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, the "bottom-up" construction of knowledge bases, learning tasks, and for specific kinds of information retrieval. So far, computing the least common subsumer has been restricted to description logics with rather limited expressivity. In this article, we continue recent research on extending this operation to more complex languages and present a least common subsumer operator for the expressive description logics $\mathcal{ALENR}$ and $\mathcal{ALQ}$.

1 Introduction

The least common subsumer (LCS) is an important reasoning service useful for a number of applications. Cohen et al. [3] consider an LCS operator for the DL $\mathcal{ALN}$ in order to approximate a disjunction operator which is not explicitly present in $\mathcal{ALN}$. Baader et al. use the operator for the "bottom-up" construction of KBs based on the DLs $\mathcal{ALN}$ with cyclic concept definitions [1] and $\mathcal{ALE}$ [2]. In [5], Möller et al. apply the operator to similarity-based information retrieval where the LCS serves as a formalization of the notion of "similarities" of concepts. As recent literature shows, there is a tendency to extend this reasoning service to more and more expressive DL languages.

The main contribution of this paper is the proposal of an LCS operator for the expressive DLs $\mathcal{ALENR}$ and $\mathcal{ALQ}$ consisting of the top and bottom concept, atoms, negations of atoms, concept conjunctions augmented by either $\exists$-, $\forall$-quantifications, role conjunctions, and number restrictions ($\mathcal{ALENR}$) or qualified number restrictions ($\mathcal{ALQ}$). Cohen, Borgida, and Hirsh [3] showed that if subsumption between two concepts C and D can be computed structurally, their LCS can be determined by a simple recursive algorithm. We will show that subsumption between $\mathcal{ALQ}$ or $\mathcal{ALENR}$ concepts can be computed structurally if they are first transformed into a normal form. Therefore, the main object of this article is to give a definition of such a normal form and provide algorithms to compute it for both considered DLs. Technical details of the presented work and proofs of the Propositions and Theorems are available in [4].

2 LCS Definition

The LCS of two $\mathcal{ALENR}$ and $\mathcal{ALQ}$ concepts C and D is defined as the set of most specific concepts which subsume both C and D.

Definition 1 *Let C and D be both either $\mathcal{ALENR}$ or $\mathcal{ALQ}$ 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. Due to this uniqueness property, we will consider $lcs(C,D)$ as a concept rather than a set of concepts in the following. Definition 1 can straightforwardly be extended to n arguments. In this case, lcs is associative and commutative and $lcs(C_1,\ldots,C_n) = lcs(C_1, lcs(C_2,\ldots lcs(C_{n-1},C_n)\ldots))$. In our analysis, it will be convenient to define the most specific role of two role conjunctions possibly occurring in $\mathcal{ALENR}$ concepts.

Definition 2 *Let $R = \sqcap_{1\leq i\leq n} R_i$ and $S = \sqcap_{1\leq j\leq m} S_j$ be roles, $\mathfrak{R} = \{R_1,\ldots,R_n\}$, and $\mathfrak{S} = \{S_1,\ldots,S_m\}$. Then we define the* most specific role *of R and S as a partial function:*

$$msr(R,S) := \begin{cases} \sqcap_{T\in\mathfrak{R}\cap\mathfrak{S}} T & \text{if } \mathfrak{R}\cap\mathfrak{S} \neq \emptyset, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Definition 2 can easily be extended to more than two role arguments. In the next section, we will define a normal form for concepts with which the subsumption problem for $\mathcal{ALQ}$ and $\mathcal{ALENR}$ can be decided by a structural subsumption algorithm.

3 Defining a Normal Form for Structural Subsumption

We are interested in a normal form for $\mathcal{ALQ}$ and $\mathcal{ALENR}$ concepts with which the subsumption relation between concepts can be decided by structural comparisons. First, we will sketch a structural subsumption algorithm. Note that Algorithm 1 does not necessarily return the correct result for *arbitrary* $\mathcal{ALQ}$ or $\mathcal{ALENR}$ concepts.

Algorithm 1 structural-subsumption(C, D)

Let $C = C_1 \sqcap \cdots \sqcap C_n$ and $D = D_1 \sqcap \cdots \sqcap D_m$. Then $D \sqsubseteq C$ iff $C = \top$ or $D = \bot$ or, for all C_i:

1. if C_i is an atomic concept, then there exists a D_j such that $C_i = D_j$,
2. if C_i is of the form $\exists\, R.C'$ $(\forall\, R.C')$, then there exists a D_j of the form $\exists\, S.D'$ $(\forall\, S.D')$ such that S (R) is a subrole of R (S) and $D' \sqsubseteq C'$,
3. if C_i is of the form $(\geq\ n\ R\ C')$ $[(\leq\ n\ R\ C')]$, then there exists a D_j of the form $(\geq\ m\ R\ D')$ $[(\leq\ m\ R\ D')]$ such that $n \leq m$ $(m \leq n)$ and $D' \sqsubseteq C'$ $(C' \sqsubseteq D')$,
4. if C_i is of the form $(\geq n\, R)$ $[(\leq n\, R)]$, then there exists a D_j of the form $(\geq\ m\, S)$ $[(\leq\ m\, S)]$ such that $n \leq m$ $(m \leq n)$ and S (R) is a subrole of R (S).

Based on Algorithm 1 we can define the structural subsumption normal form for concepts.

Definition 3 *Let $C, \hat{C}, D$ and $\hat{D}$ be either $\mathcal{ALQ}$ or $\mathcal{ALENR}$ concepts. Then $\hat{C}$ and $\hat{D}$ are in* structural subsumption normal form *(SSNF) of C and D, respectively, iff $\hat{C} \equiv C$, $\hat{D} \equiv D$, and* structural-subsumption$(\hat{C}, \hat{D})$ *returns* true *iff $D \sqsubseteq C$. $SSNF(C)$ [$SSNF(D)$] denotes the set of all concepts in SSNF of C (D).*

In the next section, we will develop algorithms for computing structural subsumption normal forms for $\mathcal{ALQ}$ and $\mathcal{ALENR}$ concepts.

4 Structural Subsumption Normal Forms

In order to transform an $\mathcal{ALQ}$ concept into SSNF, qualified number restrictions occurring in concept conjunctions require a special treatment. For example, let $X_1 := (\geq\ 2\ R\ A)$ and $Y_1 := (\geq\ 1\ R\ (A \sqcap B)) \sqcap (\geq 1\ R\ (A \sqcap \neg B))$. Then *structural-subsumption*(X_1, Y_1) returns *false*, even though $Y_1 \sqsubseteq X_1$ holds. The reason for $Y_1 \sqsubseteq X_1$ is that from Y_1 it follows that the two R-successors required to satisfy the two qualified $\geq$-restrictions must be unique and must be instances of A. Thus, $(\geq\ 2\, Rlcs(A \sqcap B, A \sqcap \neg B)) \equiv (\geq\ 2\, R\, A)$ follows from Y_1.

When determining the subsumption relation between $\mathcal{ALENR}$ concepts, two kinds of possible "interactions" between concept forming operators must be considered: $\exists$-quantifications in combination with $\forall$-quantifications as well as $\geq$-restrictions in combination with $\forall$-quantifications. We proceed with an example showing the former kind of interaction. Let $X_2 := (\geq 3\, R)$ and $Y_2 := \exists\, R.A_1 \sqcap \exists\, (R \sqcap S).\neg A_1$. Then *structural-subsumption*(X_2, Y_2) returns *false*, even though $Y_2 \sqsubseteq X_2$. The reason is that (conjunctions of) $\exists$-quantifications always imply $\geq$-restrictions. For instance, $\exists\ R.A_1$ implies $(\geq\ \ 1\ R)$ and $\exists\ R.A_1 \sqcap \exists\ (R \sqcap S).\neg A_1$ implies $(\geq\ 2\ R) \sqcap (\geq\ 1\, (R \sqcap S))$.

The following example demonstrates interactions between $\geq$-restrictions and $\forall$-quantifications. Let $X_3 := (\geq 5\, R)$ and $Y_3 := (\geq 1 (R \sqcap S)) \sqcap (\geq 2 (R \sqcap T)) \sqcap (\geq 3 (R \sqcap U)) \sqcap \forall S.A_1 \sqcap \forall T.A_2 \sqcap \forall U.(\neg A_1 \sqcap \neg A_2)$. Then *structural-subsumption*(X_3, Y_3) returns *false*, even though $Y_3 \sqsubseteq X_3$. The reason is that Y_3 implies $(\geq 5\, R)$ since, due to $\forall T.A_2$ and $\forall\, U.(\neg A_1 \sqcap \neg A_2)$, the two $R \sqcap T$-successors required for satisfying $(\geq\ 2\, (R \sqcap T))$ are different from the three $R \sqcap U$-successors required for satisfying $(\geq\ 3\, (R \sqcap U))$.

The reason for *structural-subsumption* to fail on the above three examples is that (qualified) $\geq$-restrictions following from interactions of certain concept forming operators are not explicitly present in the example concepts Y_1, Y_2, and Y_3. Therefore, in order to transform an $\mathcal{ALENR}$ ($\mathcal{ALQ}$) concept into SSNF, the idea is to first make this implicitly contained information explicit and conjunctively add it to C in the form of additional (qualified) $\geq$-restrictions. For instance, in the second example, if we add $(\geq\ 2\ R) \sqcap (\geq\ 1\ (R \sqcap S))$ as a conjunct to Y_2 yielding $\hat{Y_2}$, *structural-subsumption*$(X_2, \hat{Y_2})$ returns *true* as desired. The examples also show that two concepts can be transformed into SSNF independently from each other. Therefore, we will restrict the following presentation of the transformation algorithm to *one* concept instead of a pair of concepts.

4.1 Computing SSNFs in $\mathcal{ALQ}$

In order to transform an $\mathcal{ALQ}$ concept into SSNF, we assume the concept to be in sorted normal form. An $\mathcal{ALQ}$ concept C is in *sorted normal form* (SNF) iff

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

where A is an arbitrary conjunction of atoms and negated atoms and C_i and $C_j^{'}$ are also in SNF. Furthermore, we assume all nested conjunctions to be flattened, i.e., $A \sqcap (B \sqcap C) \longrightarrow A \sqcap B \sqcap C$.

Proposition 1 *Let C be an $\mathcal{ALQ}$ concept in SNF given by (1). If $C = \top, C = \bot$, or $C = A$, we define $\hat{C} := C$. Otherwise*

$$\begin{aligned} \hat{C} &:= A \sqcap \hat{L} \sqcap \hat{M} \sqcap \tilde{L_Q} \text{ with} \\ \tilde{L_Q} &:= \sqcap_{1 \le i \le n}(\ge \tilde{l_i}\ R_i\ lcs(C_1, \ldots, C_n)) \end{aligned} \tag{2}$$

where $\forall i \in \{1, \ldots, n\} : \tilde{l_i} = \min(\{l \in \mathbb{N} | L \sqcap M \sqcap (\le l\ R_i\ lcs(C_1, \ldots, C_n))$ *is consistent*$\})$. *Then*
$\hat{L} \in SSNF(L) \wedge \hat{M} \in SSNF(M) \Longrightarrow \hat{C} \in SSNF(C)$.

Proposition 1 states that we get a concept $\hat{C}$ in SSNF of an $\mathcal{ALQ}$ concept C if we add the qualified $\ge$-restrictions in $\tilde{L_Q}$ to C and repeat this process recursively for L and M.

According to Proposition 1, in order to determine a concept $\hat{C}$ in SSNF of an $\mathcal{ALQ}$ concept C given by (1), the numbers $\tilde{l_i}$ must be computed for every role R_i occuring on every depth of C. For simplicity of presentation, we will consider a fixed concept depth. The value of each $\tilde{l_i}$ does only depend on L and M. For computing the parameters $\tilde{l_i}$ for each $i \in \{1, \ldots, n\}$, we define $N_i := \{p_k | R_k = R_i \wedge 1 \le k \le n\}$ and observe that lower and upper bounds for $\tilde{l_i}$ are given by

$$\tilde{l_i^{min}} := \min(N_i) \le \tilde{l_i} \le \sum_{p \in N_i} p =: \tilde{l_i^{max}}.$$

Thus, $\tilde{l_i}$ can be determined by a binary search algorithm which takes $\log_2(\tilde{l_i^{max}} - \tilde{l_i^{min}}) + 1$ satisfiability tests in the worst case.

For instance, computing $\hat{Y_1}$ of our example concept Y_1, we have $\tilde{l_i^{min}} = 1$, $\tilde{l_i^{max}} = 2$ for $i \in \{1, 2\}$, and the binary search algorithm for computing $\tilde{l_i}$ returns 2. Hence, $\tilde{L_Q} = (\ge 2\,R\,lcs(A \sqcap B, A \sqcap \neg B)) \sqcap (\ge 2\,R\,lcs(A \sqcap B, A \sqcap \neg B)) \equiv (\ge\ 2\ R\ A)$ and $\hat{Y_1} \equiv Y_1 \sqcap (\ge\ 2\ R\ A)$. Since $\hat{X_1} = X_1$, we can apply *structural-subsumption* to X_1 and $\hat{Y_1}$ yielding *true* as desired. We will now deal with the problem of computing $\hat{C} \in SSNF(C)$ for an $\mathcal{ALENR}$ concept C.

4.2 Computing SSNFs in $\mathcal{ALENR}$

An $\mathcal{ALENR}$ concept C is in SNF iff

$$\begin{aligned} &C = \top \vee C = \bot \vee C = A \sqcap E \sqcap F \sqcap L \text{ with} \\ E &= \sqcap_{1 \le i \le n} \exists\, R_i.C_i \text{ and} \\ F &= \sqcap_{1 \le j \le m} \forall\, R_j^{'}.C_j^{'} \\ L &= \sqcap_{1 \le k \le q}(\ge\ n_k\ S_k) \end{aligned} \tag{3}$$

where A is an arbitrary conjunction of atoms, negated atoms, and $\le$-restrictions and C_i and $C_j^{'}$ are also in SNF. Furthermore, we assume all nested conjunctions to be flattened and conjunctions of $\forall$-quantifications are factorized, i.e., $\forall\, R.C \sqcap \forall\, R.D \rightarrow \forall\, R.(C \sqcap D)$. The example concepts Y_2 and Y_3 showed that we have to consider interactions between $\exists$- and $\forall$-quantifications as well as interactions between $\ge$-restrictions and $\forall$-quantifications. Let

$$\begin{aligned} \mathfrak{L} &:= \{(\ge\ n_k\ S_k) | 1 \le k \le q\}, \\ \mathfrak{P} &:= \{P \in 2^{\mathfrak{L}} | \sharp P \ge 2\}, \\ \mathfrak{R} &:= \{P \in \mathfrak{P} | P = \{(\ge\ n_{k_1}\ S_{k_1}), \ldots, (\ge\ n_{k_r}\ S_{k_r})\} \wedge \\ &\quad\ msr(S_{k_1}, \ldots, S_{k_r}) \text{ is defined}\}. \end{aligned}$$

The set $\mathfrak{P}$ contains all subsets of $\mathfrak{L}$ with at least two (unique) $\ge$-restrictions. In $\mathfrak{R}$ we collect all sets of $\mathfrak{P}$ from whose elements further $\ge$-restrictions relevant for the transformation of C into a concept in SSNF can be derived.

Proposition 2 *Let C be an $\mathcal{ALENR}$ concept given by (3). If $C = \top, C = \bot$, or $C = A$, we define $\hat{C} := C$. Otherwise*

$$\begin{aligned} \hat{C} &:= A \sqcap \hat{E} \sqcap \hat{F} \sqcap \tilde{E} \sqcap \tilde{L} \text{ with} \\ \tilde{E} &:= \sqcap_{1 \le i \le n}(\ge\ \tilde{e_i}\ R_i) \text{ and} \\ \tilde{L} &:= \sqcap_{P \in \mathfrak{R}}(\ge\ \tilde{l_P}\ msr(S_{k_1}, \ldots, S_{k_r})) \\ &\quad\ \text{with } P = \{(\ge\ n_{k_1}\ S_{k_1}), \ldots, (\ge\ n_{k_r}\ S_{k_r})\} \end{aligned} \tag{4}$$

where $\forall i \in \{1, \ldots, n\} : \tilde{e_i} = \min(\{e \in \mathbb{N} | E \sqcap F \sqcap L \sqcap (\le e\,R_i)$ *is consistent*$\})$, $\forall P \in \mathfrak{R} : \tilde{l_P} = \min(\{l \in \mathbb{N} | E \sqcap F \sqcap L \sqcap (\le\ l\ msr(S_{k_1}, \ldots, S_{k_r}))$ *is consistent*$\})$. *Then*

$\hat{E} \in SSNF(E) \wedge \hat{F} \in SSNF(F) \Longrightarrow \hat{C} \in SSNF(C)$.

Proposition 2 states that we get a concept $\hat{C}$ in SSNF of an $\mathcal{ALENR}$ concept C if we add the $\ge$-restrictions in $\tilde{E}$ and $\tilde{L}$ to C and repeat this process recursively for E and F.

We first deal with the determination of $\tilde{E}$ for which the numbers $\tilde{e_i}$ must be computed for every role R_i occuring on every depth of C. Again, for simplicity of presentation, we will consider a fixed concept depth. The value of each $\tilde{e_i}$ does only depend on E and F. For computing $\tilde{e_i}$, the idea is to build a set of new concepts $M_C = \{\check{C_1}, \ldots, \check{C_p}\}$ from C such that $\tilde{e_i}$ can easily be derived from M_C's elements. For example, let $S_1 := R \sqcap S', S_2 := R \sqcap S'', T := S_1 \sqcap T', U := T \sqcap U'$ be roles and $Y_4 := \sqcap_{1 \le i \le 6} Y_{4i} = \exists\, R.A_1 \sqcap \exists\, S_1.(\neg A_1 \sqcap A_3) \sqcap \exists\, S_2.A_2 \sqcap \exists\, T.A_4 \sqcap \exists\, U.A_5 \sqcap \forall\, U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4)$. For computing $\hat{Y_4}$, 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 consistent or if a $\forall$-quantification of the form $\forall\, S_2.\neg A_1$ was present. We will model the merging process as the result of an application of a *merging rule* to Y_4 yielding a new concept which only differs from Y_4 in that two $\exists$-quantifications are merged. 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. Merging rule applications are not deterministic in general. For instance, when applied to Y_4, we can either merge the conjuncts $\exists\, R.A_1$ and $\exists\, S_2.A_2$ yielding $W_1 := \exists\, S_1.(\neg A_1 \sqcap A_3) \sqcap \exists\, S_2.(A_2 \sqcap A_1) \sqcap \exists\, T.A_4 \exists\, U.A_5 \sqcap \forall U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4)$ or we can merge $\exists R.A_1$ and $\exists T.A_4$ yielding $W_2 := \exists S_1.(\neg A_1 \sqcap A_3) \sqcap \exists S_2.A_2 \sqcap \exists T.(A_4 \sqcap A_1) \sqcap \exists\, U.A_5 \sqcap \forall\, U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4)$. 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 $W_3 := \exists\, S_2.(A_2 \sqcap A_1) \sqcap \exists T.(A_4 \sqcap \neg A_1 \sqcap A_3) \sqcap \exists U.A_5 \sqcap \forall U.(\neg A_1 \sqcap \neg A_3 \sqcap \neg A_4)$ and, thus, $M_{Y_4} = \{W_2, W_3\}$. Obviously, we have $W_2 \sqsubseteq Y_4$ and $W_3 \sqsubseteq Y_4$ and, therefore, all models of W_2 and all models of W_3 are also models of Y_4. From W_3 (W_2) we can now easily derive that, for instance, at least three (four) distinct R-successors are present in every model of Y_4. We will choose the minimum of the two values for $\tilde{e_1}$ because, as W_3 shows, there is a model of Y_4 which has only three R-successors.

Definition 4 *For an $\mathcal{ALENR}$ concept C in SNF given by Equation (3) we say that C' emerges from C by application of the* merging rule *$(C \xrightarrow{m} C')$ iff $\exists k, k' \in \{1, \ldots, n\}, k < k'$, with*

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

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

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

(iv) C' is consistent.

C' emerges from C by successive applications of the merging rule *iff $\exists C_1, \ldots, C_r$, $r \in I\!N \cup \{0\}$, such that $C = C_1 \xrightarrow{m} \cdots \xrightarrow{m} C_r = C'$. C' is* merging rule complete w.r.t. *C iff C' emerges from C by successive applications of the merging rule and $\neg\exists C'' : C' \xrightarrow{m} C''$. 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 4, we can apply the merging rule to an $\mathcal{ALENR}$ concept C (yielding C') if C contains two conjunctive $\exists$-quantifications involving roles R_k and $R_{k'}$ where $R_{k'}$ is a subrole of R_k and the resulting concept C' is consistent. In [4] we give a recursive algorithm to compute the set of merging rule completions of an $\mathcal{ALENR}$ concept.

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

Definition 5 *Let C be an $\mathcal{ALENR}$ concept and $M_C = \{D_1, \ldots, D_p\}$ be the set of C's merging rule completions with $D_t := A_t \sqcap \sqcap_{1 \le i \le n_t} \exists\, R_{it}.E_{it}$ and A_t an arbitrary conjunction of atoms, negated atoms, $\ge$-, $\le$-restrictions, and $\forall$-quantifications. Then, for $i \in \{1, \ldots, n\}$, we define*

$$\check{e_{it}} := \sharp\{\exists\, R.C \in \{\exists\, R_{1t}.E_{1t}, \ldots, \exists\, R_{n_t t}.E_{n_t t}\} | \ R \text{ is a subrole of } R_i\}$$

$$\check{e_i} := \min\{\check{e_{it}} | t \in \{1, \ldots, q\}\}.$$

First, the value of $\check{e_{it}}$ is determined by counting those $\exists$-quantifications of D_t involving a role R which is subsumed by R_i. The reason is that every R-successor necessarily is also an R_i-successor if R is a subrole of R_i, and both role successors are unique (because D_t is a merging rule completion). In our example, for $M_{Y_4} = \{W_2, W_3\}$, we obtain $\check{e_{11}} = 4$, $\check{e_{21}} = 3$, $\check{e_{31}} = 1$, $\check{e_{41}} = 2$, and $\check{e_{51}} = 1$ for W_2 and $\check{e_{12}} = 3$, $\check{e_{22}} = 2$, $\check{e_{32}} = 1$, $\check{e_{42}} = 2$, and $\check{e_{52}} = 1$ for W_3. According to Definition 5, for $i \in \{1, \ldots, 5\}$ we choose $\check{e_i} = \min\{e_{i1}, e_{i2}\}$ yielding $\check{e_1} = 3, \check{e_2} = 2, \check{e_3} = 1, \check{e_4} = 2, \check{e_5} = 1$. Definition 5 can straightforwardly be implemented into an algorithm taking the set M_C as argument and returning the numbers $\check{e_i}$.

Proposition 3 *Let C be an $\mathcal{ALENR}$ concept in SNF given by (3) and $\tilde{e_i}$ be defined as in Proposition 2. Then, for all $i \in \{1, \ldots, n\} : \tilde{e_i} = \check{e_i}$.*

Having computed $\tilde{e_i}$, the additional $\ge$-restrictions $\tilde{E}$ following from the $\exists$-quantifications E can be easily constructed by a recursive algorithm. For our example concept Y_4, we get $\tilde{E} = (\ge\, 3\, R) \sqcap (\ge\, 2\, S_1) \sqcap (\ge\, 1\, S_2) \sqcap (\ge\, 2\, T) \sqcap (\ge\, 1\, U)$. Subsequently, an $\mathcal{ALENR}$ concept in SNF given by (3) for which $\tilde{E}$ has been added as a conjunct will be called $\exists$-*complete*.

Now we will deal with the problem of determining $\tilde{L}$ in Proposition 2. For an $\exists$-complete $\mathcal{ALENR}$ concept C in SNF given by (3), let $\mathfrak{A} := \{\forall\, R_j^{'}.C_j^{'} | 1 \le j \le m\}$. Then, for each $P \in \mathfrak{R}$ with $P := \{(\ge\, n_{k_1}\, S_{k_1}), \ldots, (\ge\, n_{k_r}\, S_{k_r})\}$, we define

$$\tilde{\mathfrak{L}}^P := \begin{cases} (\ge\, \sum_{1 \le i \le r} n_{k_i}\, msr(S_{k_1}, \ldots, S_{k_r})) \\ \quad \text{if } \forall i, j \in \{1, \ldots, r\}, i \ne j : \\ \quad\quad \exists (\forall\, R.C), (\forall\, R'.C') \in \mathfrak{A} : \\ \quad\quad msr(R, S_{k_i}) \text{ and } msr(R', S_{k_j}) \text{ are} \\ \quad\quad \text{defined and } C \sqcap C' \text{ is inconsistent,} \\ \top \quad \text{otherwise.} \end{cases}$$

Intuitively, for each $P \in \mathfrak{R}$, $\tilde{\mathfrak{L}}^P$ indicates the most specific $\ge$-restriction following from the $\ge$-restrictions in P. As suggested by example concept Y_3, for adding an $\ge$-restriction of the form $\tilde{\mathfrak{L}}^P = (\ge \sum_{1 \le i \le r} n_{k_i}\; msr(S_{k_1}, \ldots, S_{k_r}))$ as a conjunct to $\tilde{L}$, we check if for every pair of $\ge$-restrictions $(\ge\; n\, S)$ and $(\ge\; m\, S')$ in $\mathfrak{R}$ there is a pair of $\forall$-quantifications $\forall\, R.C$ and $\forall\, R'.C'$ involving common role conjuncts with S and S', respectively, and $C \sqcap C'$ being inconsistent.

Let us compute $\tilde{\mathfrak{L}}^P$ for $P = \{(\ge\, 1\, (R \sqcap R_1)), (\ge\, 3\, (R \sqcap R_3))\}$. $\forall\, (S \sqcap R_1).A_1$ and $\forall\, (S \sqcap R_3).(\neg A_1 \sqcap \neg A_2)$ are the only elements of $\mathfrak{A}$ which involve roles intersecting with $R \sqcap R_1$ and $R \sqcap R_3$, respectively. Since $A_1 \sqcap (\neg A_1 \sqcap \neg A_2)$ is inconsistent, we have $\tilde{\mathfrak{L}}^P = (\geq (1+3)\, msr(R \sqcap R_1, R \sqcap R_3)) \equiv (\geq 4\, R)$.

Finally, $\check{L}$ consists of the conjunction of all $\mathfrak{L}^P$:

$$\check{L} := \sqcap_{P \in \mathfrak{P}} \mathfrak{L}^P.$$

Proposition 4 *Let C be an $\exists$-complete $\mathcal{ALENR}$ concept and $\tilde{L}$ be defined as in Proposition 2. Then, $\check{L} \equiv \tilde{L}$.*

In our example, we get $\check{L} = \top \sqcap (\geq 4\, R) \sqcap (\geq 5\, R) \equiv (\geq 5\, R)$.

Given two $\mathcal{ALQ}$ ($\mathcal{ALENR}$) concepts C and D, the algorithms sketched in this section enable us to compute concepts $\hat{C}$ and $\hat{D}$ of the form (2) [(4)]. Propositions 1 and 2 guarantee that subsumption between C and D can be computed structurally using $\hat{C}$ and $\hat{D}$. We are now ready to compute the LCS of $\mathcal{ALQ}$ and $\mathcal{ALENR}$ concepts.

5 LCS computation

Given $\mathcal{ALQ}$ or $\mathcal{ALENR}$ concepts C and D and $\hat{C} \in SSNF(C)$ and $\hat{D} \in SSNF(D)$, $lcs(C, D)$ can straightforwardly be implemented into an algorithm taking $\hat{C}$ and $\hat{D}$ as arguments.

Algorithm 2 c-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** $\top$
else if $C = \exists\, R.C'$ **and** $D = \exists\, S.D'$ **then**
  if $msr(R, S)$ is undefined **then** $\top$
  else $\exists\, msr(R, S)$.c-lcs(C', D')
else if $C = \forall\, R.C'$ **and** $D = \forall\, S.D'$ **then**
  if $msr(R, S)$ is undefined **then** $\top$
  else $\forall\, msr(R, S)$.c-lcs(C', D')
else if $C = (\geq\, n\, R)$ **and** $D = (\geq\, m\, S)$ **then**
  if $msr(R, S)$ is undefined **then** $\top$
  else $(\geq\, \min\{n, m\}\, msr(R, S))$
else if $C = (\leq\, n\, R)$ **and** $D = (\leq\, m\, S)$ **then**
  $(\leq\, \max\{n, m\}\, R \sqcap S))$
else if $C = (\geq\, n\, R\, C')$ **and** $D = (\geq\, m\, R\, D')$ **then**
  $(\geq\, \min\{n, m\}\, R$ c-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 $C = C_1 \sqcap \cdots \sqcap C_n$ **then** $\sqcap_{1 \leq i \leq n}$c-lcs(C_i, D)
else if $D = D_1 \sqcap \cdots \sqcap D_n$ **then** c-lcs(D, C)
else $\top$

Algorithm 2 recursively computes $lcs(C, D)$ given $\hat{C}$ and $\hat{D}$ as arguments. Applying Algorithm 2 to $\hat{X}_1$ and $\hat{Y}_1$ yields $(\geq 2\, R\, A)$ and to $\hat{X}_3$ and $\hat{Y}_3$ yields $(\geq 5\, R)$ as desired.

Theorem 1 *If C and D are either $\mathcal{ALENR}$ or $\mathcal{ALQ}$ concepts, $\hat{C} \in SSNF(C)$, and $\hat{D} \in SSNF(D)$, then* c-lcs$(\hat{C}, \hat{D}) \in lcs(C, D)$.

Theorem 1 is a consequence of Theorem 3 in [3] which states that the LCS of concepts C and D can be computed according to Algorithm 2 if, for each language constructor, a least upper bound operation on the arguments of the constructor can be provided and subsumption between C and D can be decided by a structural subsumption algorithm. In the preceeding sections, we dealt with computing a structural subsumption normal form and the least upper bound operations are implicitly given by Algorithm 2.

6 Conclusion

We have presented an LCS operator for the expressive DLs $\mathcal{ALQ}$ and $\mathcal{ALENR}$. As shown earlier, the LCS can be computed by a simple algorithm if the concepts are first transformed into a normal form with which subsumption between the concepts can be decided by a structural subsumption algorithm. The special challenge one faces when computing the normal form is that conjunctions of qualified number restrictions (in $\mathcal{ALQ}$) and of $\exists$-quantifications (in $\mathcal{ALENR}$) imply $\geq$-restrictions which have an effect on the structural subsumption algorithm. We extended the LCS computation algorithm for the DLs $\mathcal{ALN}$ and $\mathcal{ALE}$ to the new language constructs included in $\mathcal{ALQ}$ and $\mathcal{ALENR}$. It would be an interesting future research topic to extend the notion of concept commonalities to DLs including a disjunction operator. The LCS operator does not seem to be useful for this because the LCS of two concepts would just be their disjunction which we found to not express meaningful commonalities in our similarity-based information retrieval application.

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, *Proceedings 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. In *Proc. of the IJCAI'99*, 1999. To appear.

[3] W. W. Cohen, A. Borgida, and H. Hirsh. Computing Least Common Subsumers in Description Logics. In *Proc. of the AAAI'92*, pages 754–760. AAAI Press/The MIT Press, 1992.

[4] T. Mantay. Computing Least Common Subsumers in Expressive Description Logics. TR FBI-HH-M-286/99, Dep. of CS, University of Hamburg, 1999.

[5] R. Möller, V. Haarslev, and B. Neumann. Semantics-based Information Retrieval. In *Int. Conf. on Inf. Tech. and Knowl. Sys.*, Vienna, Budapest, 1998.