Role-depth Bounded Least Common Subsumers for $\mathcal{EL}^+$ and $\mathcal{ELI}$

Andreas Ecke and Anni-Yasmin Turhan[⋆]

TU Dresden, Institute for Theoretical Computer Science

Abstract. For $\mathcal{EL}$ the least common subsumer (lcs) need not exist, if computed w.r.t. general TBoxes. In case the role-depth of the lcs concept description is bounded, an approximate solution can be obtained. In this paper we extend the completion-based method for computing such approximate solutions to $\mathcal{ELI}$ and $\mathcal{EL}^+$. For $\mathcal{ELI}$ the extension needs to be able to treat complex node labels. For $\mathcal{EL}^+$ a naive method generates highly redundant concept descriptions for which we devise a heuristic that produces smaller, but equivalent concept descriptions. We demonstrate the usefulness of this heuristic by an evaluation.

1 Introduction

The reasoning service least common subsumer (lcs) computes a concept description from set of concept descriptions expressed in a DL $\mathcal{L}$. The resulting concept description subsumes all of the input concept descriptions and is the least w.r.t. subsumption expressible in $\mathcal{L}$ to do so. This reasoning service has turned out to be useful for the augmentation of TBoxes [1] and as a subtask when computing the (dis)similarity of concept descriptions [2, 3] or other non-standard inferences.

In particular several bio-medical TBoxes are written in extensions of $\mathcal{EL}$ that allow to model roles in a more detailed way, such as SNOMED [4] which allows to use role inclusions and is written in $\mathcal{ELH}$ or the Gene Ontology [5] and the FMA ontology [6] which are both written in $\mathcal{EL}^+$, which is a DL that extends $\mathcal{ELH}$ by right identities for roles. For these extensions of $\mathcal{EL}$ standard DL reasoning can still be done in polynomial time [7]. However, the GALEN ontology uses the DL $\mathcal{ELHI}f_{\mathcal{R}^+}$—a DL with inverse roles, which are known to make subsumption w.r.t. general TBoxes ExpTime-complete [7] due to the use of inverse roles. These TBoxes are known to be very large and are mostly build by hand.

If computed w.r.t. general or just cyclic $\mathcal{EL}$-TBoxes, the lcs need not exist [8], since resulting cyclic concept descriptions cannot be expressed in $\mathcal{EL}$. In [9] an extension of $\mathcal{EL}$ by fixed-points has been investigated that can capture such concept descriptions. Since we want to obtain a concept description for the lcs that is expressed in that DL in which the TBox is written, we follow the idea from [10] and compute as an approximative solution the *role-depth bounded lcs*: k-lcs which has a maximal nesting of quantifiers limited to k.

⋆ Partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing".

Table 1. Constructors and axioms for $\mathcal{EL}$ and some of its extensions

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
inverse role	r^{-}	$\{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}\}$
general concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion axiom	$r_1 \circ \ldots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \ldots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

The approach to compute the k-lcs is to employ the completion method that is used to classify the TBox. This method builds a graph structure, which is saturated by completion rules [11, 7]. In case of $\mathcal{EL}$ the k-lcs can be more or less directly be read off from the saturated completion graph. In this paper we devise computation algorithms for the k-LCS for the DLs $\mathcal{EL}^+$ and in $\mathcal{ELI}$. It turns out that for $\mathcal{EL}^+$ the computation algorithm is the same as for $\mathcal{EL}$ [10]. While the polynomial time completion algorithm for $\mathcal{EL}^+$ works on graph structures with static node sets and have simple labellings, the algorithm for $\mathcal{ELI}$ requires dynamic nodes sets and uses complex labels. In [12] such a completion algorithm for $\mathcal{ELI}$ has been devised, which we employ for the computation of the k-lcs in $\mathcal{ELI}$.

For both methods we show that the obtained concept is a common subsumer and that it is minimal w.r.t. subsumption for the given role-depth bound k. Thus, the obtained concept description is the *exact* lcs, if the exact lcs exists for a role-depth n and the k-lcs is computed for a maximal role-depth of $k \geq n$.

The concept descriptions obtained in this way turn out to be highly redundant. In order to obtain concise and readable concept descriptions, we devise a heuristic to obtain smaller, equivalent concept descriptions.

This paper is organised as follows: next, we introduce the basic notions. In Section 3 we recall the completion algorithm for $\mathcal{EL}^+$ and devise the computation algorithms for the k-lcs in $\mathcal{EL}^+$. The computation algorithm for $\mathcal{ELI}$ is presented in Section 4. In Section 5 we present the simplification heuristic to obtain smaller $\mathcal{EL}^+$-concept descriptions. We end with conclusions and remarks on future work.

2 Preliminaries

We assume that the reader is familiar with the basic notions of DLs, for an introduction see [13]. We introduce the DLs used in this paper formally. *Concept descriptions* are inductively defined from a set of *concepts names* N_C and a set of *role names* N_R by applying the constructors from the upper half of Table 1. In particular, $\mathcal{EL}$-concept descriptions only allow for conjunctions, existential restrictions, and the top concept $\top$. $\mathcal{EL}^+$ additionally allows for complex *role inclusion axioms* (RIAs). These role inclusions can express role hierarchies ($s \sqsubseteq r$) and transitive roles ($r \circ r \sqsubseteq r$). The semantics are displayed in the lower half of Table 1. $\mathcal{ELI}$-concept description extend $\mathcal{EL}$-concept descriptions by the use of *inverse roles.*

The concept constructors and axioms are interpreted in the standard way. We denote by $N_{C,\mathcal{T}}$ and $N_{R,\mathcal{T}}$ the sets of concept names and role names that occur in a TBox $\mathcal{T}$. For a concept description C we denote by $rd(C)$ its role-depth, i.e., its maximal nesting of quantifiers. We define the central reasoning services of this paper.

Definition 1 ((Role-depth bounded) least common subsumer). *Let $\mathcal{L}$ be a DL, $\mathcal{T}$ be a $\mathcal{L}$-TBox and $C_1, \dots, C_n$ be $\mathcal{L}$-concept descriptions. Then the $\mathcal{L}$-concept description D is the* least common subsumer *of $C_1, \dots, C_n$ w.r.t. $\mathcal{T}$ iff (1) $C_i \sqsubseteq_{\mathcal{T}} D$ for all $i \in \{1, \dots, n\}$, and (2) for all $\mathcal{L}$-concept descriptions E: $C_i \sqsubseteq_{\mathcal{T}} E$ for all $i \in \{1, \dots, n\}$ implies $D \sqsubseteq_{\mathcal{T}} E$.*

Let $k \in \mathbb{N}$. Then the $\mathcal{L}$-concept description D is the role-depth bounded least common subsumer *of $C_1, \dots, C_n$ w.r.t. $\mathcal{T}$ and the role-depth k (k–$lcs(C_1, \dots, C_n)$) iff (1) $rd(D) \leq k$, (2) $C_i \sqsubseteq_{\mathcal{T}} D$ for all $i \in \{1, \dots, n\}$, and (3) for all $\mathcal{L}$-concept descriptions E with $rd(E) \leq k$: $C_i \sqsubseteq_{\mathcal{T}} E$ $\forall i \in \{1, \dots, n\}$ implies $D \sqsubseteq_{\mathcal{T}} E$.*

For the DLs considered in this paper the (k-)lcs is unique up to equivalence, thus we speak of the (k-)lcs.

3 Computing the k-lcs in $\mathcal{EL}^+$

The algorithms to compute the role-depth bounded lcs rely on completion graphs produced by completion-based subsumption algorithms. Completion algorithms work on normalized TBoxes and for which they build a completion graph and exhaustively apply completion rules. After this step, the completion graph contains all subsumption relations from the TBox explicitly.

3.1 Completion Algorithm for $\mathcal{EL}^+$

An $\mathcal{EL}^+$-TBox $\mathcal{T}$ is in normal form, if all concept inclusions in $\mathcal{T}$ are of the form $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, or $\exists r.A \sqsubseteq B$ with $A, A_1, A_2, B \in N_C$ and $r \in N_R$; and all role inclusions are of the form $s \sqsubseteq r$ or $s \circ t \sqsubseteq r$ with $\{r, s, t\} \subseteq N_R$. All $\mathcal{EL}^+$-TBoxes can be normalized by applying a set of normalization rules [11].

The completion graph for a normalized TBox $\mathcal{T}'$ used by the completion algorithm is of the form (V, E, S), where $V = N_{C,\mathcal{T}'} \cup \{\top\}$ is the set of nodes, $E \subseteq V \times N_{R,\mathcal{T}} \times V$ is the set of role name labeled edges and $S : V \to 2^{N_{C,\mathcal{T}'} \cup \{\top\}}$ is the node-labeling. The completion algorithms starts with an initial graph (V, E, S) with $E = \emptyset$ and $S(A) = \{A, \top\}$ for each $A \in N_{C,\mathcal{T}'} \cup \{\top\}$ and exhaustively applies a set of completion rules from [11] until no more rule applies.

Once the rule-applications finished, all subsumption relations can be directly be read off the completion graph. This completion algorithm is sound and complete as shown in [11]. Specifically, given a normalized $\mathcal{EL}^+$-TBox $\mathcal{T}$ and its completion graph (V, E, S) after all completions rules were applied exhaustively, we have for each $A, B \in V$ and $r \in E$:

```
Procedure k-lcs ($C, D, \mathcal{T}, k$)
Input: $C, D$: $\mathcal{EL}^+$-concept descriptions; $\mathcal{T}$: $\mathcal{EL}^+$-TBox; $k$: natural number
Output: k-lcs($C, D$): role-depth bounded $\mathcal{EL}^+$-lcs of $C, D$ w.r.t. $\mathcal{T}$ and $k$

1: $\mathcal{T}' :=$ normalize($\mathcal{T} \cup \{A \equiv C, B \equiv D\}$)
2: $(V, E, S) :=$ apply-completion-rules($\mathcal{T}'$)
3: $L :=$ k-lcs-r($A, B, (V, E, S), k$)
4: return remove-normalization-names($L$)

Procedure k-lcs-r($A, B, (V, E, S), k$)
Input: $A, B$: concept names; $(V, E, S)$: completion graph; $k$: natural number
Output: k-lcs($A, B$): role-depth bounded $\mathcal{EL}^+$-lcs of $A, B$ w.r.t. $\mathcal{T}$ and $k$

1: common-names $:= S(A) \cap S(B)$
2: if $k = 0$ then
3:     return $\bigsqcap_{P \in \text{common-names}} P$
4: else
5:     return $\bigsqcap_{P \in \text{common-names}} P \sqcap$
           $\bigsqcap_{r \in N_R} \Big( \bigsqcap_{(A,r,C) \in E, (B,r,D) \in E} \exists r.\text{k-lcs-r}(C, D, (V, E, S), k-1) \Big)$
```

Fig. 1. Computation Algorithm for role-depth bounded $\mathcal{EL}^+$-lcs.

Soundness If $B \in S(A)$, then $A \sqsubseteq_{\mathcal{T}} B$; and
if $(A, r, B) \in E$, then $A \sqsubseteq_{\mathcal{T}} \exists r.B$.

Completeness If $A \sqsubseteq_{\mathcal{T}} B$, then $B \in S(A)$; and
if $A \sqsubseteq_{\mathcal{T}} \exists r.B$, then there are $C, D \in V$ with $C \in S(A)$, $B \in S(D)$ and $(C, r, D) \in E$.

3.2 Computation Algorithm of the k-lcs in $\mathcal{EL}^+$

The resulting completion graph can be used to compute the role-depth bounded lcs. All RIAs from the $\mathcal{EL}^+$-TBox are explicitly captured in the completion graph in the following sense: for each edge in the completion graph labeled with some role r, the completion algorithm also creates edges for all its super-roles. This means that for computing the k-lcs for an $\mathcal{EL}^+$-TBox the same algorithm can be used as for $\mathcal{EL}$, which was introduced in [10] and is shown in Algorithm 3.2 for the binary lcs. The idea is to introduce new concept names for the concept descriptions of interest and to apply the completion algorithm. Then, starting from the newly introduced names A and B, traverse the completion graph simultaneously. More precisely, for the tree unravelings of depth k for A and B the cross product is computed. In a post-processing step those concept names have to be removed from the concept that were introduced during normalization. Obviously, this method creates heavily redundant concept descriptions, due to the multiple edge labellings due to RIAs.

```
Procedure simplify($C, (V, E, S), \mathcal{T}$)
Input: $C$: $\mathcal{EL}^+$-concept description; $(V, E, S)$: completion graph; $\mathcal{T}$: $\mathcal{EL}^+$-TBox
Output: simplify($C$): simplified concept description
 1: Let $C \equiv A_1 \sqcap \ldots \sqcap A_n \sqcap \exists r_1.D_1 \sqcap \ldots \sqcap \exists r_m.D_m$ with $A_i \in N_C$ for $1 \leq i \leq n$.
 2: $Conj := \{A_i \mid 1 \leq i \leq n\} \cup \{\exists r_j.D_j \mid 1 \leq j \leq m\}$
 3: for all $X \in Conj$ do
 4:    for all $Y \in Conj$ do
 5:       if $X \neq Y \wedge$ subsumes-H($X, Y, (V, E, S), \mathcal{T}$) then
 6:          $Conj := Conj \setminus \{X\}$
 7:          break
 8: for all $X \in Conj$ do
 9:    if $X = \exists r_j.D_j$ then
10:       $Conj := (Conj \setminus \{\exists r_j.D_j\}) \cup \{\exists r_j.\text{simplify}(D_j, (V, E, S), \mathcal{T})\}$
11: return $\bigsqcap_{X \in Conj} X$
```

Fig. 2. Simplification algorithm for $\mathcal{EL}^+$-concept descriptions

3.3 Simplifying $\mathcal{EL}^+$-Concept Descriptions

The highly redundant $\mathcal{ELH}$-concept descriptions obtained from the *k*-lcs algorithm, need to be simplified, in order to make the resulting concept description readable. The general idea for the simplification is to remove those subtrees from the syntax tree which are subsumers of any of their sibling subtrees. For a conjunction of concept names, this results in the least ones (w.r.t. $\sqsubseteq_{\mathcal{T}}$).

Algorithm 2 computes the simplification of an $\mathcal{EL}^+$-concept description. Note, that the algorithm needs to be applied after the normalization names were removed, otherwise it might remove names from the original TBox that subsume normalization names, which get removed later during denormalization.

For the soundness of the simplification procedure simplify, it is only necessary to ensure that the procedure 'subsumes-H' is sound. However, for our purpose this procedure does not have to be complete. This might result in simplifications that are correct *k*-lcs, but that are still redundant. This heuristic is given in [14]. The idea is to make simple structural comparison depending on the concept constructor of the concepts in question.

Obviously, it would be desirable to avoid the generation of highly redundant concept descriptions, instead of reducing them in a post-processing step. Due to interactions with denormalization, such optimizations need to be conservative. Such optimizations have been investigated in [14], which avoid unnecessary branching and role-depth of the generated concept description. Interestingly, these optimizations do not only speed-up the execution of Algorithm 3.2, but also of the subsequent simplification, see [14].

Evaluation. The *k*-lcs algorithm and the simplification algorithm are implemented in our system GEL[1], which is implemented on top of the jCEL rea-

[1] GEL is freely available from http://sourceforge.net/p/gen-el.

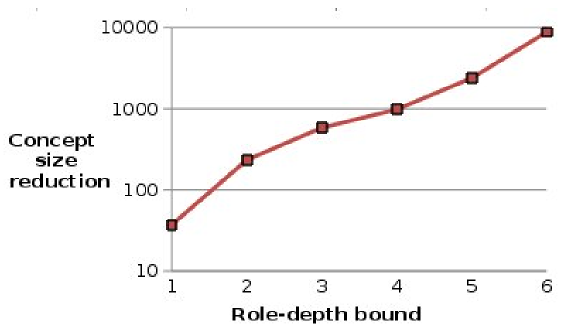

Fig. 3. Average gain in concept size for simplified k-lcs computed w.r.t. Not-Galen

soner[2] [15]. We have tested the effectiveness of the simplification procedure on the NotGalen ontology, which is a version of the GALEN ontology pruned to $\mathcal{EL}^+$. Some input concept pairs resulted in run-times over a minute for $k = 6$, which were mostly dominated by the run-time of the k-lcs-r-procedure. Simplification of larger concepts was faster by a factor of 10 or more. Figure 3 shows the average gain in concept size by simplification on various input pairs for different values of k. For $k = 6$ concepts with a size of several thousands were reduced to a concept size of 30 to 40, which are large, but still readable concept descriptions. In an extreme case a concept of size of over 10^6 was reduced to a size of 140. For more empirical results and details on the implementation of GEL see [14].

4 Computing the k-lcs in $\mathcal{ELI}$

To handle inverse roles correctly, the completion algorithm needs to be adapted in several ways. The normal form for TBoxes is the same as before.

4.1 Completion Algorithm for $\mathcal{ELI}$

The $\mathcal{EL}$-completion algorithm has been extended to $\mathcal{ELI}$ in [12]. One adaptation is that the node set V is not fixed. Consider the example TBox $\mathcal{T} = \{\exists r^-.A \sqsubseteq C, A \sqsubseteq \exists r.B\}$. In this TBox, A has an r-successor subsumed by B and each r-predecessor A implies C. However, that does not mean that C is also a subsumer of B – only those elements in $B^\mathcal{I}$, that are r-successors of elements in $A^\mathcal{I}$ are also in $C^\mathcal{I}$. Thus, $C \notin S(B)$. On the other hand we know that $A \sqsubseteq \exists r.C$. To solve this problem, we need to have a dynamic node set V, add a new node u to V for $u = B \sqcap \exists r^-.A$ and then add C to the completion set $S(u)$.

The node set V is defined as $V \subseteq N_{C,\mathcal{T}} \times 2^{\{\exists r.X \mid r \text{ is a role}, X \in N_{C,\mathcal{T}}\}}$. A node A with $A \in N_{C,\mathcal{T}}$ from the node set for $\mathcal{EL}^+$ would then correspond to the node $(A, \emptyset)$ from the node set for $\mathcal{ELI}$. We will formalize the meaning of nodes in the node set V by defining the concept descriptions that these nodes correspond to:

[2] jCEL is freely available from http://jcel.sourceforge.net.

Definition 2 (Concept descriptions for nodes). *Let $\mathcal{T}$ be a normalized $\mathcal{ELI}$-TBox and (V, E, S) its completion graph. Then we define for each node $u = (A, \phi) \in V$: vconcept$(u) = A \sqcap \bigsqcap_{\exists r.X \in \phi} \exists r.X$*

The graph (V, E, S) for the completion algorithm for $\mathcal{ELI}$ starts with $V = \{(A, \emptyset) \mid A \in N_{C,\mathcal{T}}\}$, $E = \emptyset$ and $S((A, \emptyset)) = \{A, \top\}$ for all $A \in N_{C,\mathcal{T}}$. The completions rules for $\mathcal{ELI}$ are the following:

CI1 If $A_1 \in S(v)$ and $A_1 \sqsubseteq B \in \mathcal{T}$ and $B \notin S(v)$,
then $S(v) := S(v) \cup \{B\}$
CI2 If $A_1, A_2 \in S(v)$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $B \notin S(v)$,
then $S(v) := S(v) \cup \{B\}$
CI3 If $A_1 \in S(u)$, $v = (B, \emptyset)$ and $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $(u, r, v) \notin E$,
then $E := E \cup \{(u, r, v)\}$
CI4 If $(u, r, v) \in E$, $B_1 \in S(v)$ and $\exists r.B_1 \sqsubseteq C \in \mathcal{T}$ and $C \notin S(u)$,
then $S(u) := S(u) \cup \{C\}$
CI5 If $(u, r, v) \in E$, $v = (B, \psi)$, $A_1 \in S(u)$, $\exists r^-.A_1 \sqsubseteq B_1 \in \mathcal{T}$ and $B_1 \notin S(v)$,
then
$v' := (B, \psi \cup \{\exists r^-.A_1\})$
if $v' \notin V$ then $V := V \cup \{v'\}$, $E := E \cup \{(u, r, v')\}$, $S(v') := S(v) \cup \{B_1\}$
else $E := E \cup \{(u, r, v')\}$, $S(v') := S(v') \cup \{B_1\}$

The completion algorithm for $\mathcal{ELI}$ defined this way is again sound. For completeness one needs to consider only those edges that do not point to nodes, which have an 'extended copy' generated by rule CI5, i.e., edges (u, r, v) for which there is no $\exists r^-.A \sqsubseteq B \in \mathcal{T}$ with $A \in S(u)$ and $B \notin S(v)$. We call those edges *bad edges* and collect them in the bad edge set E_{bad}. However, since for each edge (u, r, v) in E_{bad} there is $(u, r, v') \in E \setminus e_{bad}$ with *vconcept*$(v') \sqsubseteq_{\mathcal{T}}$ *vconcept*(v), completeness for *good* edges is sufficient to show that the concept description obtained by Algorithm 1 is a common subsumer [16].

4.2 Computation of the k-lcs in $\mathcal{ELI}$

Since $\mathcal{ELI}$ allows for inverse roles, we may also traverse edges backwards (i.e., use the inverse role of the role that the edge is labeled with in the k-lcs concept description). However, we can only traverse those edges backwards, that we just came from–as you can see in the example for $\mathcal{T} = \{A \sqsubseteq \exists r.\top, B \sqsubseteq \exists r.C, C \sqsubseteq \exists r^-.A\}$, which results in the following completion graph:

$$\begin{tikzcd}
A \arrow[d, "r"'] & B \arrow[d, "r"] \\
\top & C \arrow[ul, "r^-"']
\end{tikzcd}$$

Now, traversing this completion graph to compute the lcs of A and B without going backwards, we would get the result $\top \sqcap \exists r.\top$ and then get stuck in the $\top$ node. However, the lcs of A and B is $\exists r.\exists r^-.A$, therefore the algorithm must to go backwards from $\top$ to A using the edge $(A, r, \top)$ as $(\top, r^-, A)$, which yields the correct lcs. To see that the algorithm may not go backwards along arbitrary edges consider to go from A to C using the edge (C, r^-, A) as (A, r, C). This would clearly be wrong, since we don't have $A \sqsubseteq_\mathcal{T} \exists r.C$. Thus the algorithm may only traverse backwards on those edges that led to the current node.

Therefore, the recursive algorithm needs to know not only the current nodes, but also the whole path from the start to the current node. This path is given in the form $[u_0, r_1, u_1, r_2, \ldots, r_n, u_n]$ where u_0 is the starting node, u_n the current node, and $(u_{i-1}, r_i, u_i) \in E$ are edges of the completion graph that have been traversed. For each path $[u_0, r_1, u_1, r_2, \ldots, r_n, u_n]$ we will define the concept description they correspond to.

Definition 3 (Concept descriptions for paths). *Let $\mathcal{T}$ be a normalized $\mathcal{ELI}$-TBox and (V, E, S) its completion graph. Then we define for each path $l = [u_0, r_1, u_1, r_2, \ldots, r_n, u_n]$*

$$\mathit{lconcept}(l) = \mathit{vconcept}(u_n) \sqcap \exists r_n^-.(\mathit{vconcept}(u_{n-1}) \sqcap \exists r_{n-1}^-.(\ldots \sqcap \exists r_1^-.\mathit{vconcept}(u_0)\ldots))$$

Algorithm 1 depicted below computes the role-depth bounded lcs for two $\mathcal{ELI}$-concept descriptions C and D w.r.t. a general $\mathcal{ELI}$-TBox. This algorithm differs from the Algorithm 3.2 for $\mathcal{EL}^+$ mainly only in the following aspects:

– Algorithm 1 uses the whole path to the current node instead of the node itself.
– While in Algorithm 3.2 the nodes to visit from the current node are computed implicitly, Algorithm 1 stores all successors of the paths p_1 and p_2 explicitly in the sets S_1 and S_2.
– Both algorithms traverse all edges (u, r, v) from the current node u, but Algorithm 1 additionally traverses the last edge backwards, if it is the inverse of r.

We give a proof sketch that Algorithm 1 indeed computes the k-lcs. Condition (1) from the Definition of the role-depth bounded lcs is obviously given.

Common Subsumer. The fact that Algorithm 1 yields a common subsumer follows directly from the following lemma:

Lemma 1. *Let $L = k\text{-lcs-r}(p_1, p_2, (V, E, S), k)$ for the two given paths $p_1 = [u_0, r_1, u_1, r_2, \ldots, r_n, u_n]$ and $p_2 = [v_0, s_1, v_1, s_2, \ldots, s_m, v_m]$. Then $\mathit{lconcept}(p_1) \sqsubseteq_\mathcal{T} L$ and $\mathit{lconcept}(p_2) \sqsubseteq_\mathcal{T} L$.*

Proof. This lemma can be proven by induction on the role-depth k of L. For $k = 0$, $L = A_1 \sqcap A_2 \sqcap \ldots \sqcap A_l$ must be a conjunction of concept names $A_i \in S(u_n) \cap S(v_m), 0 \leq i \leq l$. Then soundness of the completion algorithm yields that for each A_i, we have $\mathit{lconcept}(p_1) \sqsubseteq_\mathcal{T} \mathit{vconcept}(u_n) \sqsubseteq_\mathcal{T} A_i$ and similarly $\mathit{lconcept}(p_2) \sqsubseteq_\mathcal{T} A_i$; therefore $\mathit{lconcept}(p_1) \sqsubseteq_\mathcal{T} L$ and $\mathit{lconcept}(p_2) \sqsubseteq_\mathcal{T} L$.

For $k \geq 1$, L is a conjunction of concept names and existential restrictions. For concept names, the same argument as above holds. All existential restrictions are of the form $\exists r.k\text{-lcs-r}(l_1, l_2, (V, E, S), k-1)$ where l_1 is either

Algorithm 1 Computation of a role-depth bounded $\mathcal{ELI}$-lcs.

```
Procedure $k$-lcs$(C, D, \mathcal{T}, k)$
Input: $C, D$: $\mathcal{ELI}$-concept descriptions; $\mathcal{T}$: $\mathcal{ELI}$-TBox; $k$: natural number
Output: $k$-lcs$(C, D)$: role-depth bounded $\mathcal{ELI}$-lcs of $C$ and $D$ w.r.t. $\mathcal{T}$ and $k$
 1: $\mathcal{T}' :=$ normalize$(\mathcal{T} \cup \{A \equiv C, B \equiv D\})$
 2: $(V, E, S) :=$ apply-completion-rules$(\mathcal{T}')$
 3: $L := k$-lcs-r$([(A, \emptyset)], [(B, \emptyset)], (V, E, S), k)$
 4: return remove-normalization-names$(L)$

Procedure $k$-lcs-r$(p_1, p_2, (V, E, S), k)$
Input: $p_1 = [(A_0, \emptyset), r_1, \ldots, r_n, (A_n, \phi_n)]$ and $p_2 = [(B_0, \emptyset), s_1, \ldots, s_n, (B_m, \psi_m)]$: two paths in the completion graph; $(V, E, S)$: completion graph; $k$: natural number
Output: role-depth bounded $\mathcal{ELI}$-lcs of $lconcept(p_1)$ and $lconcept(p_2)$ w.r.t. $\mathcal{T}$ and $k$
 1: result-concept $:= \bigsqcap_{C \in S((A_n,\phi_n)) \cap S((B_m,\psi_m))} C$
 2: if $k > 0$ then
 3:   for all $r \in N_R$ do
 4:     $S_1 := \{[(A_0, \emptyset), r_1, \ldots, r_n, (A_n, \phi_n), r, (A, \phi)] \mid ((A_n, \phi_n), r, (A, \phi)) \in E\}$
 5:     if $n > 0 \wedge r = r_n^-$ then
 6:       $S_1 := S_1 \cup \{[(A_0, \emptyset), r_1, (A_1, \phi_1), r_2, \ldots, (A_{n-2}, \phi_{n-2}), r_{n-1}, (A_{n-1}, \phi_{n-1})]\}$
 7:     $S_2 := \{[(B_0, \emptyset), s_1, \ldots, s_n, (B_m, \psi_m), r, (B, \psi)] \mid ((B_m, \psi_m), r, (B, \psi)) \in E\}$
 8:     if $n > 0 \wedge r = s_m^-$ then
 9:       $S_2 := S_2 \cup \{[(B_0, \emptyset), s_1, (B_1, \psi_1), s_2, \ldots, (B_{m-2}, \psi_{m-2}), s_{m-1}, (B_{m-1}, \psi_{m-1})]\}$
10:     result-concept := result-concept $\sqcap \bigsqcap_{l_1 \in S_1, l_2 \in S_2} \exists r.k$-lcs-r$(l_1, l_2, (V, E, S), k - 1)$
11: return result-concept
```

p_1 extended by one more edge $(u_n, r, u) \in E$ or shorted by the last edge if $r_n = r^-$. In the first case soundness of completion for $(u_n, r, u) \in E$ yields $vconcept(u_n) \sqsubseteq_{\mathcal{T}} \exists r.vconcept(u)$ and thus $lconcept(p_1) \sqsubseteq_{\mathcal{T}} \exists r.(vconcept(u) \sqcap \exists r^-.lconcept(p_1)) = \exists r.lconcept(l_1)$. In the second case we have $lconcept(p_1) = vconcept(u_n) \sqcap \exists r_n^-.lconcept(l_1) \sqsubseteq_{\mathcal{T}} \exists r.lconcept(l_1)$. Then the induction hypothesis yields that $lconcept(p_1) \sqsubseteq_{\mathcal{T}} \exists r.k\text{-lcs-r}(l_1, l_2, (V, E, S), k-1)$, therefore $lconcept(p_1) \sqsubseteq_{\mathcal{T}} L$ holds and by the same argument $lconcept(p_2) \sqsubseteq_{\mathcal{T}} L$ holds.

Minimality. To show that Algorithm 1 yields the *least* common subsumer w.r.t. the role-depth bound k, we show the following lemma.

Lemma 2. *Let p_1 and p_2 be two paths in the completion graph (V, E, S) with $p_1 = [u_0, r_1, \ldots, r_n, u_n]$ and $p_2 = [v_0, s_1, \ldots, s_m, v_m]$, such that $(u_{i-1}, r_i, u_i) \in E \setminus E_{bad}$ for all $1 \leq i \leq n$ and $(v_{j-1}, s_j, v_j) \in E \setminus E_{bad}$ for all $1 \leq j \leq m$, $u_0 = (A, \emptyset)$ and $v_0 = (B, \emptyset)$. Let $k \in \mathbb{N}$ and F an $\mathcal{ELI}$-concept description with $rd(F) \leq k$. If $lconcept(p_1) \sqsubseteq_{\mathcal{T}} F$ and $lconcept(p_2) \sqsubseteq_{\mathcal{T}} F$ then $L = k\text{-lcs-r}(p_1, p_2, (V, E, S), k) \sqsubseteq_{\mathcal{T}} F$.*

Proof. We prove this claim by induction on the role-depth bound k. For $k = 0$, $F = A_1 \sqcap \ldots \sqcap A_n$ must be a conjunction of concept names. Since $lconcept(p_1) \sqsubseteq_{\mathcal{T}}$

F and $\mathit{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} F$, we have $\mathit{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} A_i$ and $\mathit{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} A_i$ for all $1 \leq i \leq n$. Since p_1 and p_2 only traverse edges over $E \backslash E_{bad}$, all possible rule applications of CI5 during that path were applied, and we have $\mathit{vconcept}(u_n) \sqsubseteq_{\mathcal{T}} A_i$ and $\mathit{vconcept}(v_m) \sqsubseteq_{\mathcal{T}} A_i$. Then completeness of the completion algorithm yields $A_i \in S(u_n)$ and $A_i \in S(v_m)$ for all $1 \leq i \leq n$. Thus, $L \sqsubseteq_{\mathcal{T}} F$.

For $k \geq 1$, F is a conjunction of concept names and existential restrictions. The concept names in F must appear in L by the same argument as in the base case. For each existential restriction $\exists r.F'$ of F, we can again use the fact that p_1 and p_2 only traverse edges over $E \setminus E_{bad}$ to derive that there must be nodes u and v with $\mathit{vconcept}(u) \sqsubseteq_{\mathcal{T}} F', \mathit{vconcept}(v) \sqsubseteq_{\mathcal{T}} F'$, such that $\mathit{vconcept}(u_n) \sqsubseteq_{\mathcal{T}} \exists r.\mathit{vconcept}(u)$ and $\mathit{vconcept}(v_m) \sqsubseteq_{\mathcal{T}} \exists r.\mathit{vconcept}(v)$. Then completeness of completion yields that there are u' and v' with $(u_n, r, u') \in E \setminus E_{bad}$ or $u' = u_{n-1}, r = r_n^-$ and similarly $(v_m, r, v') \in E \setminus E_{bad}$ or $v' = v_{m-1}, r = s_m^-$, such that $\mathit{vconcept}(u') \sqsubseteq_{\mathcal{T}} \mathit{vconcept}(u)$ and $\mathit{vconcept}(v') \sqsubseteq_{\mathcal{T}} \mathit{vconcept}(v)$. Therefore, there are new paths $l_1 \in S_1$ and $l_2 \in S_2$, such that $\mathit{lconcept}(l_1) \sqsubseteq_{\mathcal{T}} F'$ and $\mathit{lconcept}(l_2) \sqsubseteq_{\mathcal{T}} F'$ which still only traverse edges in $E \setminus E_{bad}$, so the induction hypothesis yields k-lcs-r$(l_1, l_2, (V, E, S), k-1) \sqsubseteq_{\mathcal{T}} F'$, and thus $L = k$-lcs-r$(p_1, p_2, (V, E, S), k) \sqsubseteq_{\mathcal{T}} F$.

This shows that the Algorithm 1 computes the role-depth bounded least common subsumer for $\mathcal{ELI}$. In contrast to subsumption, the computation of k-lcs does not increase complexity-wise when going from $\mathcal{EL}$ to $\mathcal{ELI}$– it remains exponential in the size of k.

5 Conclusions and Future Work

In this paper we have extended the computation algorithm for the k-lcs in $\mathcal{EL}$ w.r.t. general TBoxes to two members of the $\mathcal{EL}$-family and showed that the proposed methods indeed compute the k-lcs. In cases where the exact lcs exists, our algorithms compute the exact lcs for a big enough k.

For $\mathcal{ELI}$ the extension of the $\mathcal{EL}$ algorithm for computing the k-lcs required traversal of the completion graph w.r.t. paths and the correct handling of complex node labels.

In case of $\mathcal{EL}^+$, the extension of the computation method for $\mathcal{EL}$ turned out to be trivial, here our contribution rather lies in the simplification procedure devised. This procedure turned out to be extremely helpful, when reducing the concept size. For the NotGalen ontology the the result concepts were reduced by several orders of magnitude. It would be desirable to obtain the simplified $\mathcal{EL}^+$-concept descriptions directly, instead of in the generate and then reduce kind of fashion employed so far. Besides this, we want to extend our results on $\mathcal{EL}^+$ and $\mathcal{ELI}$ to the computation of most specific concepts by completion.

References

1. Turhan, A.-Y.: On the Computation of Common Subsumers in Description Logics. PhD thesis, TU Dresden, Institute for Theoretical Computer Science (2007)

2. d'Amato, C., Fanizzi, N., Esposito, F.: A dissimilarity measure for $\mathcal{ALC}$ concept descriptions. In: Proceedings of the ACM symposium on Applied computing. SAC '06 (2006) 1695 – 1699
3. Janowicz, K.: Computing Semantic Similarity Among Geographic Feature Types Represented in Expressive Description Logics. PhD thesis, Institute for Geoinformatics, University of Münster, Germany (2008)
4. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. Journal of the American Medical Informatics Assoc. (2000) Fall Symposium Special Issue.
5. Consortium, T.G.O.: Gene Ontology: Tool for the unification of biology. Nature Genetics **25** (2000) 25–29
6. Rosse, C., Mejino, J.L.V.: A reference ontology for biomedical informatics: the foundational model of anatomy. Journal of Biomedical Informatics **36**(6) (2003) 478–500
7. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In Clark, K., Patel-Schneider, P.F., eds.: In Proc. of the OWLED Workshop. (2008)
8. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Gottlob, G., Walsh, T., eds.: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03), Morgan Kaufmann (2003) 325–330
9. C. Lutz, R. Piro, and F. Wolter. Enriching $\mathcal{EL}$-concepts with greatest fixpoints. In Proc. of the 19th European Conf. on Artificial Intelligence (ECAI-10). IOS Press, (2010)
10. Peñaloza, R., Turhan, A.-Y.: A practical approach for computing generalization inferences in $\mathcal{EL}$. In Grobelnik, M., Simperl, E., eds.: Proc. of the 8th European Semantic Web Conf. (ESWC'11). Lecture Notes in Computer Science, Springer (2011)
11. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05, Edinburgh, UK, Morgan-Kaufmann Publishers (2005)
12. Vu, Q.H.: Subsumption in the description logic $\mathcal{ELHI}f_{R+}$ w.r.t. general tboxes. Master's thesis, Technische Universität Dresden (2008)
13. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
14. A. Ecke and A.-Y. Turhan. Optimizations for the role-depth bounded least common subsumer in $\mathcal{EL}+$. In M. Horridge and P. Klinov, eds.: In Proc. of the OWLED Workshop, (2012) To appear.
15. Mendez, J., Ecke, A., Turhan, A.-Y.: Implementing completion-based inferences for the $\mathcal{EL}$-family. In Rosati, R., Rudolph, S., Zakharyaschev, M., eds.: Proc. of the 2011 Description Logic Workshop (DL 2011). Volume 745., CEUR (2011)
16. Ecke, A.: Completion-based role-depth bounded least common subsumer for extensions of $\mathcal{EL}$. Belegarbeit, TU Dresden (2012) Available from http://lat.inf.tu-dresden.de/ turhan/Teaching/AE-Beleg-12.pdf.