ the concepts imposed by concept assertions involving a. If there exists no role assertion $(a, b) : R \in \mathcal{A}$, $\mathbf{C}$

Algorithm 2 compute-msc$(a, d, \mathcal{A})$

```
if $d = 0$ then
  $\top \sqcap \sqcap_{a:C \in \mathcal{A}} C$
else
  $\mathbf{C} := \{\textit{compute-msc}(a, 0, \mathcal{A})\}$;
  $\mathbf{A} := \{(a, b) : R | b \in \textit{Ind}(\mathcal{A}) \wedge R \in \textit{Roles}(\mathcal{A})\}$;
  if $\mathcal{A}$ is an ABox w.r.t. $\mathcal{ALENR}$ then
    $\mathbf{C} := \mathbf{C} \cup \{\exists R.(\textit{compute-msc}(b, d-1, \mathcal{A})) | (a, b) : R \in \mathbf{A}\}$;
    for all $\mathbf{A}' \subseteq \mathbf{A}$ do
      // $\mathbf{A}' = \{(a, b_1) : R_1, \ldots, (a, b_n) : R_n\}$
      if $\textit{mss}(R_1, \ldots, R_n)$ is defined then
        $\mathbf{C} := \mathbf{C} \cup \{(\geq \ |\mathbf{A}'| \ \textit{mss}(R_1, \ldots, R_n))\}$
      end if
    end for
    $\mathbf{R} := \{R | (a, b) : R \in \mathbf{A}\}$;
    for all $\mathbf{R}' \subseteq \mathbf{R}$ do
      // $\mathbf{R}' = \{R_1, \ldots, R_n\}$;
      $\mathbf{S} := \{b \in \textit{Ind}(\mathcal{A}) | (a, b) : R \wedge R \in \mathbf{R}'\}$
      // $\mathbf{S} = \{b_1, \ldots, b_m\}$;
      if $\textit{mss}(R_1, \ldots, R_n)$ is defined and there exists $a : (\leq \ m \ R) \in \mathcal{A}$
      such that $R_1 \sqsubseteq R \wedge \cdots \wedge R_n \sqsubseteq R$ then
        $\mathbf{C} := \mathbf{C} \cup \{\forall \ \textit{mss}(R_1, \ldots, R_n).\textit{lcs}(\textit{compute-msc}(b_1, d -$
        $1, \mathcal{A}), \ldots, \textit{compute-msc}(b_m, d-1, \mathcal{A}))\}$;
      end if
    end for
  else
    // $\mathcal{A}$ is an ABox w.r.t. $\mathcal{ALQ}$
    for all $\mathbf{A}' \subseteq \mathbf{A}$ do
      // $\mathbf{A}' = \{(a, b_1) : R_1, \ldots, (a, b_n) : R_n\}$
      if $\textit{mss}(R_1, \ldots, R_n)$ is defined then
        $\mathbf{C} := \mathbf{C} \cup \{(\geq \ |\mathbf{A}'| \ R_1$
        $\textit{lcs}(\textit{compute-msc}(b_1, d-1, \mathcal{A}), \ldots, \textit{compute-msc}(b_n, d-1, \mathcal{A})))\}$
      end if
    end for
  end if
  $\sqcap_{C \in \mathbf{C}} C$
end if
```

is left unchanged and we correctly return $msc_{\mathcal{A}}(a)$ according to the initial case. Otherwise, we collect in $\mathbf{A}$ the set of role assertions which involve any role successor of a in $\mathcal{A}$. Suppose $\mathcal{A}$ is an ABox w.r.t. $\mathcal{ALENR}$. Then, by induction hypothesis, the invocation *compute-msc*$(b, d-1, \mathcal{A})$ (yielding C_b) is equivalent to *msc-approx*$_{\mathcal{A},d}(b)$. Now let $\mathbf{A}' = \{(a, b_1) : R_1, \ldots, (a, b_n) : R_n\}$ be a subset of $\mathbf{A}$. If $mss(R_1, \ldots, R_n)$ is defined, then a has obviously at least $|\mathcal{A}'|$ R-successors in $\mathcal{A}'$ (and thus in $\mathcal{A}$) since $\mathcal{A}'$ does not include any pair of identical role assertions and $\mathcal{A}$ is in preprocessing complete form. Hence, we add the $\geq$-restriction $(\geq |\mathbf{A}'|\, mss(R_1, \ldots, R_n))$ to $\mathbf{C}$. Now we collect into $\mathbf{R}$ the set of roles R for which there exists some R-successor of a in $\mathcal{A}$. Let $\mathbf{R}' = \{R_1, \ldots, R_n\}$ be a subset of $\mathbf{R}$ and $\mathbf{S} = \{b_1, \ldots, b_m\}$ the set of R-successors of a in $\mathcal{A}$ for all $R \in \mathbf{R}'$. If $mss(R_1, \ldots, R_n)$ is defined, $R_1 \sqsubseteq R \wedge \cdots \wedge R_n \sqsubseteq R$ holds, and there exists a concept assertion of the form $a : (\leq\ m\ R)$ in $\mathcal{A}$, then obviously all successors of a in the role $mss(R_1, \ldots, R_n)$ must have at least property *lcs*(*msc-approx*$_{\mathcal{A},d-1}(b_1), \ldots,$ *msc-approx*$_{\mathcal{A},d-1}(b_m)$). The relation $|\mathbf{S}| > m$ cannot hold because $\mathcal{A}$ would be unsatisfiable in this case. Since, for all b_i, *compute-msc*$(b_i, d-1, \mathcal{A})$ is equivalent to *msc-approx*$_{\mathcal{A},d-1}(b_i)$ holds by induction hypothesis, we add to $\mathbf{C}$ the universal role quantification $\forall\, mss(R_1, \ldots, R_n)$.*lcs*(*compute-msc*$(b_1, d-1, \mathcal{A}), \ldots,$ *compute-msc*$(b_m, d-1, \mathcal{A})$). If $\mathcal{A}$ is an ABox w.r.t. $\mathcal{ALQ}$, let $\mathbf{A}' = \{(a, b_1) : R_1, \ldots, (a, b_n) : R_n\}$ be a subset of $\mathbf{A}$. According to the induction hypothesis, the invocation *compute-msc*$(b_i, d-1, \mathcal{A})$ (yielding C_{b_i}) returns a concept equivalent to *msc-approx*$_{\mathcal{A},d}(b_i)$ for all $i \in \{1, \ldots, n\}$. Hence, a has at least $|\mathcal{A}'|$ successors in the role $mss(R_1, \ldots, R_n)$ which have property *lcs*(*msc-approx*$_{\mathcal{A},d-1}(b_1)$, $\ldots,$ *msc-approx*$_{\mathcal{A},d-1}(b_m)$). Thereby, if $mss(R_1, \ldots, R_n)$ is defined, it is equivalent to R_1 in $\mathcal{ALQ}$. Consequently, we add the qualified $\geq$-restriction $(\geq |\mathbf{A}'|\, R_1\, C_{b_i})$ to $\mathbf{C}$. Eventually, the conjunction of all concepts included in $\mathbf{C}$ is returned which is equivalent to $msc_{\mathcal{A}}(a)$. □

Example 6 *Let us compute the MSC of a which appears in the satisfiable and preprocessing complete ABox w.r.t.* $\mathcal{ALENR}$

$$\mathcal{A} := \{a : (\leq\ 2\,R) \sqcap \exists\, R.B, (a, b) : R, (a, c) : R, b : A, b : B, c : A, c : C\}.$$

Then, $depth(\mathcal{A}) = 1$ *and* compute-msc *is invoked with the parameters* a, 1, *and* $\mathcal{A}$. *The recursive invocation* compute-msc$(a, 0, \mathcal{A})$ *yields* $\mathbf{C} = \{(\leq 2R) \sqcap \exists\, R.B\}$. *Furthermore, we get* $\mathbf{A} = \{(a, b) : R, (a, c) : R\}$ *and the recursive invocations* compute-msc$(b, 0, \mathcal{A})$ *and* compute-msc$(c, 0, \mathcal{A})$ *yield* $A \sqcap B$ *and* $A \sqcap C$*, respectively, and* $\exists\, R.(A \sqcap B)$ *and* $\exists\, R.(A \sqcap C)$ *are added to* $\mathbf{C}$. *The following* **for**-*loop adds* $(\geq 1\,R)$ *and* $(\geq 2\,R)$ *to* $\mathbf{C}$. *Then, for* $\mathbf{R} = \mathbf{R}' = \{R\}$ *and* $\mathbf{S} = \{b, c\}$*, the subsequent* **for**-*loop adds* $\forall mss(R, R).lcs(A \sqcap B, A \sqcap C) \equiv$

$\forall\, R.A$ *to* $\mathbf{C}$*. Finally, we return the conjunction of concepts in* $\mathbf{C}$*, which is equivalent to* $(\leq\ 2\ R) \sqcap \exists\, R.(A \sqcap B) \sqcap \exists\, R.(A \sqcap C) \sqcap (\geq\ 2\ R) \sqcap \forall\, R.A$ *as desired.*

We will now introduce the MSC-completion of an ABox.

Definition 20 *Let $\mathcal{A}$ and $\mathcal{B}$ be satisfiable ABoxes. Then we define $\mathcal{A}$'s* MSC-completion *w.r.t. $\mathcal{B}$ as*

$$\hat{\mathcal{A}}_{\mathcal{B}} := \{a : \textit{msc-approx}_{\mathcal{A},\max(\{\textit{depth}(\mathcal{A}),\textit{depth}(\mathcal{B})\})}(a) | a \in \textit{Ind}(\mathcal{A})\} \cup \mathcal{A}. \qquad \square$$

Intuitively, an MSC-completion $\hat{\mathcal{A}}_{\mathcal{B}}$ of an ABox $\mathcal{A}$ w.r.t. $\mathcal{B}$ emerges from $\mathcal{A}$ by adding the MSC approximation of every individual $a \in \textit{Ind}(\mathcal{A})$ up to a number which is given by the maximum of the depths of $\mathcal{A}$ and $\mathcal{B}$.

We will now get back to the problem that the LCSA of ABoxes does not always exist.

Example 7 *Let*

$$\begin{aligned} \mathcal{A} &:= \{(a,b) : R, (b,a) : R\} \text{ and} \\ \mathcal{B} &:= \{(a,a) : R\}. \end{aligned}$$

Then, $lcsa(\mathcal{A},\mathcal{B})$ does not exist. The reason for this is that, for each ABox $\mathcal{C}$, which subsumes both $\mathcal{A}$ and $\mathcal{B}$, there exists an ABox $\mathcal{C}'$ with $\mathcal{C}' \sqsubset \mathcal{C}$. More specifically, if we recursively define a series of concepts C_i, $i = 0, 1, 2, 3, \ldots,$ as

$$\begin{aligned} C_1 &:= \exists\, R.\top \text{ and} \\ C_{i+1} &:= \exists\, R.(C_i), \end{aligned}$$

then it is easy to see that, for all $i \in I\!N \cup \{0\}$, $\mathcal{A} \sqsubseteq \{a : C_i\}$, $\mathcal{B} \sqsubseteq \{a : C_i\}$, and $\{a : C_{i+1}\} \sqsubset \{a : C_i\}$.

Informally speaking, the reason that the LCSA does not exist in Example 7 is given by the fact that all pairs of cyles $\mathcal{C}, \mathcal{D}$ with $\mathcal{C} \subseteq \mathcal{A}$ and $\mathcal{D} \subseteq \mathcal{B}$ which involve at least one common individual, have the following properties:

- the sequence of individuals induced by $\mathcal{C}$ is different from the one induced by $\mathcal{D}$, and
- the MSSs of the role pairs corresponding to each pair of corresponding sequence individuals is defined.

These properties of two ABoxes form a set of necessary and sufficient conditions for the (non-) existence of the LCSA.

Proposition 7 *Let $\mathcal{A}$ and $\mathcal{B}$ be satisfiable ABoxes and $\mathbf{C}$ and $\mathbf{D}$ be the set of $\mathcal{A}$'s and $\mathcal{B}$'s cycles, respectively. Then, $lcsa(\mathcal{A},\mathcal{B})$ does not exist iff $Ind(\mathcal{A}) \cap Ind(\mathcal{B}) \neq \emptyset$ and there exist $\mathcal{C} \in \mathbf{C}$ of the form $\{(a_0,a_1):R_0,\ldots,(a_{n-1},a_0):R_{n-1}\}$ and $\mathcal{D} \in \mathbf{D}$ of the form $\{(a_0,b_1):S_0,\ldots,(b_{m-1},a_0):S_{m-1}\}$ such that the following conditions hold (assuming $m \leq n$ without loss of generality):*

(i) there exists an $i \in \mathbb{N}$ such that $a_{(i \bmod n)} \neq b_{(i \bmod m)}$, and

(ii) $mss(R_{(i \bmod n)}, S_{(i \bmod m)})$ is defined for all $i \in \mathbb{N} \cup \{0\}$.

Proof. We first prove "$\Leftarrow$" by contradiction. Suppose, $\mathcal{C} \in lcsa(\mathcal{A},\mathcal{B})$ exists with $d = depth(\mathcal{C})$ and there are cycles $\mathcal{C} = \{(a_0,a_1):R_0,\ldots,(a_{n-1},a_0):R_{n-1}\}$ in $\mathcal{A}$ and $\mathcal{D} = \{(a_0,b_1):S_0,\ldots,(b_{m-1},a_0):S_{m-1}\}$ in $\mathcal{B}$ such that the conditions (i) and (ii) hold. Then, we create a sequence of ABoxes $\mathcal{C}_1,\mathcal{C}_2,\mathcal{C}_3,\ldots$ such that, for all $i \in \mathbb{N} \cup \{0\}$, $\mathcal{A} \sqsubseteq \mathcal{C}_i$ and $\mathcal{B} \sqsubseteq \mathcal{C}_i$ hold. We recursively define concepts

$$\begin{aligned} C_0 &:= \exists\, mss(R_0,S_0).\top \text{ and} \\ C_{i+1} &:= \exists\, mss(R_i, S_{(i \bmod m)}).(C_i). \end{aligned}$$

Note that, for all $i \in \mathbb{N} \cup \{0\}$, the concept C_i is well-defined since, due to condition (ii), the MSSs in C_i all exist. Now let $\mathcal{C}_i := \{a_0 : C_i\}$ and observe that, for all $i \in \mathbb{N}\cup\{0\}$, $\mathcal{A} \sqsubseteq \mathcal{C}_i$ and $\mathcal{B} \sqsubseteq \mathcal{C}_i$ since $a_0 \in Ind(\mathcal{A}\cap\mathcal{B})$ and $\mathcal{C}_{i+1} \sqsubset \mathcal{C}_i$. But then, for all $i > d$, we have that $\mathcal{C}_i \sqsubset \mathcal{C}$ which contradicts to our assumption that $\mathcal{C} \in lcsa(\mathcal{A},\mathcal{B})$. Now we will prove "$\Rightarrow$" by contraposition. If either $\mathcal{A}$ or $\mathcal{B}$ is non-cyclic, then $lcsa(\mathcal{A},\mathcal{B})$ obviously exists. Therefore, let $\mathcal{C} \in \mathbf{C}$ be of the form $\{(a_0,a_1):R_0,\ldots,(a_{n-1},a_0):R_{n-1}\}$ and let $\mathcal{D} \in \mathbf{D}$ be of the form $\{(a_0,b_1):S_0,\ldots,(b_{m-1},a_0):S_{m-1}\}$. Then, $lcsa(\mathcal{A},\mathcal{B})$ exists iff $lcsa(\mathcal{C},\mathcal{D})$ exists. Suppose that $a_{(i \bmod n)} = b_{(i \bmod m)}$ holds for all $i \in \mathbb{N} \cup \{0\}$. Furthermore, assume that $mss(R_{(i \bmod n)}, S_{(i \bmod m)})$ is defined for all $i \in \mathbb{N} \cup \{0\}$. Then, $lcsa(\mathcal{C},\mathcal{D})$ is given by the ABox $\{(a,b) : mss(R_{(i \bmod n)}, S_{(i \bmod m)}) | 1 \leq i \leq mn \wedge (a,b) : R_{(i \bmod n)} \in \mathcal{C} \wedge (a,b) : S_{(i \bmod m)} \in \mathcal{D}\}$. Now we will consider the case in which there exists an $i \in \mathbb{N} \cup \{0\}$ such that $mss(R_{(i \bmod n)}, S_{(i \bmod m)})$ is undefined. Let $\mathcal{C}'$ ($\mathcal{D}'$) be $\mathcal{C}$'s ($\mathcal{D}$'s) MSC-completion. Then, $lcsa(\mathcal{C},\mathcal{D})$ is given by $\{a : lcs(C,D) | a : C \in \mathcal{C}' \wedge a : D \in \mathcal{D}'\} \cup \{(a,b) : mss(R,S) | (a,b) : R \in \mathcal{C}' \wedge (a,b) : S \in \mathcal{D}' \wedge mss(R,S) \text{ is defined}\}$. Finally, if there exists an $i \in \mathbb{N} \cup \{0\}$ such that $mss(R_{(i \bmod n)}, S_{(i \bmod m)})$ is undefined and there exists a $j \in \mathbb{N} \cup \{0\}$ such that $a_{(j \bmod n)} \neq b_{(j \bmod m)}$, we again define $\mathcal{C}'$ ($\mathcal{D}'$) as $\mathcal{C}$'s ($\mathcal{D}$'s) MSC-completion and observe that $lcsa(\mathcal{C},\mathcal{D})$ is given by $\{a : lcs(C,D) | a : C \in$

$\mathcal{C}' \wedge a : D \in \mathcal{D}'\} \cup \{(a,b) : mss(R,S) | (a,b) : R \in \mathcal{C}' \wedge (a,b) : S \in \mathcal{D}' \wedge mss(R,S)$ is defined$\}$, which completes the proof. $\square$

The proposition states a set of necessary and sufficient condition for the (non-) existence of the LCSA of ABoxes. Before stating an algorithm which implements the test of the conditions, let us first introduce some useful notation. For individuals $a_1, \ldots, a_n$, $n \in \mathbb{N} \cup \{0\}$, we introduce a *sequence* as a syntactic expression of the form $[a_1, \ldots, a_n]$. Informally speaking, sequences correspond to multisets in which the order of elements is obeyed. We also introduce two types of operations on sequences. For sequences, $[a_1, \ldots, a_n]$ and $[b_1, \ldots, b_m]$, the sequence $[a_1, \ldots, a_n] \setminus [b_1, \ldots, b_m]$ emerges from $[a_1, \ldots, a_n]$ by eliminating all a_i from $[a_1, \ldots, a_n]$ such that there exists a $b_j \in \{b_1, \ldots, b_m\}$ with $a_i = b_j$. Moreover, $[a_1, \ldots, a_n] \circ [b_1, \ldots, b_m] = [a_1, \ldots, a_n, b_1, \ldots, b_m]$ denotes the concatenation of $[a_1, \ldots, a_n]$ and $[b_1, \ldots, b_m]$. Eventually, $[a_1, \ldots,$
$a_n]$ and $[b_1, \ldots, b_n]$ are equal iff $a_i = b_i$ holds for all $i \in \{1, \ldots, n\}$.

Algorithm 3 lcsa-undefined($firstp, cycle1p, cycle2p, startind1, startind2, ind1, ind2, sequence1, sequence2, \mathcal{A}, \mathcal{B}$)

```
if $(\neg firstp \wedge (ind1 = startind1) \wedge (ind2 = startind2))$ or $((ind1 = startind1) \wedge cycle2p)$ or $((ind2 = startind2) \wedge cycle1p)$ then
  $\neg(sequence1 \setminus [startind1] = sequence2 \setminus [startind2])$
else
  $\mathbf{S}_1 := \{(ind1, b) : R \in \mathcal{A}\}$;
  $\mathbf{S}_2 := \{(ind2, b) : R \in \mathcal{B}\}$;
  for all $(a,b) : R \in \mathbf{S}_1$ do
    for all $(c,d) : S \in \mathbf{S}_2$ do
      $mss(R,S) \wedge lcsa\text{-}undefined(false, \neg firstp \wedge (ind1 = startind1),$
      $\neg firstp \wedge (ind2 = startind2), startind1, startind2, b, d,$
      $sequence1 \circ [a], sequence2 \circ [b], \mathcal{A}, \mathcal{B})$
    end for
  end for
end if
```

Algorithm 3 checks if the conditions given in Proposition 7 hold for ABoxes $\mathcal{A}$ and $\mathcal{B}$. The first three parameters are Boolean variables, where $firstp$ indicates whether or not *lcsa-undefined* is in its first incarnation. Hence, at first invocation $firstp$ is set to $true$. $cycle1p$ ($cycle2p$) indicates whether a cycle has been detected in $\mathcal{A}$ ($\mathcal{B}$). Consequently, both variables are initialized with $false$. $startind1$ ($startind2$) takes the variable in $\mathcal{A}$ ($\mathcal{B}$) from which the cycle-test is started. $ind1$ ($ind2$) takes a variable which is checked for being the "end" of a cycle in $\mathcal{A}$ ($\mathcal{B}$). When invoking *lcsa-undefined*, both $startind1, startind2, ind1$, and $ind2$ are initialized with a variable $a \in Ind(\mathcal{A}) \cap Ind(\mathcal{B})$. Since a is always considered the starting individual of a possible cycle in $\mathcal{A}$ and $\mathcal{B}$, condition (i) need not be checked explicitly in the algorithm. *sequence*1 (*sequence*2) contains the sequence of variables involved in a possible cycle in $\mathcal{A}$ ($\mathcal{B}$). Since concept assertions do not influence the conditions in Proposition 7, we eliminate all concept assertions from $\mathcal{A}$ and $\mathcal{B}$ before invoking the algorithm. In the first **then**-clause, we check if either *ind*1 (*ind*2) is the last individual in the sequence of individuals forming a cycle in $\mathcal{A}$ ($\mathcal{B}$) or *ind*1 (*ind*2) is the last individual in the sequence of individuals forming a cycle in $\mathcal{A}$ ($\mathcal{B}$) and a cycle in $\mathcal{B}$ ($\mathcal{A}$) has already been detected before. If the **then**-clause evaluates to *true*, we check condition (i) in Proposition 7. Otherwise, in the **else**-clause we collect in $\mathbf{S}_1$ ($\mathbf{S}_2$) all role assertions of the form $(ind1, b) : R$ $[(ind2, b) : R]$. For each $(ind1, b) : R \in \mathbf{S}_1$ and $(ind2, d) : S \in \mathbf{S}_2$, we check if $mss(R, S)$ is defined (condition (ii) in Proposition 7). In case, $mss(R, S)$ is undefined, the algorithm correctly returns *false*. Otherwise, *lcsa-undefined* is invoked recursively with the following values. $firstp$ is set to false since its meaning is only to prevent us from entering the **then**-clause at first call. If $ind1 = startind1$ ($ind2 = startind2$) and $firstp$ does not hold, then a cycle is present in $\mathcal{A}$ ($\mathcal{B}$). Hence, in the next incarnation of *lcsa-undefined*, *cycle*1*p* (*cycle*2*p*) is set to *false* and *ind*1 (*ind*2) is concatenated to *sequence*1 (*sequence*2). In order to complete the test of the conditions (i) and (ii) in Proposition 7, we recursively call *lcsa-undefined* with $ind1 := b$ and $ind2 := d$. With these considerations, the following theorem can be proved.

Theorem 5 *Let $\mathcal{A}$ and $\mathcal{B}$ be satisfiable ABoxes and $\mathcal{A}' := \mathcal{A} \setminus \{a : C \in \mathcal{A}\}$ and $\mathcal{B}' := \mathcal{B} \setminus \{a : C \in \mathcal{B}\}$. Then, lcsa$(\mathcal{A}, \mathcal{B})$ does not exist iff $Ind(\mathcal{A}, \mathcal{B}) \neq \emptyset$ and, for all $a \in Ind(\mathcal{A}') \cap Ind(\mathcal{B}')$, the invocation* lcsa-undefined$(false, false, false, a, a, a, a, false, false, \mathcal{A}', \mathcal{B}')$ *returns* false. □

Thus, Algorithm 3 provides a decision procedure with which the (non-) existence of the LCSA of two ABoxes can be decided. If the LCSA exists, the following theorem shows how to compute it.

Theorem 6 *Let $\mathcal{A}$ and $\mathcal{B}$ be ABoxes and $\hat{\mathcal{A}}_{\mathcal{B}}$ $\mathcal{A}$'s MSC-completion w.r.t. $\mathcal{B}$ and $\hat{\mathcal{B}}_{\mathcal{A}}$ $\mathcal{B}$'s MSC-completion w.r.t. $\mathcal{A}$. Furthermore, let*

$$\begin{aligned} \mathcal{C} \;:=\; & \{a : lcs(C, D) | a : C \in \hat{\mathcal{A}}_{\mathcal{B}} \wedge a : D \in \hat{\mathcal{B}}_{\mathcal{A}}\} \cup \\ & \{(a, b) : mss(R, S) | (a, b) : R \in \hat{\mathcal{A}}_{\mathcal{B}} \wedge (a, b) : S \in \hat{\mathcal{B}}_{\mathcal{A}} \wedge \\ & \;\; mss(R, S) \textit{ is defined}\}. \end{aligned} \quad (3)$$

Then, if lcsa$(\mathcal{A}, \mathcal{B})$ exists, it is equivalent to $\mathcal{C}$.

Proof. Suppose $lcsa(\mathcal{A}, \mathcal{B})$ exists. We prove the claim by showing by induction on $\mathcal{A}$'s and $\mathcal{B}$'s cardinalities that, for all $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$, there exists a $\gamma \in \mathcal{C}$ such that $\alpha \sqsubseteq \gamma$, $\beta \sqsubseteq \gamma$, and, for all γ' with $\alpha \sqsubseteq \gamma'$ and $\beta \sqsubseteq \gamma'$, $\gamma \sqsubseteq \gamma'$ holds and $\mathcal{C}$ contains no further assertions. In case $|\mathcal{A}| = 0$ ($|\mathcal{B}| = 0$), we have $\hat{\mathcal{A}}_{\mathcal{B}} = \emptyset$ ($\hat{\mathcal{B}}_{\mathcal{A}} = \emptyset$) and $\mathcal{C} = \emptyset$ and the claim follows by Proposition 2 (i). Now let $\mathcal{A}$ and $\mathcal{B}$ be given with $n = |\mathcal{A}|$ and $m = |\mathcal{B}|$ and suppose $lcsa(\mathcal{A}, \mathcal{B}) \equiv \mathcal{C}$ (induction hypothesis). We define $\mathcal{A}' := \mathcal{A} \cup \{\alpha'\}$, where α' is either of the form $a : C$ or $(a, b) : R$. Now let $\alpha \in \mathcal{A}'$. In case $\alpha \neq \alpha'$, the claim is a consequence of the induction hypothesis. Therefore, let $\alpha = \alpha'$ and suppose α is of the form $a : C$. Since $\hat{\mathcal{B}}_{\mathcal{A}}$ is the MSC-completion of $\mathcal{B}$, it follows that either the individual a does not occur in $\mathcal{B}$ (and hence not in $\hat{\mathcal{B}}_{\mathcal{A}}$ either) or there exists a concept assertion β of the form $a : D \in \hat{\mathcal{B}}_{\mathcal{A}}$. In the former case, nothing needs to be shown since $lcsa(\{a : C\}, \mathcal{B}) \equiv \emptyset$. Otherwise, there exists a concept assertion $a : lcs(C, D) \in \mathcal{C}'$ and $a : C \sqsubseteq a : lcs(C, D)$, $a : D \sqsubseteq a : lcs(C, D)$, and $a : lcs(C, D) \sqsubseteq \gamma'$, for all γ', follows according to Proposition 4 (i). Now suppose that α is of the form $(a, b) : R$. Unless there exists a role assertion $\beta \in \hat{\mathcal{B}}_{\mathcal{A}}$ of the form $(a, b) : S$ where $mss(R, S)$ is defined, nothing needs to be shown. Otherwise, there exists a role assertion $(a, b) : mss(R, S) \in \mathcal{C}'$ and $(a, b) : R \sqsubseteq (a, b) : mss(R, S)$, $(a, b) : S \sqsubseteq (a, b) : mss(R, S)$ and $(a, b) : mss(R, S) \sqsubseteq \gamma'$, for all γ', holds according to Proposition 4 (ii). According to (3), $\mathcal{C}'$ contains no further assertions and thus, $lcsa(\mathcal{A}', \mathcal{B}) \equiv \mathcal{C}'$ holds. The proof for the case $|\mathcal{A}| = n$ and $|\mathcal{B}| = m + 1$ is analogous. □

Given any of the description logics $\mathcal{ALENR}$ or $\mathcal{ALQ}$, Algorithm 4 is an LCSA implementation taking $\mathcal{A}$ and $\mathcal{B}$ which are both either ABoxes w.r.t $\mathcal{ALENR}$ or $\mathcal{ALQ}$. The function *compute-lcs* implements the LCS of concepts as in Definition 11. In [6], we give an LCS algorithm for the description logics $\mathcal{ALENR}$ and $\mathcal{ALQ}$.

Theorem 7 *Let $\mathcal{A}$ and $\mathcal{B}$ be ABoxes. Then, if $lcsa(\mathcal{A}, \mathcal{B})$ exists, the invocation* compute-lcsa$(\mathcal{A}, \mathcal{B})$ *terminates and returns an ABox $\mathcal{C}$ which is equivalent to $lcsa(\mathcal{A}, \mathcal{B})$.*

Proof. Termination of the algorithm follows by the termination of the functions *compute-preprocessing-completion*, *compute-msc*, and *compute-lcs* since *compute-lcsa* is not recursively invoked. If $\mathcal{A}$ ($\mathcal{B}$) is unsatisfiable, we return $\mathcal{B}$ ($\mathcal{A}$). The correctness of this follows by Proposition 2. Otherwise, in the **else**-branch we first compute the preprocessing completions of $\mathcal{A}$ and $\mathcal{B}$ by means of the function *compute-preprocessing-completions* and construct the MSC-completions $\hat{\mathcal{A}}_{\mathcal{B}}$ of $\mathcal{A}$ and $\hat{\mathcal{B}}_{\mathcal{A}}$ of $\mathcal{B}$ in the first two **for**-loops. Then, we initialize $\mathcal{C}$ to the empty ABox and add a concept assertion $a : \textit{compute-lcs}(C, D)$

Algorithm 4 compute-lcsa($\mathcal{A}, \mathcal{B}$)

```
if $\mathcal{A}$ ($\mathcal{B}$) is unsatisfiable then
  $\mathcal{B}$ ($\mathcal{A}$)
else
  $\mathcal{A} := \mathit{compute\text{-}preprocessing\text{-}completion}(\mathcal{A})$;
  $\mathcal{B} := \mathit{compute\text{-}preprocessing\text{-}completion}(\mathcal{B})$;
  $\hat{\mathcal{A}}_{\mathcal{B}} := \mathcal{A}$;
  $\hat{\mathcal{B}}_{\mathcal{A}} := \mathcal{B}$;
  for all $a \in \mathit{Ind}(\mathcal{A})$ do
    $\hat{\mathcal{A}}_{\mathcal{B}} := \hat{\mathcal{A}}_{\mathcal{B}} \cup \{a : \mathit{compute\text{-}msc}(a, \max(\{\mathit{depth}(\mathcal{A}), \mathit{depth}(\mathcal{B})\}), \mathcal{A})\}$
  end for
  for all $b \in \mathit{Ind}(\mathcal{B})$ do
    $\hat{\mathcal{B}}_{\mathcal{A}} := \hat{\mathcal{B}}_{\mathcal{A}} \cup \{b : \mathit{compute\text{-}msc}(b, \max(\{\mathit{depth}(\mathcal{A}), \mathit{depth}(\mathcal{B})\}), \mathcal{B})\}$
  end for
  $\mathcal{C} := \emptyset$;
  for all $\alpha$ in $\hat{\mathcal{A}}_{\mathcal{B}}$ do
    for all $\beta$ in $\hat{\mathcal{B}}_{\mathcal{A}}$ do
      if $\alpha$ is of the form $a : C$ and $\beta$ is of the form $a : D$ then
        $\mathcal{C} := \mathcal{C} \cup \{a : \mathit{compute\text{-}lcs}(C, D)\}$
      else if $\alpha$ is of the form $(a, b) : R$ and $\beta$ is of the form $(a, b) : S$ and
      $\mathit{mss}(R, S)$ is defined then
        $\mathcal{C} := \mathcal{C} \cup \{(a, b) : \mathit{mss}(R, S)\}$
      end if
    end for
  end for
  $\mathcal{C}$
end if
```

to $\mathcal{C}$ iff there exists a pair of concept assertions $a : C \in \hat{\mathcal{A}}_{\mathcal{B}}$ and $a : D \in \hat{\mathcal{B}}_{\mathcal{A}}$. We add a role assertion $(a, b) : mss(R, S)$ to C iff there exists a pair of role assertions $(a, b) : R \in \hat{\mathcal{A}}_{\mathcal{B}}$ and $(a, b) : S \in \hat{\mathcal{B}}_{\mathcal{A}}$ and $mss(R, S)$ is defined. Since *compute-lcs* is assumed to be a correct LCS implementation, *compute-lcsa*$(\mathcal{A}, \mathcal{B})$ returns an ABox equivalent to $\mathcal{C}$ in (3), and the claim is a consequence of Theorem 6. □

Summarizing, in order to compute the LCSA of ABoxes $\mathcal{A}$ and $\mathcal{B}$, we first check whether $lcsa(\mathcal{A}, \mathcal{B})$ is defined. This can be done by Algorithm 3. If $lcsa(\mathcal{A}, \mathcal{B})$ is defined, it can be determined by Algorithm 4.

5 Complexity Results

In this section, we will state complexity results starting with a definition of the size of an ABox. It turns out that the size of the LCSA of n ABoxes w.r.t. either $\mathcal{ALENR}$ or $\mathcal{ALQ}$ is polynomial in the sizes of the input concepts. The ABox size definition does not take into account sizes of concepts which occurs in concept assertions included in the input ABoxes. Therefore, we will give an alternative ABox size definition which takes concepts into consideration. As a consequence, the size of the LCSA will blow up exponentially in the sizes of the input ABoxes in the worst case.

Definition 21 (Size of an ABox) *Let $\mathcal{A}$ be an ABox. Then we define the* size *of $\mathcal{A}$ as*

$$size(\mathcal{A}) := |\mathcal{A}|. \qquad \square$$

Theorem 8 *The size of the LCSA of $\mathcal{A}_1, \dots, \mathcal{A}_n$, which are all ABoxes w.r.t. either $\mathcal{ALENR}$ or $\mathcal{ALQ}$, is polynomial in the sizes of $\mathcal{A}_1, \dots, \mathcal{A}_n$ if $lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ exists.*

Proof. According to the LCSA algorithm, we add at most $|Ind(\mathcal{A})|$ ($|Ind(\mathcal{B})|$) concept assertions to $\mathcal{A}$ ($\mathcal{B}$). Hence, the size of $\hat{\mathcal{A}}_{\mathcal{B}}$ ($\hat{\mathcal{B}}_{\mathcal{A}}$) is polynomial in the size of $\mathcal{A}$ ($\mathcal{B}$). From Theorem 6 it follows that the size of $lcsa(\mathcal{A}, \mathcal{B})$ is polynomial in the sizes of $\hat{\mathcal{A}}_{\mathcal{B}}$ and $\hat{\mathcal{B}}_{\mathcal{A}}$. Now it can easily be shown by induction on n that the size of $lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ is also polynomial in the sizes of $\mathcal{A}_1, \dots, \mathcal{A}_n$. □

Theorem 8 shows that the size of $lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ does not grow exponentially in the sizes of $\mathcal{A}_1, \dots, \mathcal{A}_n$. However, in real applications the size of storage needed to store the ABox $lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ can be exponential in the storage needed for $\mathcal{A}_1, \dots, \mathcal{A}_n$. The reason is that the ABox size does not take into account the sizes of concepts involved in concept assertions.

Definition 22 (Size of a Concept) *Let C be either an $\mathcal{ALENR}$ or an $\mathcal{ALQ}$ concept. Then we define the* size *of C, $|C|$, recursively on C's structure as follows:*

(i) If $C = \top$ or $C = \bot$, then $|C| := 1$.

(ii) If C is an atomic or negated atomic concept, then $|C| := 1$.

(iii) If C is of the form $\exists\, R.D$, $\forall\, R.D$, $(\geq\ n\ R\ D)$, or $(\leq\ n\ R\ D)$, then $|C| := 1 + |D|$.

(iv) If C is of the form $C_1 \sqcap \cdots \sqcap C_n$, then $|C| := |C_1| + \cdots + |C_n|$.

Now we can give the definition of the exact ABox size which takes concept sizes into consideration.

Definition 23 (Exact Size of an ABox) *Let $\mathcal{A}$ be an ABox w.r.t. $\mathcal{ALENR}$ or $\mathcal{ALQ}$. Then we define the* exact size *of $\mathcal{A}$ as*

$$e\text{-}size(\mathcal{A}) := |\{(a,b) : R \in \mathcal{A}\}| + \sum_{a:C\in\mathcal{A}} |C|.$$

Now we introduce the notion of minimality of ABoxes w.r.t. their exact size.

Definition 24 (Minimal Exact ABox Size) *Let $\mathcal{A}$ be an ABox. Then we say that $\mathcal{A}$ has* minimal *exact size iff, for all $\mathcal{A}'$, we have that $\mathcal{A} \equiv \mathcal{A}'$ implies $e\text{-}size(\mathcal{A}) \leq e\text{-}size(\mathcal{A}')$.*

Thus, an ABox $\mathcal{A}$ has minimal exact size if there exists no equivalent ABox with smaller exact size than $\mathcal{A}$. Unfortunately, the result in Theorem 8 does not hold as soon as we consider exact ABox sizes. Subsequently, we will also need a similar notion of minimality of concepts.

Definition 25 (Minimal Concept Size) *Let C be an $\mathcal{ALENR}$ ($\mathcal{ALQ}$) concept. Then we say that C has* minimal *size iff, for all $C' \in \mathcal{ALENR}$ ($\mathcal{ALQ}$), we have that $C \equiv C'$ implies $|C| \leq |C'|$.*

With these preparations, we can state the following theorem.

Theorem 9 *The exact size of the LCSA of $\mathcal{A}_1, \dots, \mathcal{A}_n$, which are all ABoxes w.r.t. either $\mathcal{ALENR}$ or $\mathcal{ALQ}$, can be exponential in the exact sizes of $\mathcal{A}_1, \dots, \mathcal{A}_n$ if $lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ exists.*

Proof. The problem of computing the LCS of $\mathcal{ALENR}$ ($\mathcal{ALQ}$) concepts can be reduced to computing the LCSA of ABoxes w.r.t. either $\mathcal{ALENR}$ or $\mathcal{ALQ}$. Let $C_1, \dots, C_n$ be either $\mathcal{ALENR}$ or $\mathcal{ALQ}$ concepts and $\mathcal{A}_1 = \{a : C_1\}, \dots, \mathcal{A}_n = \{a : C_n\}$ be ABoxes. Then $lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n) \equiv \{a : lcs(C_1, \dots, C_n)\}$. In [6], it is shown that the LCS of n $\mathcal{ALENR}$ or $\mathcal{ALQ}$ concepts $C_1, \dots, C_n$ can blow up exponentially in the sizes of $C_1, \dots, C_n$. Obviously, if E with $E \equiv lcs(C_1, \dots, C_n)$ has minimal size, then $\{a : E\}$ has minimal exact size, which proves the claim. $\square$

Even though, Theorem 9 shows that the LCSA of ABoxes can blow up exponentially in the worst case, retrieval in the commonality-based information retrieval framework can be optimized by first sorting the ABoxes in the database according to the subsumption relation. Given a database DB and ABoxes $\mathcal{A}_1, \dots, \mathcal{A}_n, \mathcal{D}, \mathcal{D}' \in DB$, we can omit the subsumption test between $lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ and $\mathcal{D}'$ if both $\mathcal{D} \not\sqsubseteq lcsa(\mathcal{A}_1, \dots, \mathcal{A}_n)$ and $\mathcal{D} \sqsubseteq \mathcal{D}'$ hold.

6 Conclusion and Future Work

In this article, we have introduced new description logic inference services useful for commonality-based information retrieval which is considered to be an interesting research topic in description logic applications. We showed with an example that performing commonality-based information retrieval in the usual way (i.e. based on applying the least common subsumer on the most specific concepts of knowledge base individuals which represent user-specified information examples) is not always appropriate and suggested a theoretical framework to overcome the shortcomings. We first gave a definition of subsumption for ABoxes and provided an algorithm for deciding this problem. Thereby, only two requirements are imposed on the description logic underlying the ABoxes: A constructor for full concept negation must be present and an algorithm for checking ABox satisfiability must be available. We proved soundness and completeness of the ABox subsumption algorithm and showed that ABox subsumption is at most as complex as ABox satisfiability checking. The notion of ABox subsumption is used for the definition of a least common subsuming ABox operation which was introduced as a generalization operation for ABoxes in a similar way as the least common subsumer for concepts. We gave algorithms for both checking the existence of the least common subsuming ABox and, in case of existence, for computing the least common subsuming ABox of ABoxes w.r.t. to the two description logics $\mathcal{ALENR}$ and $\mathcal{ALQ}$. We showed that the exact size of the least common subsuming ABox can become exponential in the exact sizes of the ABoxes to which it is applied. As a by-product, we developed an algorithm for computing the most specific concept of ABox individuals occurring in ABoxes w.r.t. $\mathcal{ALENR}$ and $\mathcal{ALQ}$. The reason for restricting the LCSA operation to ABoxes w.r.t. the two mentioned languages is that the least common subsumer operation is needed in our algorithm and $\mathcal{ALENR}$ and $\mathcal{ALQ}$ are among the most expressive languages for which the least common subsumer is available. Future research should include the extension of the LCSA operation to ABoxes w.r.t. more expressive description logic languages. Possibly this requires the extension of the least common subsumer to more expressive description logics as well.

References

[1] M. Buchheit, F.-M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[2] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. *Principles of Knowledge Representation*, chapter Reasoning in Description Logics, pages 191–236. CSLI Publications, 1996.

[3] V. Haarslev and R. Möller. Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles. Technical Report FBI-HH-M-288/99, Department of Computer Science, University of Hamburg, 1999.

[4] B. Hollunder. *Algorithmic Foundations of Terminological Knowledge Representation Systems*. PhD thesis, Universität des Saarlandes, 1994.

[5] I. Horrocks, U. Sattler, and S. Tobies. A Description Logic with Transitive and Converse Roles, Role Hierarchies, and Qualifying Number Restrictions. LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999.

[6] T. Mantay. Computing Least Common Subsumers in Expressive Description Logics. Technical Report FBI-HH-M-286/99, Department of Computer Science, University of Hamburg, 1999.

[7] R. Möller and M. Wessel. Terminological Default Reasoning about Spatial Information: A First Step. In *Proceedings of the International Conference on Spatial Information Theory, COSIT'99*, Stade, 1999. Springer-Verlag.