Computing Least Common Subsumers in Expressive Description Logics

Thomas Mantay, AI Lab, University of Hamburg
E-Mail: mantay@informatik.uni-hamburg.de

Abstract

Computing least common subsumers in description logics is an important reasoning service useful for a number of applications. As shown in the literature, this reasoning service can be used for the approximation of concept disjunctions in description logics, for the "bottom-up" construction of knowledge bases, and for specific kinds of information retrieval. So far, the least common subsumer operator has been restricted to description logics which do not contain both existential restrictions and number restrictions. In this article, we present a least common subsumer operator for the expressive description logics $\mathcal{ALENR}$ and $\mathcal{ALEQ}$ which both include these operators.

1 Introduction

Knowledge representation languages based on Description Logics (DLs) have proven to be a useful means for representing the terminological knowledge of an application domain in a structured and formally well understood way [5]. In DLs, knowledge bases are formed out of *concepts* representing sets of individuals. Complex concepts are built out of atomic concepts and atomic roles (representing binary relations between individuals) using the concept constructors provided by the DL language. For example, the set of *mothers having at least one child and at least one adult daughter* can be described by the concept special-mother using the atomic concepts mother, adult, and person and the roles has-child and has-daughter:

$$\mathsf{special\text{-}mother} \doteq \mathsf{mother} \sqcap \exists\, \mathsf{has\text{-}child.person} \sqcap \exists\, \mathsf{has\text{-}daughter.adult}.$$

In this article, we focus on the DL $\mathcal{ALENR}$ and $\mathcal{ALEQ}$ whose syntax and semantics will be introduced below.

A central feature of knowledge representation systems based on DLs is a set of reasoning services with the ability to deduce implicit knowledge from explicitly represented knowledge. For instance, the subsumption relation between two concepts can be determined. Intuitively speaking, 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. Furthermore, *instantiation* describes the problem of determining whether or not a given individual is an instance of a concept.

As another reasoning service, the *least common subsumer* (LCS) operation, applied to concepts C and D, computes the most specific concept which subsumes C and D. The LCS operation is an important reasoning service useful for a number of applications. [3] considers an LCS operator for the DL $\mathcal{ALN}$ in order to approximate a disjunction operator which is not explicitly included in $\mathcal{ALN}$. Also, the operator is used as a subtask for the "bottom-up" construction of KBs based on the DLs $\mathcal{ALN}$ with cyclic concept definitions [1] and $\mathcal{ALE}$ [2]. In our applications, the LCS operation is used as a subtask for similarity-based information retrieval [9] (see [8] for an LCS operator for a probabilistic extension of $\mathcal{ALN}$). The goal is to provide a user of an information system with an example-based query mechanism. The data of an information system are modeled as DL individuals. For instance, in a document retrieval application, the Technical Report tr-fbi-hh-m-286/99 could be modeled as an instance of the concept DL-document. The "commonalities" of the selected data of interest to the user are formalized by a DL concept which (i) the user-selected examples are instances of and (ii) is the most specific concept (w.r.t. subsumption) with property (i). A concept fulfilling properties (i) and (ii) will then be used as a retrieval filter. The task of similarity-based information retrieval can be split into three subtasks: First, the most specific concepts of a finite set of individuals is computed yielding a finite set of concepts. Then the LCS of these concepts is computed. Eventually, the instances of the LCS concept are determined. Similarity-based information retrieval can also be applied in the presence of a terminology ("TBox"). However, all concept specializations must be transformed into concept definitions and the concepts involved in a retrieval must be unfolded before the three mentioned operations can be applied. Computing the most specific concept of *arbitrary* individuals was investigated in [1] for the DL $\mathcal{ALN}$ with cyclic concept definitions. However, for the purpose of similarity-based information retrieval, this task is trivial since a KB engineer will design the KB in such a way that each KB individual is an instance of named concepts. The third subtask has been subject to many investigations (see [7] for the DL $\mathcal{ALCQ}$ and [5] for several other DLs). So far, LCS computation has only been possible for DLs with strongly

limited expressivity. None of the DLs that the LCS operation has yet been introduced for includes an operator for number restrictions and existential quantifications.

The main contribution of this paper is the proposal of an LCS operator for the expressive DLs $\mathcal{ALENR}$ and $\mathcal{ALEQ}$ consisting of the top and bottom concept, atomic concepts, negations of atomic concepts, concept conjunctions augmented by either existential and universal role quantifications, role conjunctions, and number restrictions ($\mathcal{ALENR}$) or qualified number restrictions ($\mathcal{ALEQ}$). The special challenge is a proper treatment of conjunctions of existential role quantifications in combination with number restrictions (in $\mathcal{ALENR}$) and conjunctions of qualified number restrictions (in $\mathcal{ALEQ}$). Qualifying number restrictions are essential language constructs partly available in some knowledge representation systems. For instance, the concept language used in KANDOR allows for qualifying number restrictions in a restricted form. Also, the constructors are included into the assertional part ("ABox") of the system MESON [10]. Section 2 formally introduces syntax and semantics of $\mathcal{ALENR}$ and $\mathcal{ALEQ}$ and gives an LCS definition. Section 3 explains how the LCS will be computed. The computation process rests on a special representation of the involved concepts in which all relevant information contained in concepts is made explicit in order to simplify the determination of their LCS. In Section 4 we show an algorithm to compute this information for $\mathcal{ALEQ}$ concepts and Section 5 $\mathcal{ALENR}$ aims at the same task for $\mathcal{ALENR}$ concepts. Finally, in Section 6 we present the LCS algorithm and show soundness, completeness, and complexity results. We conclude with a summary and proposals for future research topics.

2 The Underlying Description Logics

In this section, we review the definition and some properties of the DLs $\mathcal{ALENR}$ (e.g., considered in [4, 5]) and $\mathcal{ALEQ}$ (a sublanguage of $\mathcal{ALCQ}$ introduced in [6]).

Definition 1 (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 $n \in \mathbb{N} \cup \{0\}$. 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 concepts.
- *If R and S are roles, then $R \sqcap S$ is a role (role conjunction).* □

Definition 2 (Syntax of $\mathcal{ALEQ}$) *Let $\mathcal{C}$ be a set of* atomic concepts *and $\mathcal{R}$ a set of* atomic roles *disjoint from $\mathcal{C}$. $\mathcal{ALEQ}$ concepts are recursively defined as follows:*

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

  are also concepts. □

A *subconcept* of a concept C is a substring of C that qualifies as a concept. The semantics of an $\mathcal{ALENR}$ and $\mathcal{ALEQ}$ concept is defined in terms of an interpretation.

Definition 3 (Interpretation, model, coherence) *An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of an $\mathcal{ALENR}$ or $\mathcal{ALEQ}$ 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 $\mathcal{ALEQ}$ concept as follows. Assume that $A^{\mathcal{I}}, C^{\mathcal{I}}, D^{\mathcal{I}}$ and $R^{\mathcal{I}}, S^{\mathcal{I}}$ are already given and $n \in \mathbb{N} \cup \{0\}$. 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* $\mathcal{ALEN}\mathcal{R}$ *or* $\mathcal{ALE}\mathcal{Q}$ *concept* C *iff* $C^{\mathcal{I}} \neq \emptyset$. *In this case,* C *is called* coherent. □

Note that both constructors $\top$ and $\bot$ are expressible by $(\geq\ 0\ R)$ and $A \sqcap \neg A$, respectively. We will call R a *subrole* of S iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$. One of the most important inference services provided by knowledge base systems is the determination of the subsumption relationship between two concepts.

Definition 4 (Subsumption, equivalence) *A concept* C is subsumed by *a concept* D *(*$C \sqsubseteq D$*) iff* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *holds for all interpretations* $\mathcal{I}$ *of* C *and* D. C *is* equivalent *to* D *(*$C \equiv D$*) iff* $C \sqsubseteq D \wedge D \sqsubseteq C$. □

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

Definition 5 *The* depth *of a concept* C *is recursively defined over its structure.*

- *If* $C = \exists\, R.C'$, $C = \forall\, R.C'$, $C = (\geq\ n\ R\, C')$, $C = (\leq\ n\ R\, C')$, *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,* $depth(C) = 0$.

□

Furthermore, for a coherent concept C in which a role R occurs as a substring, we want to express that in any model of C an individual functioning as an R-successor is required.

Definition 6 *Given a coherent concept* C *and a role* R *occurring in* C, *we say that* C has an R-successor *iff for all models* $\mathcal{I}$ *of* C *there are individuals* $i, j \in \Delta^{\mathcal{I}}$ *such that* $(i, j) \in R^{\mathcal{I}}$.

□

Example. Let $C_1 = \exists\, R.A'$ and $C_2 = A_1 \sqcap \forall\, R.A_2$ be concepts. Then C_1 has an R-successor because an individual as an instance of A' is required to satisfy C_1. C_2 has no R-successor since, for instance, $\{i : A_1\}$ is a model of C_2.

For the presentation of the algorithms involved in the LCS computation, it is convenient to arrange a concept in sorted normal form.

Definition 7 (Sorted normal form) *An* $\mathcal{ALEN}\mathcal{R}$ *concept* C *is in* sorted normal form *(SNF) iff*

$$\begin{aligned} C &= \top \vee C = \bot \vee C = A \sqcap E \sqcap F \text{ with} \\ E &= \sqcap_{1 \leq i \leq n} \exists\, R_i.C_i \text{ and} \\ F &= \sqcap_{1 \leq j \leq m} \forall\, R_j^{'}.C_j^{'}. \end{aligned} \tag{1}$$

where A *is an arbitrary conjunction of atomic concepts, negated atomic concepts,* $\geq$*-, or* $\leq$*-restrictions. An* $\mathcal{ALEQ}$ *concept* C *is in SNF if*

$$\begin{aligned} C &= \top \vee C = \bot \vee A \sqcap L \sqcap M \text{ with} \\ L &= \sqcap_{1 \leq i \leq n} (\geq\ p_i\, R_i\, C_i) \text{ and} \\ M &= \sqcap_{1 \leq j \leq m} (\leq\ p_j^{'}\, R_j^{'}\, C_j^{'}). \end{aligned} \tag{2}$$

where A *is an arbitrary conjunction of atomic concepts or negated atomic concepts and* C_i *and* $C_j^{'}$ *are also in SNF.*

□

Obviously, any $\mathcal{ALEN}\mathcal{R}$ and $\mathcal{ALEQ}$ concept can be transformed into an equivalent concept in SNF in linear time by sorting its components on each depth.

In this article, we are interested in the inference task of computing the LCS of $\mathcal{ALEN}\mathcal{R}$ and $\mathcal{ALEQ}$ concepts.

Definition 8 *Let C and D be either both $\mathcal{ALENR}$ or $\mathcal{ALEQ}$ concepts. Then we recursively define the set of* least common subsumers *as:*

$$lcs(C,D) := \{E \mid C \sqsubseteq E \wedge D \sqsubseteq E \wedge \forall E' : C \sqsubseteq E' \wedge D \sqsubseteq E' \Longrightarrow E \sqsubseteq E'\}.$$

□

From this definition it follows immediately that, for concepts C and D, all pairs of elements of $lcs(C,D)$ are equivalent.

Proposition 1 *Let C and D be either both $\mathcal{ALENR}$ or $\mathcal{ALEQ}$ concepts. Then*

$$\forall E, E' \in lcs(C,D) : E \equiv E'.$$

Proof. Assume $E, E' \in lcs(C,D)$ with $E \not\equiv E'$. Then $E \sqcap E'$ would be a more specific subsumer of C and D than E and E' and, hence, E (resp. E') cannot be an LCS of C and D which is a contradiction. □

Due to this uniqueness property, we will consider $lcs(C,D)$ as a concept rather than a set of concepts in the following.

Definition 8 can straightforwardly be extended to n arguments. In this case, lcs is associative and commutative and $lcs(C_1, \ldots, C_n) = lcs(C_1, lcs(C_2, \ldots lcs(C_{n-1}, C_n) \ldots))$. We will restrict the attention to the problem of computing the LCS of two concepts since the LCS of $n > 2$ concepts can be obtained by iterated applications of the binary LCS operation.

Definition 9 *Let R and S be roles in $\mathcal{ALENR}$. Then we define the* most specific role *of R and S as a partial function:*

$$msr(R,S) := \begin{cases} \sqcap_{T \in \{R_1,\ldots,R_n\} \cap \{S_1,\ldots,S_m\}} T & \text{if } R = R_1 \sqcap \cdots \sqcap R_n \text{ and } S = S_1 \sqcap \cdots \sqcap S_m \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

□

In the next section, we will discuss how to determine the LCS of two $\mathcal{ALENR}$ and $\mathcal{ALEQ}$ concepts and describe an algorithm to compute it.

3 Outline of the LCS Algorithm

Since $\mathcal{ALE}$ and $\mathcal{ALN}$ are sublanguages of both $\mathcal{ALENR}$ and $\mathcal{ALEQ}$, some of the operations for computing the LCS of $\mathcal{ALE}$ and $\mathcal{ALN}$ concepts can also be applied to $\mathcal{ALENR}$ and $\mathcal{ALEQ}$. As in [3], we discriminate over the language constructors.

Subsumption. If C and D are both either $\mathcal{ALENR}$ or $\mathcal{ALEQ}$ concepts with $D \sqsubseteq C$ ($C \sqsubseteq D$), their LCS is C (D).

$\top$ and $\bot$. This case can be reduced to the subsumption case because $\top$ subsumes any other concept and $\bot$ is subsumed by any other concept.

Atomic concepts. Given atomic concepts A and B, the least common concept subsuming both A and B is $\top$. The same holds in case A or B (or both) are negated atomic concepts.

Existential and universal role quantifications. In $\mathcal{ALENR}$, the LCS of two existential role quantifications $\exists R.C$ and $\exists S.D$ is the concept $\exists msr(R,S).lcs(C,D)$ if $msr(R,S)$ exists. Otherwise, their LCS is $\top$. Likewise, the LCS of universal role quantifications $\forall R.C$ and $\forall S.D$ is $\forall msr(R,S).lcs(C,D)$ in case $msr(R,S)$ exists and $\top$ otherwise.

$\geq$- and $\leq$-restrictions. For two $\mathcal{ALENR}$ constructs $(\geq n\,R)$ and $(\geq m\,S)$, the LCS is represented by the concept $(\geq \min\{n,m\}\ msr(R,S))$. Analogously, the LCS of $(\leq n\,R)$ and $(\leq m\,S)$ is $(\leq \max\{n,m\}\ msr(R,S))$.

Qualified $\geq$- and $\leq$-restrictions. For $\mathcal{ALEQ}$ concepts $(\geq n\,R\,C)$ and $(\geq mRD)$, the LCS is represented by the concept $(\geq \min\{n,m\}Rlcs(C,D))$. As in the case of $\leq$-restrictions, the LCS of two qualified $\leq$-restrictions $(\leq nRC)$ and $(\leq mRD)$ is a qualified $\leq$-restriction taking $\max\{n,m\}$ as its first and R as its second argument. The qualified $\leq$-constructor in $\mathcal{ALEQ}$ is an anti-monotonic operator, i.e., if its concept argument becomes "larger", the resulting concept after applying this operator becomes "smaller" and vice versa. Thus, we have to consider the most specific concept which is subsumed by C and D as the concept argument of the LCS concept: $C \sqcap D$. Hence, the LCS of $(\leq n\,R\,C)$ and $(\leq m\,R\,D)$ is $(\leq \max\{n,m\}\,R\,(C \sqcap D))$.

"Incomparable" concepts. All other concept combinations yield $\top$ as their LCS, except for concept conjunctions which will be considered below.
Concept conjunctions. Concept conjunctions require a special treatment

in the computation process. For the DLs $\mathcal{ALN}$ and $\mathcal{ALE}$ (see [3, 2]), the LCS of two concept conjunctions simply is the conjunction of the LCS of each pair of conjuncts. However, as the following example shows, this does not yield the correct result in the presence of $\mathcal{ALEQ}$ concepts.

Example 1 *Let*

$$\begin{aligned} X_1 &:= (\geq\ 1\, R\, (A \sqcap B)) \sqcap (\geq\ 1\, R\, (A \sqcap \neg B)) \\ Y_1 &:= (\geq\ 3\, R\, A). \end{aligned}$$

Then $lcs((\geq\ 1\, R\, (A \sqcap B)), (\geq\ 3\, R\, A)) \sqcap lcs((\geq\ 1\, R\, (A \sqcap \neg B)), (\geq\ 3\, R\, A))$ *yields a concept equivalent to* $(\geq\ 1\, R\, A)$ *which is a less specific subsumer of* X_1 *and* Y_1 *than* $(\geq\ 2\, R\, A)$.

The same problem occurs when handling conjunctions of $\mathcal{ALENR}$ concepts. However, the situation is even more complicated due to the possible existence of role conjunctions.

Example 2 *Let*

$$\begin{aligned} X_2 &:= \exists\, R.A_1 \sqcap \exists\, (R \sqcap S).\neg A_1 \text{ and} \\ Y_2 &:= (\geq\ 3\, R). \end{aligned}$$

Then $lcs(\exists\, R.A_1, (\geq\ 3\, R)) \sqcap lcs(\exists\, (R \sqcap S).\neg A_1, (\geq\ 3\, R))$ *yields a concept equivalent to* $\top$ *whereas* $lcs(X_2, Y_2) = (\geq\ 2\, R)$.

The reason for the incorrect result in Example 1 is that (conjunctions of) qualified $\geq$-restrictions always imply further qualified $\geq$-restrictions not already explicitly present. For instance, X_1 has at least two distinct R-successors (one successor as an instance of $A \sqcap B$ and another one as an instance of $A \sqcap \neg B$). Therefore, we can conclude that X_1 has at least two distinct R-successors as instances of A. In other words, $(\geq 2RA)$ follows from X_1. Thereby, the qualification A results from the LCS of the qualifications of the involved qualified $\geq$-restrictions: $lcs(A \sqcap B, A \sqcap \neg B) = A$. As Example 2 shows, (conjunctions of) existential role quantifications always imply $\geq$-restrictions. For instance, the subconcept $\exists\, R.A_1$ of X_2 implies $(\geq\ 1\, R)$ and $\exists R.A_1 \sqcap \exists (R \sqcap S).\neg A_1$ implies $(\geq 2R)$. Analyzing the concept X_2, we observe that X_2 has at least one unique $R \sqcap S$-successor and at least one unique R-successor. Thus, X_2 has two R-successors since every $R \sqcap S$-successor is also an R-successor. In other words, $(\geq\ 2\, R) \sqcap (\geq\ 1\, (R \sqcap S))$ follows from X_2. In general, for an $\mathcal{ALENR}$ or $\mathcal{ALEQ}$ concept C, the idea is to first make this *implicitly* contained information explicit for every role occurring in C

in an existential role quantification or qualified $\geq$-restriction and add it to C in the form of conjuncts of additional $\geq$-restrictions. For an $\mathcal{ALENR}$ concept, these additional $\geq$-restrictions have the form $(\geq\ n\,R)$ and for an $\mathcal{ALEQ}$ concept they have the form $(\geq\ n\,R\,C)$. Intuitively, in $\mathcal{ALENR}$ these additional $\geq$-restrictions indicate the minimum number of R-successors for each occurring role R. In $\mathcal{ALEQ}$ the additional qualified number restrictions have the same meaning, however, as Example 1 shows, a qualification C is necessary. In our example, we have $\hat{X_2} := X_2 \sqcap (\geq\ 2\,R) \sqcap (\geq\ 1 (R \sqcap S))$. Since Y_2 does not contain any implicit information relevant for LCS computation, it remains unchanged: $\hat{Y_2} := Y_2$. $\hat{X_2}$ contains all relevant information for computing $lcs(X_2, Y_2)$ and will therefore be called an $\geq$-*completion* of X_2. Once the $\geq$-completions $\hat{C}$ and $\hat{D}$ of two $\mathcal{ALENR}$ or $\mathcal{ALEQ}$ concepts C and D are computed, $lcs(C, D)$ can be determined by the conjunction of the LCS of $\hat{C}$'s and $\hat{D}$'s conjuncts:

$$\begin{aligned}
lcs(\hat{X_1}, \hat{Y_1}) &\equiv lcs((\geq\ 1\,R\,(A \sqcap B)), (\geq\ 3\,R\,A)) \sqcap \\
&\quad\ lcs((\geq\ 1\,R\,(A \sqcap \neg B)), (\geq\ 3\,R\,A)) \sqcap \\
&\quad\ lcs((\geq\ 2\,R\,A), (\geq\ 3\,R\,A)) \\
&\equiv (\geq\ 1\,R\,A) \sqcap (\geq\ 1\,R\,A) \sqcap (\geq\ 2\,R\,A) \\
&\equiv (\geq\ 2\,R\,A)
\end{aligned}$$

$$\begin{aligned}
lcs(\hat{X_2}, \hat{Y_2}) &\equiv lcs(\exists\,R.A_1, (\geq\ 3\,R)) \sqcap \\
&\quad\ lcs(\exists\,(R \sqcap S).\neg A_1, (\geq\ 3\,R)) \sqcap \\
&\quad\ lcs((\geq\ 2\,R), (\geq\ 3\,R)) \sqcap \\
&\quad\ lcs((\geq\ 1\,(R \sqcap S)), (\geq\ 3\,R)) \\
&\equiv \top \sqcap \top \sqcap (\geq\ 2\,R) \sqcap (\geq\ 1\,R) \\
&\equiv (\geq\ 2\,R).
\end{aligned}$$

Note that an analogous operation for universal role quantifications (in $\mathcal{ALENR}$) and qualified $\leq$-restrictions (in $\mathcal{ALEQ}$) is not necessary. Table 1 summarizes the necessary LCS operations in dependence of the concept forming operators.

In the following, we will define the $\geq$-completion of a concept and give an algorithm to compute it.

Definition 10 *For an $\mathcal{ALENR}$ concept C in SNF given by Equation (1), we define the* $\geq$-completion $\hat{C}$ *of C as*

$$\hat{C} := A \sqcap \hat{E} \sqcap \hat{F} \sqcap \tilde{E} \text{ with } \tilde{E} := \sqcap_{1 \leq i \leq n} (\geq\ \tilde{e_i}\,R_i) \tag{3}$$

Condition	$lcs(C, D)$
$D \sqsubseteq C$	C
$C \sqsubseteq D$	D
$C = A, C = B$	A if $A = B$, $\top$ otherwise
$C = \exists\, R.C'$ $D = \exists\, S.D'$	$\exists\, msr(R, S).lcs(C', D')$ if $msr(R, S)$ exists, $\top$ otherwise
$C = \forall\, R.C'$ $D = \forall\, S.D'$	$\forall\, msr(R, S).lcs(C', D')$ if $msr(R, S)$ exists, $\top$ otherwise
$C = (\geq\ n\, R)$ $D = (\geq\ m\, S)$	$(\geq\ \min\{n, m\}\ msr(R, S))$ if $msr(R, S)$ exists, $\top$ otherwise
$C = (\leq\ n\, R)$ $D = (\leq\ m\, S)$	$(\leq\ \max\{n, m\}\ msr(R, S))$ if $msr(R, S)$ exists, $\top$ otherwise
$C = (\geq\ n\, R\, C')$, $D = (\geq\ m\, R\, D')$	$(\geq\ \min\{n, m\}\ R\, lcs(C', D'))$
$C = (\leq\ n\, R\, C')$, $D = (\leq\ m\, R\, D')$	$(\leq\ \max\{n, m\}\ R\, (C' \sqcap D'))$
$C = C_1 \sqcap \cdots \sqcap C_n$, $D = D_1 \sqcap \cdots \sqcap D_m$	$lcs(\hat{C}, \hat{D})$, see text

Table 1: Computing the LCS of $\mathcal{ALEN\mathcal{R}}$ and $\mathcal{ALEQ}$ concepts C and D. The LCS of all concept combinations not listed here yield $\top$.

where $\forall i \in \{1,\ldots,n\} : \tilde{e_i} = \min(\{k \in I\!N | E \sqcap F \sqcap (\leq\ k\ R_i)$ *is coherent*$\})$ *and* $\hat{E}$ *and* $\hat{F}$ *are* $\geq$*-completions of* E *and* F*, respectively. If* $C = A$*, then* $\hat{C} := C$*. For an* $\mathcal{ALE}\mathcal{Q}$ *concept* C *in SNF given by Equation (2), we define the* $\geq$-completion $\hat{C}$ *of* C *as*

$$\hat{C} := A \sqcap \hat{L} \sqcap \hat{M} \sqcap \tilde{L} \text{ with } \tilde{L} := \sqcap_{1 \leq i \leq n} (\geq\ \tilde{l_i}\ R_i\ lcs(C_1,\ldots,C_n)) \quad (4)$$

where $\forall i \in \{1,\ldots,n\} : \tilde{l_i} = \min(\{k \in I\!N | L \sqcap M \sqcap (\leq\ k\, R_i\, lcs(C_1,\ldots,C_n))$ *is coherent*$\})$ *and* $\hat{L}$ *and* $\hat{M}$ *are* $\geq$*-completions of* L *and* M*, respectively. If* $C = A$*, then* $\hat{C} := C$*.*

In Definition 10 the relevant information implicitly contained in an $\mathcal{ALENR}$ ($\mathcal{ALE}\mathcal{Q}$) concept C is made explicit by $\tilde{E}$ ($\tilde{L}$). Intuitively, $\tilde{e_i}$ ($\tilde{l_i}$) represents the minimum number of role successors (as instances of $lcs(C_1,\ldots,C_n)$) required to satisfy the constraints imposed by $E \sqcap F$ ($L \sqcap M$) on depth 0 w.r.t. R_i. We can state the following proposition.

Proposition 2 *Let* C, D *be* $\mathcal{ALENR}$ *or* $\mathcal{ALE}\mathcal{Q}$ *concepts and* $\hat{C}$ *and* $\hat{D}$ *their corresponding* $\geq$*-completions. Then*

(i) $C \equiv \hat{C}$ *and*

(ii) if $C = C_1 \sqcap \cdots \sqcap C_n, D = D_1 \sqcap \cdots \sqcap D_m$ *and* $\hat{C} = \hat{C_1} \sqcap \cdots \sqcap \hat{C_n} \sqcap C_{n+1} \sqcap \cdots \sqcap C_{n'}, \hat{D} = \hat{D_1} \sqcap \cdots \sqcap \hat{D_m} \sqcap D_{m+1} \sqcap \cdots \sqcap D_{m'}$*, then* $lcs(C, D) = \sqcap_{\substack{1 \leq i \leq n' \\ 1 \leq j \leq m'}} lcs(C_i', D_j')$.

Proof. We show (i) by induction over the depth of C. If $depth(C) = 0$, we have $C = \hat{C} = A$. If $depth(C) = n + 1$, we have $C = A \sqcap E \sqcap F$ and $\hat{C} = A \sqcap \hat{E} \sqcap \hat{F} \sqcap \tilde{E}$ and, by assumption, $E \equiv \hat{E}$ and $F \equiv \hat{F}$. Now $\hat{C} \sqsubseteq C$ is obvious since $\tilde{E}$ is a conjunct in $\hat{C}$ but not in C. On the other hand, we have $C \sqsubseteq \tilde{E}$ because, by the definition of an $\geq$-completion, the constraints imposed by the $\geq$-restrictions of $\tilde{E}$ are already implicitly contained in C. Thus, $C \sqsubseteq \hat{C}$ holds as well. The same proof can be applied to $\mathcal{ALE}\mathcal{Q}$ concepts. (ii) also follows since, by construction, $\hat{C}$ and $\hat{D}$ are $\geq$-completions w.r.t. C and D, respectively, including all relevant subconcepts for computing $lcs(C, D)$. □

In order to be able to use the formula given by (ii) in Proposition 2 for computing the LCS of concept conjunctions, we will give an algorithm to compute the $\geq$-completion of an $\mathcal{ALE}\mathcal{Q}$ concept. $\mathcal{ALENR}$ concepts will be treated in Section 5.

4 Determining $\geq$-completions in $\mathcal{ALEQ}$

In order to determine the $\geq$-completion of an $\mathcal{ALEQ}$ concept C given by Equation (2), the numbers $\tilde{l}_i$ must be computed for any role R_i occurring on any depth of C. For simplicity of presentation, we will consider a fixed concept depth. The value of $\tilde{l}_i$ does only depend on L and M. For computing the parameters $\tilde{l}_i$ for each $i \in \{1, \ldots, n\}$ in Definition 10, we define $N_i := \{p_k | R_k = R_i \wedge 1 \leq k \leq n\}$ and observe that lower and upper bounds for $\tilde{l}_i$ are given by

$$l_i^{\tilde{min}} := \min(N_i) \leq \tilde{l}_i \leq \sum_{p \in N_i} p =: l_i^{\tilde{max}} \tag{5}$$

Thus, $\tilde{l}_i$ can be determined by a binary search algorithm which takes $\log_2(l_i^{\tilde{max}} \perp l_i^{\tilde{min}})$ satisfiability tests in the worst case.

Algorithm 1 recursively computes the $\geq$-completion of an $\mathcal{ALEQ}$ concept C. First, C is transformed into SNF. If C is $\top$ or $\perp$, C is already an $\geq$-completion and is returned unchanged. If no $\geq$-restrictions are present, $\hat{E}$ and $\tilde{L}$ will be set to $\top$. Otherwise, for each role R_i occurring on depth 0 of C, $l_i^{\tilde{min}}$ and $l_i^{\tilde{max}}$ are computed according to Equation (5). The variable D represents a concept which is built of a conjunction of all $\geq$- and $\leq$-restrictions involving role R_i. The function *compute-*$\tilde{l}_i$ implements the binary search algorithm for computing $\tilde{l}_i$ given D and the lower and upper bound for $\tilde{l}_i$. It falls back on a satisfiability checking algorithm of $\mathcal{ALEQ}$ concepts (see, for example, [6]). Next the algorithm recursively computes the $\geq$-completions of all $\geq$-restrictions of C on depth 0 ($\hat{E}$) and collects the information implicitly contained in C ($\tilde{L}$). At this point, the algorithm must perform an LCS computation. In Section 6 we will give an implementation of the LCS operation. As for $\leq$-restrictions only the $\geq$-completions of the quantifiers of the $\leq$-restrictions must be computed. If no $\leq$-restrictions are present, $\hat{F}$ is set to $\top$. Eventually, *completion*(C) returns C's $\geq$-completion $\hat{C} = A \sqcap \hat{E} \sqcap \hat{F} \sqcap \tilde{L}$.

Theorem 1 *For any $\mathcal{ALEQ}$ concept C, the call* completion*(C) returns a concept which is equivalent to C's $\geq$-completion $\hat{C}$.* □

Example 3 *Let*

$$\begin{aligned} X_3 \;=\; & (\geq\ 1\, R\, A) \sqcap (\geq\ 1\, R\, B) \sqcap (\geq\ 2\, R\, \neg A) \sqcap (\geq\ 2\, R\, \neg B) \sqcap \\ & (\leq\ 1\, R\, (A \sqcap B)) \end{aligned}$$

Algorithm 1 completion(*concept*)

$C :=$ SNF $(concept)$
$//C = A \sqcap \sqcap_{1 \leq i \leq n} (\geq\ p_i\ R_i\ C_i) \sqcap \sqcap_{1 \leq i \leq m} (\geq\ p_i^{'}\ R_i^{'}\ C_i^{'})$
if $C = \top \vee C = \bot \vee (n = 0 \wedge m = 0)$ **then**
  return C
else
  if $n = 0$ **then**
    $\tilde{L} := \top; \hat{E} := \top$
  else
    for $i \in \{1, \ldots, n\}$ **do**
      $N := \{p_k | R_k = R_i \wedge 1 \leq k \leq n\}$
      $min := \min(N)$
      $max := \sum_{p \in N} p$
      $D := \sqcap_{\substack{1 \leq j \leq n \\ R_j = R_i}} (\geq\ p_j\ R_j\ C_j) \sqcap M$
      $\tilde{l_i} := compute\text{-}\tilde{l_i}(D, min, max)$
      $\tilde{L_i} := (\geq\ \tilde{l_i}\ R_i\ lcs(C_1, \ldots, C_n))$
    end for
    $\hat{E} := \sqcap_{1 \leq i \leq n} (\geq\ p_i\ R_i\ completion(C_i))$
    $\tilde{L} := \sqcap_{1 \leq i \leq n} \tilde{L_i}$
  end if
  if $m = 0$ **then**
    $\hat{F} := \top$
  else
    $\hat{F} := \sqcap_{1 \leq j \leq m} (\leq\ p_j^{'}\ R_j^{'}\ completion(C_j^{'}))$
  end if
  return $(A \sqcap \hat{E} \sqcap \hat{F} \sqcap \tilde{L})$
end if

and let us compute the $\geq$-completion for X_3, completion(X_3). *Since X_3 is in SNF and we have four $\geq$-restrictions in X_3 the loop is iterated four times. In all four iterations we get $min = 1$, $max = 6$. The variable D collects all $\geq$- and $\leq$-restrictions involving the role of the current $\geq$ restriction. In our example we get $D = (\geq\ 1\, R\, A) \sqcap (\geq\ 1\, R\, B) \sqcap (\geq\ 2\, R\, \neg A) \sqcap (\geq\ 2\, R\, \neg B) \sqcap (\leq\ 1\, R\, (A \sqcap B))$ and the call* compute-$\tilde{l}_i(D, min, max)$ *returns 3 because D has three R-successors. Hence, $\forall i \in \{1, \ldots, 4\} : \hat{L}_i = (\geq\ 3\, R\, lcs(A, B, \neg A, \neg B))$ and $lcs(A, B, \neg A, \neg B) = \top$. Thus, $\tilde{L} = \sqcap_{i=1}^{4}(\geq\ 3\, R\, \top) \equiv (\geq\ 3\, R\, \top)$. Since $depth(X_3) = 1$, we get $\hat{E} = (\geq\ 1\, R\, A) \sqcap (\geq\ 1\, R\, B) \sqcap (\geq\ 2\, R\, \neg A) \sqcap (\geq\ 2\, R\, \neg B)$ and $F = (\leq\ 1\, R\, (A \sqcap B))$. Eventually,* completion(X_3) *returns the concept*

$$\begin{array}{rcl} \hat{X}_3 & = & (\geq\ 1\, R\, A) \sqcap (\geq\ 1\, R\, B) \sqcap (\geq\ 2\, R\, \neg A) \sqcap (\geq\ 2\, R\, \neg B) \sqcap \\ & & (\leq\ 1\, R\, (A \sqcap B)) \sqcap (\geq\ 3\, R\, \top) \end{array}$$

as desired.

5 Determining $\geq$-completions in $\mathcal{ALENR}$

In order to determine the $\geq$-completion of an $\mathcal{ALENR}$ concept given by Equation (1), the numbers $\tilde{e_i}$ must be computed for any role R_i occurring on any depth of C. Again, for simplicity of presentation, we will consider a fixed concept depth. The value of $\tilde{e_i}$ does only depend on E and F. In particular, the algorithm for computing the parameters $\tilde{e_i}$ must take into account the presence of role conjunctions.

For computing $\tilde{e_i}$ given an $\mathcal{ALENR}$ concept C, the idea is to build a set of new concepts $M_C := \{\check{C}_1, \ldots, \check{C}_q\}$ from C such that $\tilde{e_i}$ can easily be derived from M_C's elements.

Example 4 *Let*

$$S_1 := R \sqcap S', S_2 := R \sqcap S'', T := S_1 \sqcap T', U := T \sqcap U' \text{ be roles and}$$

$$\begin{array}{rcl} X_4 & := & \sqcap_{1 \leq i \leq 6} X_{4i} \\ & := & \exists\, R.A_1 \sqcap \exists\, S_1.(\neg A_1 \sqcap A_3) \sqcap \exists\, S_2.A_2 \sqcap \\ & & \exists\, T.A_4 \sqcap \exists\, U.A_5 \sqcap \forall\, U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4). \end{array}$$

We observe that the role successors satisfying the constraints $\exists\, R.A_1$ and $\exists\, S_2.A_2$ can be "merged" such that we only need one S_2-successor (as an instance of $A_1 \sqcap A_2$) in order to satisfy both constraints. Obviously, this merge would not be possible if the concept $\exists\, S_2.(A_1 \sqcap A_2)$ was not coherent or if a universal role quantification of the form $\forall\, S_2.\neg A_1$ was present.

In general, we will model the merging process as the result of an application of a *merging rule* to a concept C yielding a new concept C' which only differs from C in that two existential role quantifications are merged. By this construction we can guarantee that every model of C' is also a model of C and, hence, $C' \sqsubseteq C$. Our intention is to recursively apply this merging rule as long as no further rule applications are possible. A concept resulting from this series of merging rule applications will be called *merging rule complete*. Intuitively, this means that no further role successors can be merged.

Definition 11 *For an $\mathcal{ALENR}$ concept C in SNF given by Equation (1) we say that C' emerges from C by application of the* merging rule *($C \overset{m}{\perp\!\!\!\rightarrow} C'$) iff $\exists k, k' \in \{1, \ldots, n\}, k < k'$, with*

(i) $C = A \sqcap E \sqcap F$,

(ii) $C' = A \sqcap \sqcap_{\substack{1 \leq i < k \\ k+1 \leq i < k' \\ k'+1 \leq i \leq n}} \exists\, R_i.C_i \sqcap \exists\, R_{k'}.(C_{k'} \sqcap C_k) \sqcap F$,

(iii) $R_{k'}$ *is a subrole of* R_k, *and*

(iv) C' *is coherent.*

C' also merges from C by application of the merging rule if D' is a subconcept of C', D is a subconcept of C, and $D \overset{m}{\perp\!\!\!\rightarrow} D'$. C' emerges from C by successive applications of the merging rule *iff $\exists C_1, \ldots, C_p$, $p \in \mathbb{N} \cup \{0\}$, such that $C = C_1 \overset{m}{\perp\!\!\!\rightarrow} \cdots \overset{m}{\perp\!\!\!\rightarrow} C_p = C'$. C' is* merging rule complete w.r.t. *C iff C' emerges from C by successive applications of the merging rule and $\neg\exists C'' : C' \overset{m}{\perp\!\!\!\rightarrow} C''$. In this case, C' is called a* merging rule completion *of C. Furthermore, we define the* set of merging rule completions of C *as*

$$M_C := \{C' | C' \text{ is a merging rule completion of } C.\}$$

According to Definition 11, we can apply the merging rule to an $\mathcal{ALENR}$ concept C (yielding C') if C contains two conjunctive existential role quantifications involving roles R_k and $R_{k'}$, $R_{k'}$ is a subrole of R_k and the resulting concept C' is coherent. Furthermore, since we allowed for $p = 0$, a concept which is already merging rule complete qualifies as a merging rule completion.

Lemma 1 *Let C and C' be $\mathcal{ALENR}$ concepts with $C \overset{m}{\perp\!\!\!\rightarrow} C'$. Then*

$$C' \sqsubseteq C.$$

Proof. Let $C \stackrel{m}{\perp\!\!\!\rightarrow} C'$ with

$$\begin{aligned} C &= A \sqcap E \sqcap F \\ C' &= A \sqcap \sqcap_{\substack{1 \le i < k \\ k+1 \le i < k' \\ k'+1 \le i \le n}} \exists\, R_i.C_i \sqcap \exists\, R_{k'}.(C_{k'} \sqcap C_k) \sqcap F \end{aligned}$$

for some $k, k' \in \{1, \ldots, n\}$, $k < k'$, and $R_{k'}$ is a subrole of R_k. Then C and C' only distinguish by the subconcepts $\exists\, R_k.C_k \sqcap \exists\, R_{k'}.C_{k'}$ (included in C) and $\exists\, R_{k'}.(C_{k'} \sqcap C_k)$ (included in C'). Therefore, it suffices to show that $\exists\, R_{k'}.(C_{k'} \sqcap C_k) \sqsubseteq \exists\, R_k.C_k \sqcap \exists\, R_{k'}.C_{k'}$ which obviously holds. □

Theorem 2 *Let C be an $\mathcal{ALENR}$ concept and M_C be the set of C's merging rule completions. Then*

$$\forall C' \in M_C : C' \sqsubseteq C.$$

Proof. Let $C' \in M_C$. The case $C' = C$ is trivial. Therefore, let $C' \neq C$. Then $\exists C_1, \ldots, C_p$, $p \in \mathbb{N}$, such that $C = C_1 \stackrel{m}{\perp\!\!\!\rightarrow} \cdots \stackrel{m}{\perp\!\!\!\rightarrow} C_p = C'$. By applying Proposition 1 $p \perp 1$ times we have $C' = C_p \sqsubseteq C_{p-1}, \ldots, C_2 \sqsubseteq C_1 = C$ and, hence, $C' \sqsubseteq C$. □

As Theorem 2 shows, a merging rule completion C' of a concept C is always more special w.r.t. subsumption than C. This means that any model of C' is also a model of C. Hence, if we know that C' has n R-successors for a role R occurring in C', this also holds for C.

Example 5 *When applied to X_4, we can either merge the conjuncts $\exists\, R.A_1$ and $\exists\, S_2.A_2$ yielding*

$$\begin{aligned} W_1 := \; & \exists\, S_1.(\neg A_1 \sqcap A_3) \sqcap \\ & \exists\, S_2.(A_2 \sqcap A_1) \sqcap \\ & \exists\, T.A_4 \sqcap \\ & \exists\, U.A_5 \sqcap \\ & \forall\, U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4) \end{aligned}$$

or we can merge $\exists\, R.A_1$ and $\exists\, T.A_4$ yielding

$$\begin{aligned} W_2 := \; & \exists\, S_1.(\neg A_1 \sqcap A_3) \sqcap \\ & \exists\, S_2.A_2 \sqcap \\ & \exists\, T.(A_4 \sqcap A_1) \sqcap \\ & \exists\, U.A_5 \sqcap \\ & \forall\, U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4). \end{aligned}$$

Since no further merging rule applications are applicable to W_2, W_2 is merging rule complete, whereas, for W_1, the rule can be applied to the conjuncts $\exists\, S_1.(\neg A_1 \sqcap A_3)$ and $\exists\, T.A_4$ yielding the merging rule complete concept

$$\begin{aligned} W_3 \;:=\; & \exists\, S_2.(A_2 \sqcap A_1) \sqcap \\ & \exists\, T.(A_4 \sqcap \neg A_1 \sqcap A_3) \sqcap \\ & \exists\, U.A_5 \sqcap \\ & \forall\, U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4). \end{aligned}$$

Thus, $M_{X_4} = \{W_2, W_3\}$. This example shows that merging rule applications are not deterministic in general.

Iterated merging rule applications can be represented by a tree in which the nodes are formed out of concepts and an edge is drawn from a node representing concept C to the node representing concept C' if $C \stackrel{m}{\longrightarrow} C'$. The leaves of this tree represent the concepts of M_C. Figure 5 shows the corresponding tree for the iterated merging rule applications to X_4.

Proposition 3 *Let C be an $\mathcal{ALENR}$ concept and M_C be the set of C's $\geq$-completions. Then M_C is finite.*

Proof. For a fixed concept depth, there are only finitely many possibilities of merging rule applications, since the number of existential quantifications in a concept is finite. Furthermore, each merging rule application reduces the number of existential role quantifications by one. Since C has a finite length, by iterated merging rule applications we will always yield a concept to which the merging rule is no longer applicable. Thus, each node in the corresponding tree for iterated merging rule applications has finitely many successors and every path has a finite length. Now the proposition follows by König's Lemma. □

Algorithm 2 recursively computes the set of merging rule completions M_C for an $\mathcal{ALENR}$ concept C according to Definition 11. First, C is transformed into SNF. If no (conjunctions of) existential or universal role quantifications occur in C, C is already merging rule complete. Therefore, a set with C as the only element is returned. In case C is an existential (universal) role quantification $\exists\, R.C'$, we return a set of existential (universal) role quantifications of the form $\exists\, R.D$ where D is a recursively computed merging rule completion of C'. For the treatment of a concept conjunction, we first collect all existential and universal role quantifications in the set D and all other concept components in $\bar{D}$. We add $\top$ to both D and $\bar{D}$ in order to guarantee

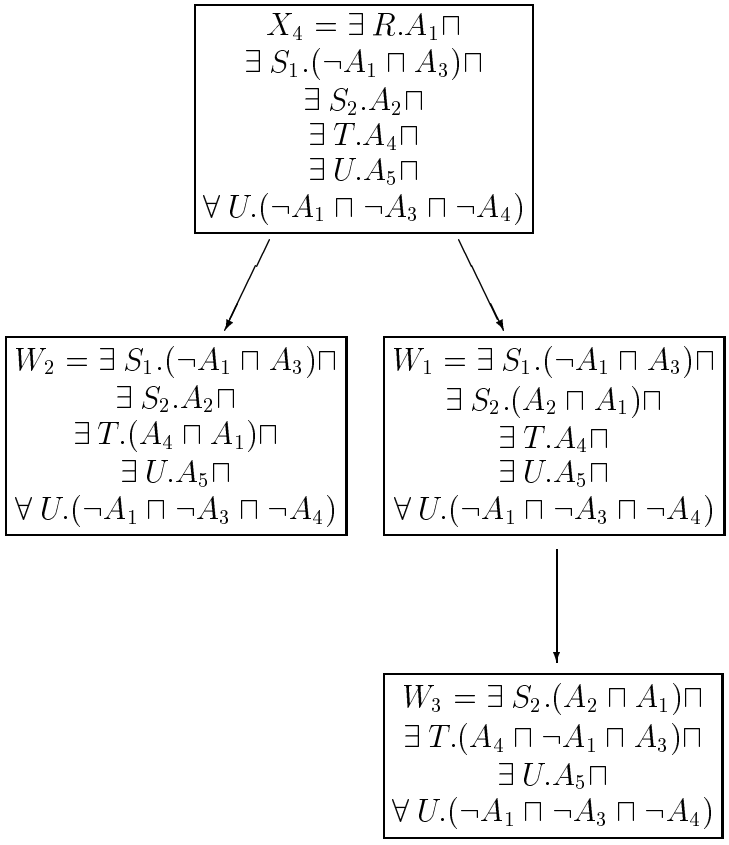

Figure 1: Iterated merging rule applications to example concept X_4.

Algorithm 2 merging-rule-completions(*concept*)

$C := SNF(concept)$
if $(C = A \vee C = \neg A \vee C = \top \vee C = \bot) \vee C = (\geq\ n\ R) \vee C = (\leq\ n\ R)$ **then**
  $\{C\}$
else if $C = \exists R.C'$ **then**
  $D :=$ merging-rule-completions(C')
  $\{\exists R.D' | D' \in D\}$
else if $C = \forall R.C'$ **then**
  $D :=$ merging-rule-completions(C')
  $\{\forall R.D' | D' \in D\}$
else if $C = C_1 \sqcap \cdots \sqcap C_n$ **then**
  $D := \{E \in \{C_1, \ldots, C_n\} | E = \exists R.C' \vee E = \forall R.C'\} \cup \{\top\}$
  $\bar{D} := \{C_1, \ldots, C_n\} \setminus D \cup \{\top\}$
  $E :=$ compute-merge-set(D)
  $F := \sqcap_{G \in \bar{D}} G$
  $\bigcup_{H \in E}$ merging-rule-completions$(F \sqcap H)$
end if

the existence of at least one element in D and $\bar{D}$. The function *compute-merge-set(D)* returns the set of merging rule completions given a set D of existential and universal role quantifications which is interpreted as a concept conjunction of D's elements. The variable F takes a concept representing the conjunction of all concept components which do not influence the merging rule completion process (i.e., the elements of $\bar{D}$). Then the conjunction of each merging rule completion and the concept stored in F is built. Since these conjunctions may contain existential or universal role quantifications, we have to apply the algorithm recursively on each conjunction and return the union of the partial results.

Theorem 3 *Let C be an $\mathcal{ALENR}$ concept. Then algorithm* merging-rule-completions(C) *returns the set of merging rule completions M_C.* □

Example 6 *Let us compute* merging-rule-completions(X_4) *for our example concept X_4. Since X_4 is in SNF, we have $C = X_4$ and the algorithm reaches the part handling concept conjunctions. We get $D = \{X_{41}, \ldots, X_{45}, \forall U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4), \top\}$ and $\bar{D} = \{\top\}$. Furthermore, after calling* compute-merge-set(D), *$E = \{W_2, W_3\}$ contains C's merging rule completions. Since $\bar{D} = \{\top\}$, we obtain $F = \top$ and, finally, $\{\top \sqcap W_2, \top \sqcap W_3\}$ is returned because further recursive calls of the function* merging-rule-completions *only return their arguments as a set.*

Given an $\mathcal{ALENR}$ concept C, for each of C's merging rule completions, we can now determine the numbers $\tilde{e}_i$.

Definition 12 *Let C be an $\mathcal{ALENR}$ concept given by Equation (1) and $M_C = \{D_1, \ldots, D_q\}$ be the set of C's merging rule completions with $D_t := A_t \sqcap \sqcap_{1 \leq j \leq n_t} \exists\, R_{jt}.C_{jt} \sqcap \sqcap_{1 \leq k \leq m_t} \forall\, R_{kt}.C_{kt}$ and A_t an arbitrary conjunction of atomic concepts, negated atomic concepts, $\geq$- and $\leq$-restrictions. Then, for $i \in \{1, \ldots, n\}$, we define*

$$\begin{aligned} \check{e_{it}} &:= \sharp\{\exists\, R.C \in \{\exists\, R_{1t}.D_{1t}, \ldots, \exists\, R_{n_t t}.D_{n_t t}\} | R \text{ is a subrole of } R_i\} \\ \check{e_i} &:= \min\{\check{e_{it}} | t \in \{1, \ldots, q\}\} \end{aligned}$$

First, for a concept C, each value of $\check{e_{it}}$ is determined by counting those existential role quantifications of D_t involving a role R_{jt} ($j \in \{1, \ldots, n_t\}$) which is subsumed by R_i (occurring in C). The reason is that every R_{jt}-successor necessarily is also an R_i-successor if $R_{jt} \sqsubseteq R_i$, and both role successors are unique (because D_t is a merging rule completion). Definition 12 can straightforwardly be implemented into an algorithm taking the set M_C as argument and returning the numbers $\check{e}_i$. With these considerations we can state:

Theorem 4 *Let C be an $\mathcal{ALENR}$ concept and $\hat{C}$ be C's $\geq$-completion given by Equation (3). Then, for all $i \in \{1, \ldots, n\}$:*

$$\tilde{e_i} = \check{e}_i.$$

Proof. Let $M_C = \{D_1, \ldots, D_q\}$ be the set of C's merging rule completions. Assume $\exists i \in \{1, \ldots, n\} : \tilde{e_i} \neq \check{e}_i$. In case $\tilde{e_i} < \check{e}_i$ there exists $t \in \{1, \ldots, q\}$ such that $\check{e}_{it} < \tilde{e_i}$ according to Definition 12. But then there exists a concept C' such that $C \stackrel{m}{\sqsubset\!\!\rightarrow} \cdots \stackrel{m}{\sqsubset\!\!\rightarrow} D_t \stackrel{m}{\sqsubset\!\!\rightarrow} C'$. Hence, D_t is not merging rule complete which is a contradiction since $D_t \in M_C$. Now assume $\tilde{e_i} > \check{e}_i$. According to Definition 12 there exists $t \in \{1, \ldots, q\}$ such that $\check{e}_{it} < \tilde{e_i}$. But then there exists a concept C' such that $C \stackrel{m}{\sqsubset\!\!\rightarrow} \cdots \stackrel{m}{\sqsubset\!\!\rightarrow} C' \stackrel{m}{\sqsubset\!\!\rightarrow} \cdots \stackrel{m}{\sqsubset\!\!\rightarrow} D_t$. However, since $\check{e}_{it} < \tilde{e_i}$, the merging rule cannot be applied to C' because condition (iv) is violated in Definition 11. This leads to a contradiction. □

Theorem 4 shows that, for a concept C, the numbers $\tilde{e_i}$ can be determined by the previously sketched procedure.

Example 7 *In our example, for $M_C = \{W_2, W_3\}$, we obtain*

$$\begin{array}{ll} W_2: & \check{e_{11}} = 4, \check{e_{21}} = 3, \check{e_{31}} = 1, \check{e_{41}} = 2, \check{e_{51}} = 1, \text{ and} \\ W_3: & \check{e_{12}} = 3, \check{e_{22}} = 2, \check{e_{32}} = 1, \check{e_{42}} = 2, \check{e_{52}} = 1. \end{array}$$

According to Definition 12, for $i \in \{1, \ldots, 5\}$, we choose $\check{e}_i = \min(\{e_{i1}, e_{i2}\})$ yielding $\check{e_1} = 3, \check{e_2} = 2, \check{e_3} = 1, \check{e_4} = 2, \check{e_5} = 1$.

Having computed $\check{e}_i$, Theorem 4 can be used in order to obtain $\tilde{e_i}$. Then C's $\geq$-completion $\hat{C}$ can be easily constructed by a recursive algorithm.

Example 8 *For our example concept X_4, we get*

$$\tilde{L} \;=\; (\geq\, 3\, R) \sqcap (\geq\, 2\, S_1) \sqcap (\geq\, 1\, S_2) \sqcap (\geq\, 2\, T) \sqcap (\geq\, 1\, U)$$

and, thus,

$$\begin{array}{rcl} \hat{X_4} & = & \sqcap_{1 \leq i \leq 6} X_{4i} \sqcap \tilde{L} \\ & = & \exists\, R.A_1 \sqcap \exists\, S_1.(\neg A_1 \sqcap A_3) \sqcap \exists\, S_2.A_2 \sqcap \\ & & \exists\, T.A_4 \sqcap \exists\, U.A_5 \sqcap \forall\, U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4) \sqcap \\ & & (\geq\, 3\, R) \sqcap (\geq\, 2\, S_1) \sqcap (\geq\, 1\, S_2) \sqcap (\geq\, 2\, T) \sqcap (\geq\, 1\, U) \end{array}$$

as desired.

6 The LCS algorithm

Once the $\geq$-completions $\hat{C}$ and $\hat{D}$ are computed for concepts C and D, Definition 8 can straightforwardly be implemented into a corresponding algorithm taking $\hat{C}$ and $\hat{D}$ as arguments. By Proposition 2 we have an easy formula for the correct treatment of concept conjunctions.

When called with $\geq$-completions $\hat{C}$ and $\hat{D}$ of $\mathcal{ALEN}\mathcal{R}$ or $\mathcal{ALE}\mathcal{Q}$ concepts C and D, Algorithm 3 recursively computes $lcs(\hat{C}, \hat{D})$ given $\leq$-completions $\hat{C}$ and $\hat{D}$ of $\mathcal{ALEN}\mathcal{R}$ or $\mathcal{ALE}\mathcal{Q}$ concepts C and D. It is a straightforward implementation of the LCS rules given by Table 1.

Theorem 5 *Algorithm* compute-lcs *is sound, complete, and terminates. In other words, if C, D are $\mathcal{ALEN}\mathcal{R}$ or $\mathcal{ALE}\mathcal{Q}$ concepts and $\hat{C}, \hat{D}$ their corresponding $\geq$-completions, then* compute-lcs$(\hat{C}, \hat{D}) = lcs(C, D)$. □

Example 9 *Let*

$$\begin{aligned} Y_3 &:= (\geq\ 4\, R\, \top) \text{ and} \\ Y_4 &:= (\geq\ 5\, R). \end{aligned}$$

Then $\hat{Y_3} = Y_3$ and $\hat{Y_4} = Y_4$. Applying Algorithm 3 to $\hat{X_3}$ and $\hat{Y_3}$ yields $(\geq\ 3\, R\, \top)$ and to $\hat{X_4}$ and $\hat{Y_4}$ yields $(\geq\ 3\, R)$ as desired.

We can state a lower bound complexity for computing the LCS of two concepts.

Theorem 6 *Let C and D be either both $\mathcal{ALEN}\mathcal{R}$ or $\mathcal{ALE}\mathcal{Q}$ concepts. Then $lcs(C, D)$ may be of size exponential in the size of C and D.*

Proof. In [2], this statement is shown for C and D being $\mathcal{ALE}$ concepts. Since $\mathcal{ALE}$ is a sublanguage of both $\mathcal{ALE}\mathcal{Q}$ and $\mathcal{ALEN}\mathcal{R}$, the result can be considered a lower bound complexity for computing the LCS of $\mathcal{ALE}\mathcal{Q}$ and $\mathcal{ALEN}\mathcal{R}$ concepts. □

7 Conclusion and Future Work

We have presented an LCS operator for the expressive DLs $\mathcal{ALEN}\mathcal{R}$ and $\mathcal{ALE}\mathcal{Q}$ and showed first decidability results. To the best of our knowledge, it is the first approach towards an LCS algorithm for a description logic containing both number restrictions and existential quantifications. Both languages are expressive enough to be quite useful in our similarity-based

Algorithm 3 compute-lcs (C, D)

if $D \sqsubseteq C$ **then**
  C
else if $C \sqsubseteq D$ **then**
  D
else if $(C = A \vee C = \neg A)$ **and** $(D = B \vee D = \neg B)$ for atoms A and B **then**
  if $C = D$ **then**
    C
  else
    $\top$
  end if
else if $C = C_1 \sqcap \cdots \sqcap C_n$ **then**
  $\sqcap_{1 \leq i \leq n}$compute-lcs(C_i, D)
else if $C = \exists\, R.C'$ **and** $D = \exists\, S.D'$ **then**
  if $msr(R, S) = \top$ **then**
    $\top$
  else
    $\exists\, msr(R, S)$.compute-lcs(C', D')
  end if
else if $C = \forall\, R.C'$ **and** $D = \forall\, S.D'$ **then**
  if $msr(R, S) = \top$ **then**
    $\top$
  else
    $\forall\, msr(R, S)$.compute-lcs(C', D')
  end if
else if $C = (\geq\ n\ R)$ **and** $D = (\geq\ m\ S)$ **then**
  if $msr(R, S) = \top$ **then**
    $\top$
  else
    $(\geq\ \min\{n, m\}\ msr(R, S))$
  end if
else if $C = (\leq\ n\ R)$ **and** $D = (\leq\ m\ S)$ **then**
  if $msr(R, S) = \top$ **then**
    $\top$
  else
    $(\leq\ \max\{n, m\}\ msr(R, S))$
  end if
else if $C = (\geq\ n\ R\ C')$ **and** $D = (\geq\ m\ R\ D')$ **then**
  $(\geq\ \min\{n, m\}\ R$ compute-lcs$(C', D'))$
else if $C = (\leq\ n\ R\ C')$ **and** $D = (\leq\ m\ R\ D')$ **then**
  $(\leq\ \max\{n, m\}\ R\ (C' \sqcap D'))$
else if $D = D_1 \sqcap \cdots \sqcap D_n$ **then**
  compute-lcs(D, C)
else
  $\top$
end if

information retrieval application. The special challenge one faces is that conjunctions of qualified number restrictions (in $\mathcal{ALEQ}$) and of existential role quantifications (in $\mathcal{ALENR}$) imply $\geq$-restrictions which have an effect on the LCS determination. In our approach, we first describe an algorithm to compute the $\geq$-completions of the input concepts. The $\geq$-completions include all implicitly contained information relevant for computing the LCS in a simple way. We extended the LCS computation algorithm for the DLs $\mathcal{ALN}$ and $\mathcal{ALE}$ to the new language constructs included in $\mathcal{ALENR}$ and $\mathcal{ALEQ}$ and showed soundness and completeness of our LCS operator. It would be an interesting future research topic to extend the notion of concept commonalities to DLs including a disjunction operator. The LCS operator is not useful for this because the LCS of two concepts would just be their disjunction which does not express meaningful commonalities. Also, we have not yet performed a detailed complexity analysis for LCS determination in $\mathcal{ALENR}$ and $\mathcal{ALEQ}$.

References

[1] F. Baader and R. Küsters. Computing the Least Common Subsumer and the Most Specific Concept in the Presence of Cyclic $\mathcal{ALN}$-concept Descriptions. In O. Herzog and A. Günter, editors, *Proc. of the 22nd KI-98*, volume 1504, pages 129–140, 1998.

[2] F. Baader, R. Küsters, and R. Molitor. Computing Least Common Subsumer in Description Logics with Existential Restrictions. LTCS-Report 98-09, LuFG, RWTH Aachen, Germany, 1998.

[3] W. W. Cohen, A. Borgida, and H. Hirsh. Computing Least Common Subsumers in Description Logics. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1036–1043, Japan, 1992. Ass. for Computing Machinery.

[4] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The Complexity of Concept Languages. *Information and Computation*, 134(1):1–58, April 1997.

[5] 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.

[6] B. Hollunder and F. Baader. Qualifying Number Restrictions in Concept Languages. In *Proceedings of the Second International Conference on*

Principles of Knowledge Representation and Reasoning, pages 335–346, San Mateo, CA, USA, April 1991.

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

[8] T. Mantay and R. Möller. Content-based Information Retrieval by Computing Least Common Subsumers in a Probabilistic Description Logic. In *Proceedings of the ECAI Workshop Intelligent Information Integration*, Brighton, 1998.

[9] R. Möller, V. Haarslev, and B. Neumann. Semantics-based Information Retrieval. In *International Conference on Information Technology and Knowledge Systems*, Vienna, Budapest, 1998.

[10] B. Owsnicki-Klewe. A Cardinality-Based Approach to Incomplete Knowledge. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 491–496, Stockholm, Sweden, 1990.