Technische Universität Dresden
Fakultät Informatik
Institut für Theoretische Informatik

Completion-based Role-depth Bounded Least Common Subsumer for Extensions of $\mathcal{EL}$

Andreas Ecke

January 31, 2012

Advisor: Dr. Ing. Anni-Yasmin Turhan
Overseeing Professor: Prof. Dr.-Ing. Franz Baader

1 Introduction

Description Logics (DLs) are a formalism for knowledge representation. They can be used to describe the terminological knowledge of an application domain in a way that makes it possible to derive implicit knowledge from the explicit description. Description logics are the basis of modern ontology languages such as OWL and are widely used in subjects like life sciences and the semantic web.

The terminology of an application domain is expressed in a TBox, which consists of several axioms. Each axiom is based on atomic concepts and roles, which can be used to create more complex concept descriptions by applying constructors. The axioms describe the relationship between concept descriptions and roles. For example

$$\text{Myocarditis} \sqsubseteq \text{inflammation} \sqcap \exists \text{has-location.heart}$$

expresses that myocarditis is a kind of inflammation located in the heart. In this case, myocarditis, inflammation and heart are atomic concepts and has-location is a role.

Any DL system implements a variety of reasoning services, inference algorithms that can derive certain implicit knowledge from ontologies. Some of the commonly used reasoning services are concept subsumption and classification. Subsumption tests whether a concept description is a sub-concept of another concept description. Classification computes all subsumption relations between atomic concepts of the ontology.

Reasoning in many of the descriptions logics is untractable. One notable exception is the description logic $\mathcal{EL}$, for which many of the reasoning problems, among others subsumption and classification, can be solved in polynomial time. One of the extensions of $\mathcal{EL}$ that still retains tractability is $\mathcal{EL}^+$. Despite being quite limited, the expressivity of $\mathcal{EL}^+$ is enough for some of the large bio-medical ontologies like SNOMED and the Gene Ontology[1].

Another popular bio-medical ontology is GALEN[2]. This ontology can be represented in the DL $\mathcal{ELHI}f_{\mathcal{R}^+}$, an extension of $\mathcal{EL}$ for which subsumption w.r.t. a general TBox is know to be EXPTIME-complete[1].

In recent years reasoners for $\mathcal{EL}^+$ that satisfy the upper polynomial time bound for classification were developed[2]. These algorithms work by exhaustively applying completion rules to a so-called completion graph. The algorithms have also been extended to $\mathcal{ELHI}f_{\mathcal{R}^+}$[11]. While not working polynomial time anymore, it still works on a completion graph structure.

These completion graphs have been employed for the computation of other non-standard inferences such as the least common subsumer (lcs) and most specific concept (msc) for $\mathcal{EL}$ [9]. The least common subsumer is a generalization inference that, given a set of concept descriptions, computes a single concept description that subsumes all these and is least w.r.t. subsumption. Intuitively, it computes what the concept descriptions have in common. Since for $\mathcal{EL}$-TBoxes the lcs does not need to exist, the introduced algorithms only compute an approximation that is least for a given role-depth bound. The generalizations can be effectively used for engineering knowledge bases in a bottom-up approach[6] and other applications like similarity measures.

[1] `http://www.geneontology.org/`
[2] `http://www.opengalen.org`

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	$r_1 \circ \ldots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \ldots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

Table 1: Constructors and axioms for $\mathcal{EL}$ and some extensions

This paper extends the algorithm for the role-depth bounded least common subsumer to $\mathcal{EL}^+$ and $\mathcal{ELI}$, a fragment of $\mathcal{ELHI}f_{\mathcal{R}^+}$. It also presents an implementation of the algorithm for $\mathcal{EL}^+$ and shows optimization and simplification procedures which improve the usability of the implementation and make the result easier to understand.

2 Description Logics with Existential Restrictions

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 Table 1. In particular, $\mathcal{EL}$-concept descriptions only allow for conjunctions, existential restrictions, and the top concept $\top$. The only allowed axioms are *general concept inclusions* (GCIs).

$\mathcal{EL}^+$ additionally allows for *role inclusion axioms*. These role inclusions can express role hierarchies (of the form $s \sqsubseteq r$) and transitive roles ($r \circ r \sqsubseteq r$). In addition to $\mathcal{EL}$, $\mathcal{ELI}$-concept descriptions may have inverse roles, but no role inclusion axioms.

The semantics of $\mathcal{EL}$ and its extensions is defined by *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ which consist of a non-empty set $\Delta^{\mathcal{I}}$, called the *domain*, and the *interpretation function* $\cdot^{\mathcal{I}}$ that maps every concept name $A \in N_C$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ of the domain and every role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation functions are extended to complex concept descriptions and axioms as described in the last column of Table 1.

If an interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ or a role inclusion axiom $r_1 \circ \ldots \circ r_k \sqsubseteq r$ we write $\mathcal{I} \models C \sqsubseteq D$ or $\mathcal{I} \models r_1 \circ \ldots \circ r_k \sqsubseteq r$. A *TBox* is a set of axioms. An interpretation $\mathcal{I}$ is a *model* for the TBox $\mathcal{T}$, if it satisfies all the axioms in $\mathcal{T}$. We say that a concept description C is *subsumed by* a concept description D w.r.t. $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ $\mathcal{I} \models C \sqsubseteq D$ holds, and similary a role r is subsumed by a role s w.r.t. $\mathcal{T}$ (denoted $r \sqsubseteq_{\mathcal{T}} s$) if for every model $\mathcal{I}$ of $\mathcal{T}$ $\mathcal{I} \models r \sqsubseteq s$ holds.

We denote $N_{C,\mathcal{T}}$ and $N_{R,\mathcal{T}}$ the sets of concept names (including $\top$) and role names that occur in a TBox $\mathcal{T}$.

3 Role-depth Bounded Least Common Subsumer

The *least common subsumer* (lcs) for concept descriptions C_1 to C_n is a concept description that subsumes all of $C_1, \ldots, C_n$ w.r.t. a given TBox, and is the least one w.r.t. subsumption. Formally, this can be defined as follows:

Definition 1 (Least Common Subsumer)**.** Let $\mathcal{T}$ be a TBox and $C_1, \ldots, C_n$ be concept descriptions. Then the concept description D is the least common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ iff

1. $C_i \sqsubseteq_{\mathcal{T}} D$ for all $i \in \{1, \ldots, n\}$, and
2. for all concept descriptions E with $C_i \sqsubseteq_{\mathcal{T}} E$ for all $i \in \{1, \ldots, n\}$ we have $D \sqsubseteq_{\mathcal{T}} E$.

Because TBoxes as defined here can contain cycles, the least common subsumer does not always exist [3]. For example, consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A \sqcap C, B \sqsubseteq \exists r.B \sqcap C\}$. The least common subsumer of A and B will have the form $C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \ldots$ and can not be expressed as a finite concept description.

To avoid infinite unraveling of cycles, the notion of role-depth bounds has been introduced. For the role-depth bounded least common subsumers, we will just look for the best least common subsumer or most specific up too a given bound. The *role-depth* $rd(C)$ of a concept description C is recursively defined as follows:

- 0, if C is the top concept $\top$ or a concept name
- $max(rd(D), rd(E))$, if C is a conjunction of the form $D \sqcap E$
- $1 + rd(D)$, if C is an existential restriction of the form $\exists r.D$.

Using this, we can define the role-depth bounded version of the least common subsumer.

Definition 2 (Role-depth bounded least common subsumer)**.** Let $\mathcal{T}$ be a TBox, $C_1, \ldots, C_n$ concept descriptions and $k \in \mathbb{N}$. Then a concept description D is the role-depth bounded least common subsumer of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ and the role-depth k (written $k{-}lcs(C_1, \ldots, C_n)$) iff

1. $rd(D) \leq k$
2. $C_i \sqsubseteq_{\mathcal{T}} D$ for all $i \in \{1, \ldots, n\}$
3. for all concept descriptions E with $rd(E) \leq k$ and $C_i \sqsubseteq_{\mathcal{T}} E \; \forall i \in \{1, \ldots, n\}$, we have $D \sqsubseteq_{\mathcal{T}} E$.

4 Completion-based Subsumption

The algorithms to compute the role-depth bounded lcs we present rely on completion sets that explicitly store subsumption relations between all concept names. This chapter introduces the completion-based subsumption algorithms for $\mathcal{ELI}$ and $\mathcal{EL}^+$, which compute these completion sets.

The completion algorithms work on a graph (V, E, S), where V is the set of nodes, E is the set of role labeled edges ($E \subseteq V \times N_{R,\mathcal{T}} \times V$) and S is a node-labeling which corresponds to the set of subsuming concepts for each node ($S(u) \subseteq N_{C,\mathcal{T}}$ for each $u \in V$). The algorithms work in three steps: First, the TBox is normalized and an initial graph (V, E, S) is constructed. In the second step, this graph is completed using the axioms of the normalized TBox. After the completion, the subsumption relations can be directly read off from the graph, thus giving a classification of the normalized TBox. When we speak of a *completion graph* of a TBox $\mathcal{T}$, we mean the graph (V, E, S) from the completion algorithm for $\mathcal{T}$ after all completion rules have been exhaustively applied.

4.1 Normalization

Definition 3 (Normal form)**.** An $\mathcal{EL}^+$ ($\mathcal{ELI}$) TBox $\mathcal{T}$ is in normal form, if all concept inclusions in $\mathcal{T}$ have one of the following forms

$$\begin{aligned} A &\sqsubseteq B \\ A_1 \sqcap A_2 &\sqsubseteq B \\ A &\sqsubseteq \exists r.B \\ \exists r.A &\sqsubseteq B \end{aligned}$$

where $A, A_1, A_2, B \in N_{C,\mathcal{T}}$ and r is role (or an inverse role for $\mathcal{ELI}$). Additionally, for $\mathcal{EL}^+$, all role inclusion axioms must have the form $s \sqsubseteq r$ or $s \circ t \sqsubseteq r$.

All TBoxes can be normalized by exhaustively applying the following normalization rules.

NF1 $C \sqcap \hat{D} \sqsubseteq E \longrightarrow \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E$

NF2 $\exists r.\hat{C} \sqsubseteq D \longrightarrow \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D$

NF3 $\hat{C} \sqsubseteq \hat{D} \longrightarrow \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}$

NF4 $B \sqsubseteq \exists r.\hat{C} \longrightarrow B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}$

NF5 $B \sqsubseteq C \sqcap D \longrightarrow B \sqsubseteq C, B \sqsubseteq D$

where $C, D, E, \hat{C}, \hat{D}$ are concept descriptions over $N_{C,\mathcal{T}}, N_{R,\mathcal{T}}$ such that $\hat{C}, \hat{D} \notin N_{C,\mathcal{T}}$; A is a new concept name.

For $\mathcal{EL}^+$, there is an additional rule to normalize role inclusion axioms, which introduces a new role name u:

NF6 $r_1 \circ r_2 \circ \ldots \circ r_n \sqsubseteq s \longrightarrow r_1 \circ r_2 \circ \ldots \circ r_{n-1} \sqsubseteq u, u \circ r_n \sqsubseteq s$

The normalized TBox $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$ as stated in [9].

4.2 Completion for $\mathcal{EL}^+$

The completion algorithm for $\mathcal{EL}^+$ was introduced by [5] and extended by [1]. Let $\mathcal{T}$ be a normalized $\mathcal{EL}^+$-TBox. Then the set of nodes V for the graph (V, E, S) of $\mathcal{T}$ is fixed with $V = N_{C,\mathcal{T}}$, E and S are initialized with $E := \emptyset$ and $S(A) := \{A, \top\}$ for all $A \in N_{C,\mathcal{T}}$. The graph is completed by exhaustively applying the following completion rules:

CR1 If $A_1 \in S(A)$, $A_1 \sqsubset B \in \mathcal{T}$ and $B \notin S(A)$,
then $S(A) := S(A) \cup \{B\}$

CR2 If $A_1, A_2 \in S(A)$, $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $B \notin S(A)$,
then $S(A) := S(A) \cup \{B\}$

CR3 If $A_1 \in S(A)$, $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $(A, r, B) \notin E$,
then $E := E \cup \{(A, r, B)\}$

CR4 If $(A, r, B) \in E$, $B_1 \in S(B)$, $\exists r.B_1 \sqsubseteq C \in \mathcal{T}$ and $C \notin S(u)$,
then $S(A) := S(A) \cup \{C\}$

CR5 If $(A, r, B) \in E$, $r \sqsubseteq s \in \mathcal{T}$ and $(A, s, B) \notin E$,
then $E := E \cup \{(A, s, B)\}$

CR6 If $(A, r_1, B), (B, r_2, C) \in E$, $r_1 \circ r_2 \sqsubseteq s \in \mathcal{T}$ and $(A, s, C) \notin E$,
then $E := E \cup \{(A, s, C)\}$

This completion algorithm is sound and complete as shown in [4]. Specifically, we have the following two lemmas.

Lemma 1 (Soundness of completion)**.** *Given a normalized $\mathcal{EL}^+$-TBox $\mathcal{T}$ and its completion graph (V, E, S), we have for each $A, B \in V$ and $r \in E$:*

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

Lemma 2 (Completeness of completion)**.** *Given a normalized $\mathcal{EL}^+$-TBox $\mathcal{T}$ and its completion graph (V, E, S), we have for each $A, B \in V$ and $r \in E$:*

1. *If $A \sqsubseteq_{\mathcal{T}} B$ for a concept name B, then $B \in S(A)$; and*
2. *if $A \sqsubseteq_{\mathcal{T}} \exists r.B$, then $(A, r, B) \in E$.*

4.3 Completion for $\mathcal{ELI}$

The completion algorithm for $\mathcal{ELI}$ was introduced in [11]. The basic idea is that the node set V can not be fixed as in the case of $\mathcal{EL}^+$. This can be seen for 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 B, that are r-successors of A. Therefore, it would be wrong to add C to the completion set $S(B)$. To solve this problem, we need to have a varying 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 $S(u)$.

Formally, 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 for $A \in N_{C,\mathcal{T}}$ from the node set for $\mathcal{EL}^+$ would then correspond to the node $(A, \emptyset)$ in 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:

Definition 4 (Concept descriptions for nodes)**.** Let $\mathcal{T}$ be a normalized TBox and (V, E, S) its completion graph. Then we define for each node $u = (A, \phi) \in V$

$$\text{vconcept}(u) = A \sqcap \prod_{\exists r.X \in \phi} \exists r.X$$

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 $E := E \cup \{(u, r, v')\}$, $S(v') := S(v') \cup \{B_1\}$

Then we can prove that the completion algorithm for $\mathcal{ELI}$, which exhaustively applies the completion rules to the graph, is sound and complete.

4.3.1 Soundness

Lemma 3 (Soundness of completion)**.** *Given the normalized TBox $\mathcal{T}$ and its completion graph (V, E, S), we have for each $u, v \in V$, $r \in E$ and concept name C:*

1. *If $C \in S(u)$, then* $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} C$*; and*
2. *if $(u, r, v) \in E$, then* $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$

Proof. Let $(V_0, E_0, S_0), \ldots, (V_n, E_n, S_n)$ be a sequence of graphs, where (V_i, E_i, S_i) is the graph produced by the completion algorithm by the ith rule applications. The lemma is proven by induction on the number i of these rule applications.

For $i = 0$, we know $V_0 = \{(A, \emptyset) \mid A \in N_{C,\mathcal{T}}\}$, $S_0(u) = \{\top, A\}$ for each $u = (A, \emptyset) \in V_0$, and $E_0 = \emptyset$. Therefore, for (1), $C \in S_n((A, \emptyset))$ implies $C = \top$ or $C = A$, and thus $\text{vconcept}((A, \emptyset)) \sqsubseteq_{\mathcal{T}} C$. Because of $E_0 = \emptyset$, (2) holds.

For $i > 0$, we need to check how the completion graph (V_i, E_i, S_i) changed compared to $(V_{i-1}, E_{i-1}, S_{i-1})$ and prove that any additions still satisfy the claims. The additions depend on the completion rule which was applied in the ith step:

CI1 Suppose that after applying this rule, $S_i(u) = S_{i-1}(u) \cup \{B\}$. This means that there exist $A_1 \in S_{i-1}(u)$ and $A_1 \sqsubseteq B \in \mathcal{T}$. From the induction hypothesis we know that $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} A_1$, and thus also $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} B$ holds. Therefore (1) is still satisfied and, because CI1 does not change E, (2) is satisfied as well.

CI2 Suppose that after applying this rule, $S_i(u) = S_{i-1}(u) \cup \{B\}$. This means that there exist $A_1, A_2 \in S_{i-1}(u)$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$. From the induction hypothesis we know that $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} A_1$ and $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} A_2$, and thus also $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} B$ holds. Therefore (1) is still satisfied and, because CI2 does not change E, (2) is satisfied as well.

CI3 Suppose that after applying this rule, $E_i = E_{i-1} \cup \{(u, r, v)\}$. This means that there exist $A_1 \in S_{i-1}(u)$ and $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$. From the induction hypothesis we know that $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} A_1$, thus $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} \exists r.B$. Because $v = (B, \emptyset)$ for some concept name B, (2) is satisfied and, because CI3 does not $S(u)$ for any $u \in V_i$, (1) is satisfied as well.

CI4 Suppose that after applying this rule, $S_i(u) = S_{i-1}(u) \cup \{C\}$. This means that there exist $(u, r, v) \in E_{i-1}$, $B_1 \in S_{i-1}(v)$, and $\exists r.B_1 \sqsubseteq C \in \mathcal{T}$. From the induction hypothesis, we know $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$ and $\text{vconcept}(v) \sqsubseteq_{\mathcal{T}} B_1$, thus $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} \exists r.B_1$. Together with $\exists r.B_1 \sqsubseteq C \in \mathcal{T}$, this yields $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} C$. Therefore (1) is still satisfied and, because CI4 does not change E, (2) is satisfied as well.

CI5 Suppose that after applying this rule, $E_i = E_{i-1} \cup \{(u, r, v')\}$. This means that there exist $A_1 \in S_{i-1}(u)$, $(u, r, v) \in E_{i-1}$ with $v = (B, \psi)$ and $v' = (B, \psi \cup \{\exists r^-.A_1\})$, and $\exists r^-.A_1 \sqsubseteq B_1 \in \mathcal{T}$.

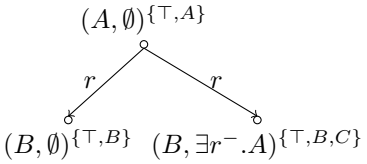
1. This rule will add v' to V if it is not already an element. In this case $S_i(v') = S_{i-1}(v) \cup \{B_1\}$. Because $v' = (B, \psi \cup \{\exists r^-.A_1\})$, $\text{vconcept}(v') \sqsubseteq_\mathcal{T} \text{vconcept}(v)$. Together with the induction hypothesis, that for each $C \in S_{i-1}(v)$ we have $\text{vconcept}(v) \sqsubseteq_\mathcal{T} C$, we also have $\text{vconcept}(v') \sqsubseteq_\mathcal{T} C$. Regardless of whether v' is added to V, the addition of B_1 to $S_i(v')$ can be justified as follows: Because $v' = (B, \psi \cup \{\exists r^-.A_1\})$, we have $\text{vconcept}(v') \sqsubseteq_\mathcal{T} \exists r^-.A_1$. Together with $\exists r^-.A_1 \sqsubseteq B_1 \in \mathcal{T}$ this yields $\text{vconcept}(v') \sqsubseteq_\mathcal{T} B_1$. Therefore (1) holds after applying CI5.
2. The addition of (u, r, v') to E_i is justified as follows. Given an model $\mathcal{I}$ of $\mathcal{T}$ and an element $x \in \text{vconcept}(u)^\mathcal{I}$, the induction hypothesis for $(u, r, v) \in E_{i-1}$ yields that $\text{vconcept}(u) \sqsubseteq_\mathcal{T} \exists r.\, \text{vconcept}(v)$ and therefore there must be $y \in \text{vconcept}(v)^\mathcal{I}$ such that $(x, y) \in r^\mathcal{I}$. This means we have $(y, x) \in r^{-\mathcal{I}}$. Because of $A_1 \in S_{i-1}(u)$ and hence $\text{vconcept}(u) \sqsubseteq_\mathcal{T} A_1$, we have $x \in A_1^\mathcal{I}$, and therefore $y \in (\exists r^-.A_1)^\mathcal{I}$. Thus $y \in (\text{vconcept}(v) \sqcap \exists r^-.A_1)^\mathcal{I} = \text{vconcept}(v')^\mathcal{I}$, $\text{vconcept}(u) \sqsubseteq_\mathcal{T} \exists r.\, \text{vconcept}(v')$ holds and (2) is satisfied.

□

4.3.2 Completeness

Simply put, we will prove the completeness by constructing a model for the TBox $\mathcal{T}$ from the completion graph that shows that all concept names that are not in the completion set $S(u)$ of a node u are not subsumers of $\text{vconcept}(u)$ and similary, when there is no r-edge between nodes u and v' with $\text{vconcept}(v') \sqsubseteq_\mathcal{T} \text{vconcept}(v)$, the subsumption $\text{vconcept}(u) \sqsubseteq_\mathcal{T} \exists r.\, \text{vconcept}(v)$ does not hold.

However, there is a problem with edges that were splitted by completion rule CI5. For example, the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.B, \exists r^-.A \sqsubseteq C\}$ yields the following description graph:



In this graph, the node $(B, \emptyset)$ would have an r-predecessor with subsumer A, but the subsumer set $\{\top, B\}$ of $(B, \emptyset)$ does not include C – thus the direct interpretation constructed from this completion graph would not be a model of $\mathcal{T}$.

Therefore we restrict the proof for completeness of the completion algorithm for $\mathcal{ELI}$ to nodes in the completion graph that are reachable by edges that were not split by CI5. For this, we define a set E_{bad} of bad edges as follows. Given a description graph (V, E, S), an edge $(u, r, v) \in E$ belongs to the bad edge set E_{bad} iff there is an $\exists r^-.A \sqsubseteq B \in \mathcal{T}$ with $A \in S(u)$ and $B \notin S(v)$. The set V_{good} is the set of all nodes $u = (A, \phi) \in V$ such that $\phi = \emptyset$ or that is reachable by edges from $E \setminus E_{bad}$ from a node $(B, \emptyset)$.

Claim 1. Given the description graph (V, E, S) and the bad edge set E_{bad}, for each $(u, r, v) \in E_{bad}$ with $v = (A, \phi)$, there is an edge $(u, r, v') \in E \setminus E_{bad}$ with $v' = (A, \psi)$ and $\phi \subset \psi$.

Proof. This claim is proved as follows. Let (u, r, v) be a bad edge with $v = (A, \phi)$. This means that there is $\exists r^-.A_1 \sqsubseteq B_1 \in \mathcal{T}$ with $A_1 \in S(u)$ and $B_1 \notin S(v)$. Then we know that $\exists r^-.A_1 \notin \phi$, otherwise completion rule CI5 would have added B_1 to $S(v)$ – a contradiction to our assumption that $B_1 \notin S(v)$.

Since $\exists r^-.A_1 \notin \phi$ and the completion rules were applied exhaustively, there must be a different node $v'' = (A, \phi \cup \{\exists r^-.A_1\}) \in V$ with $(u, r, v'') \in E$ and $B_1 \in S(v'')$. If $(u, r, v'') \in E \setminus E_{bad}$ the claim is proven. Otherwise there are still $\exists r^-.A' \sqsubseteq B' \in \mathcal{T}$ with $A' \in S(u)$ and $B' \notin S(v'')$. Each rule application of CI5 for such a GCI yields a new node $v' = (A, \psi) \in V$ with $(u, r, v') \in E$, $B' \in S(v')$ and $\phi \subset \phi \cup \{\exists r^-.A_1\} \subseteq \psi$ until there are no more GCIs $\exists r^-.A' \sqsubseteq B' \in \mathcal{T}$ with $A' \in S(u)$ and $B' \notin S(v')$. Then by the definition of E_{bad} we have $(u, r, v') \in E \setminus E_{bad}$ which proves the claim. □

Lemma 4 (Completeness of completion)**.** *Let $u \in V_{good}$. Then*

1. *if* $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} B$ *for a concept name B, then $B \in S(u)$; and*
2. *if* $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$ *for $v = (B, \psi) \in V_{good}$, then there is an v' with* $\text{vconcept}(v') \sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$ *and $(u, r, v') \in E \setminus E_{bad}$, or for each $v'' \in V_{good}$ with $(v'', r^-, u) \in E \setminus E_{bad}$ we have* $\text{vconcept}(v'') \sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$.

Proof. We will show this result by contradiction. We assume that 1. $B \notin S(u)$ and 2. there is no $(u, r, v') \in E$ or $(v', r^-, u) \in E$. We will then construct an interpretation $\mathcal{I}$ that is a model of $\mathcal{T}$ and that shows $\text{vconcept}(u) \not\sqsubseteq_{\mathcal{T}} B$ and $\text{vconcept}(u) \not\sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$.

The interpretation $\mathcal{I}$ is constructed as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}} &= V_{good} \\
A^{\mathcal{I}} &= \{u \in V_{good} \mid A \in S(u)\} \\
r^{\mathcal{I}} &= \{(u, v) \in V_{good} \times V_{good} \mid (u, r, v) \in E \setminus E_{bad}\} \\
&\quad \cup \{(v, u) \in V_{good} \times V_{good} \mid (u, r^-, v) \in E \setminus E_{bad}\}
\end{aligned}$$

We show that $\mathcal{I}$ is a model of $\mathcal{T}$. For this, we show that $\mathcal{I}$ satisfies all general concept inclusions in $\mathcal{T}$ by making a case distinction on their form.

$A \sqsubseteq B$: Let $u \in A^{\mathcal{I}}$ for $u \in V_{good}$. By definition of $\mathcal{I}$, we have $A \in S(u)$. Due to rule CI1, this implies $B \in S(u)$, thus $u \in B^{\mathcal{I}}$.

$A_1 \sqcap A_2 \sqsubseteq B$: Let $u \in (A_1 \sqcap A_2)^{\mathcal{I}} = A_1^{\mathcal{I}} \cap A_2^{\mathcal{I}}$ for $u \in V_{good}$. By definition of $\mathcal{I}$, we have $A_1, A_2 \in S(u)$. Due to rule CI2, this implies $B \in S(u)$, thus $u \in B^{\mathcal{I}}$.

$A \sqsubseteq \exists r.B$: Let $u \in A^{\mathcal{I}}$ for $u \in V_{good}$. By definition of $\mathcal{I}$, we have $A \in S(u)$. Due to rule CI3, there exists $(u, r, v) \in E$ with $v = (B, \emptyset)$. There are two cases:

- $(u, r, v) \in E \setminus E_{bad}$. From the definition of $\mathcal{I}$, we have $(u, v) \in r^{\mathcal{I}}$ and, because $B \in S((B, \emptyset))$, we have $v \in B^{\mathcal{I}}$. Together, this yields $u \in (\exists r.B)^{\mathcal{I}}$.
- $(u, r, v) \in E_{bad}$. Claim 1 says there is $v' = (B, \psi)$ such that $(u, r, v') \in E \setminus E_{bad}$ and thus also $v' \in V_{good}$. Therefore $(u, v') \in r^{\mathcal{I}}$. Because $B \in S((B, \psi))$ for all ψ, we have $v' \in B^{\mathcal{I}}$. Together, this yields $u \in (\exists r.B)^{\mathcal{I}}$.

$\exists r.A \sqsubseteq B$**:** Let $u \in (\exists r.A)^{\mathcal{I}}$ for $u \in V_{good}$, i.e., there is $v \in A^{\mathcal{I}}$ such that $(u, v) \in r^{\mathcal{I}}$. According to the definition of $r^{\mathcal{I}}$, there are two cases:

- $(u, r, v) \in E \setminus E_{bad}$. Due to rule CI4 and $A \in S(v)$, we have $B \in S(u)$, which means that $u \in B^{\mathcal{I}}$.
- $(v, r^-, u) \in E \setminus E_{bad}$. Then, with $s = r^-$, we have: $(v, s, u) \in E \setminus E_{bad}$, $\exists s^-.A \sqsubseteq B \in \mathcal{T}$ and $A \in S(v)$. Since $(v, s, u) \notin E_{bad}$, rule CI5 yields $B \in S(u)$, hence $u \in B^{\mathcal{I}}$.

Now we only need to show that this model yields 1. $\text{vconcept}(u) \not\sqsubseteq_{\mathcal{T}} B$ if $B \notin S(u)$ and 2. $\text{vconcept}(u) \not\sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$ if there is no $(u, r, v') \in E \setminus E_{bad}$ with $\text{vconcept}(v') \sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$ and there is $(v'', r^-, u) \in E$ with $\text{vconcept}(v'') \not\sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$.

1. First we show that $u = (A, \phi) \in \text{vconcept}(u)^{\mathcal{I}}$ for each $u \in V_{good}$. We have $A \in S((A, \phi))$ for all (A, ϕ), i.e., $u \in A^{\mathcal{I}}$. Additionally, if $\phi \neq \emptyset$, we know that u has an incident edge in $E \setminus E_{bad}$ because $u \in V_{good}$. Thus, there must be a node v, such that for each $\exists r.X \in \phi$ it holds that $X \in S(v)$ and $(v, r^-, u) \in E \setminus E_{bad}$ because these $\exists r.X \in \phi$ can only be added by rule CI5 which also adds this edge. Therefore $u \in (\exists r.X)^{\mathcal{I}}$ for each $\exists r.X \in \phi$. This yields $u \in \text{vconcept}(u)^{\mathcal{I}}$.

   Since $B \notin S(u)$, we have, by definition of $\mathcal{I}$, $u \notin B^{\mathcal{I}}$. Therefore u shows that $\text{vconcept}(u)^{\mathcal{I}} \not\subseteq B^{\mathcal{I}}$ and thus $\text{vconcept}(u) \not\sqsubseteq_{\mathcal{T}} B$.

2. Since $\text{vconcept}(u) = A \sqcap \bigsqcap_{\exists r.X \in \phi} \exists r.X$, the subsumption $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$ must be due to one of two reasons:

   a) There is an $\exists r.X \in \phi$ with $X \sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$. Then, due to how rule CI5 works, for all $v' \in V_{good}$ with $(v', r^-, u) \in E \setminus E_{bad}$ we would have $X \in S(v')$, i.e. $v' \in X^{\mathcal{I}}$ and hence $v' \in \text{vconcept}(v)^{\mathcal{I}}$. This is a contradiction to our assumption that for some $v' \in V_{good}$ with $(v', r^-, u) \in E \setminus E_{bad}$ we have $\text{vconcept}(v') \not\sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$.

   b) $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$ follows from rule $Y \sqsubseteq \exists r.Z$ and possibly some rules $\exists r^-.Y' \sqsubseteq Z'$ for $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} Y \sqcap \bigsqcap Y'$ and $Z \sqcap \bigsqcap r^-.Y' \sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$. We already showed that if $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} Y$ resp. $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} Y'$, then $Y \in S(u)$ resp. $Y' \in S(u)$. Then rule CI3 for $Y \sqsubseteq \exists r.Z$ and rules CI5 for $\exists r^-.Y' \sqsubseteq Z'$ will yield a node $v' = (Z, \psi)$ with $(u, r, v') \in E$, $Z \in S(v')$ and $\exists r^- Y' \in \psi$. Thus $v' \in \text{vconcept}(v)^{\mathcal{I}}$ which, together with claim 1, is a contradiction to our assumption that there is no v' with $\text{vconcept}(v') \sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$ and $(u, r, v') \in E \setminus E_{bad}$.

   Therefore, we have $\text{vconcept}(u) \not\sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$.

□

The soundness and completeness results for the completion algorithms are important for the correctness of the algorithms for the role-depth bounded least common subsumer, which will be presented in the next section.

5 Algorithms for the Role-depth Bounded Lcs

We will now extend the algorithm to compute the role-depth bounded lcs in $\mathcal{EL}$ as given in [9] to $\mathcal{EL}^+$ and $\mathcal{ELI}$. The basic idea of this algorithm is the following: the least common subsumer for two $\mathcal{EL}$ concept descriptions without an underlying TBox can be computed by calculating the product of their corresponding description trees [7]. With respect to a TBox $\mathcal{T}$, we can construct the least common subsumer of two concept descriptions by extending the cross-product construction up to the role-depth bound to completion sets. These completion sets will then be combined and intersected in an appropriate way.

Because the completion sets are constructed from the normalized TBox, the resulting concept descriptions may contain normalization names that were not part of the original TBox. This can be expressed by the *signature* of the resulting concept description. For a concept C, the signature of C (denoted $\mathrm{sig}(C)$) is the set of concept names and role names that appear in C. Similarly, the signature of a TBox $\mathcal{T}$ (denoted $\mathrm{sig}(\mathcal{T})$) is the set of concept names and role names that appear in $\mathcal{T}$. Using this, the following two lemmas prove that the normalization names, that occur only in the signature of the normalized TBox, may be simply removed from the concept description.

Lemma 5 (Denormalization 1)**.** *Let $\mathcal{T}$ be an $\mathcal{EL}^+$- or $\mathcal{ELI}$-TBox and $\mathcal{T}'$ be the TBox obtained from $\mathcal{T}$ by applying the normalization rules, C, D be concept descriptions with $\mathrm{sig}(C) \subseteq \mathrm{sig}(\mathcal{T})$, $\mathrm{sig}(D) \subseteq \mathrm{sig}(\mathcal{T}')$ and let D' be the concept description obtained from D by removing all normalization names $A \in \mathrm{sig}(\mathcal{T}') \setminus \mathrm{sig}(\mathcal{T})$ and all existential restrictions $\exists r.E$ for normalization names $r \in \mathrm{sig}(\mathcal{T}') \setminus \mathrm{sig}(\mathcal{T})$. Then $C \sqsubseteq_{\mathcal{T}'} D$ implies $C \sqsubseteq_{\mathcal{T}} D'$.*

Proof. Since D' is obtained from D by removing some concept names and existential restrictions, we have $D \sqsubseteq_{\mathcal{T}'} D'$. Together with $C \sqsubseteq_{\mathcal{T}'} D$, it follows that $C \sqsubseteq_{\mathcal{T}'} D'$. Because $\mathcal{T}$ and $\mathcal{T}'$ are $\mathrm{sig}(\mathcal{T})$-inseparable (i.e., for all $\mathcal{EL}$-concept descriptions C, D with $\mathrm{sig}(C) \cup \mathrm{sig}(D) \subseteq \mathrm{sig}(\mathcal{T})$, we have $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}'} D$, for a proof see [8]), we finally get $C \sqsubseteq_{\mathcal{T}} D'$. □

The first denormalization result shows that if we denormalize the least common subsumer computed in the normalized TBox, it is still a common subsumer in the original TBox. To show that the denormalization also retains minimality, we need the additional restriction that the least common subsumer is *fully expanded*.

Definition 5. Let $\mathcal{T}$ be a TBox and C be a concept description with $\mathrm{sig}(C) \subseteq \mathrm{sig}(\mathcal{T})$. Then C is fully expanded up to the role-depth k w.r.t. $\mathcal{T}$, if

– for all concept names $A \in N_{C,\mathcal{T}}$ with $C \sqsubseteq_{\mathcal{T}} A$ we have that A is a conjunct of C and

– if $k > 0$ then for all concept descriptions F with $C \sqsubseteq_{\mathcal{T}} \exists r.F$ we have a concept description F' with $F' \sqsubseteq_{\mathcal{T}} F$ such that $\exists r.F'$ is a conjunct of C and F' is fully expanded up to role-depth $k-1$.

Lemma 6 (Denormalization 2)**.** *Let $\mathcal{T}$ be an $\mathcal{EL}^+$- or $\mathcal{ELI}$-TBox and $\mathcal{T}'$ be the TBox obtained from $\mathcal{T}$ by applying the normalization rules, C, D be concept descriptions with $\mathrm{sig}(C) \subseteq \mathcal{T}'$, $\mathrm{sig}(D) \subseteq \mathcal{T}$ and let C' be the concept description obtained from C by removing all normalization names $A \in \mathrm{sig}(\mathcal{T}') \setminus \mathrm{sig}(\mathcal{T})$ and all existential restrictions $\exists r.E$ for normalization names $r \in \mathrm{sig}(\mathcal{T}') \setminus \mathrm{sig}(\mathcal{T})$. Let further $k = rd(D)$ and let C be fully expanded up to the role-depth k w.r.t. $\mathcal{T}'$. Then $C \sqsubseteq_{\mathcal{T}'} D$ implies $C' \sqsubseteq_{\mathcal{T}} D$.*

Proof. Since C is fully expanded up to the role-depth of D, and $C \sqsubseteq'_{\mathcal{T}} D$, any part of D must also be in C. Since $\mathrm{sig}(D) \subseteq \mathcal{T}$, i.e. D does not contain any auxiliary concept or role names, we can remove any of those from C without affecting subsumption. Hence also $C' \sqsubseteq_{\mathcal{T}'} D$, and because $\mathcal{T}$ and $\mathcal{T}'$ are $\mathrm{sig}(\mathcal{T})$-inseparable, we have $C' \sqsubseteq_{\mathcal{T}} D$. □

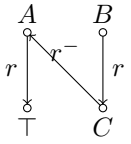
5.1 Role-depth Bounded Lcs for $\mathcal{EL}^+$

Algorithm 1 computes the role-depth bounded least common subsumer for two concept descriptions C and D.

This algorithm indeed yields the role-depth bounded least common subsumer, as proven in [8]. Although this proof is for $\mathcal{EL}$ and not $\mathcal{EL}^+$ it is directly applicable here, as the role inclusion axioms only matter for the construction of the completion graph, not for the construction of the lcs. This can be seen since for example for a completion graph (V, E, S), rule CR5 constructs for each edge $(A, r, B) \in E$ and each super-role $r \sqsubseteq s$ a new edge $(A, s, B) \in E$, therefore all role inclusion axioms are explicitely included in the completion graph and do not need special treatment during the k-lcs construction.

5.2 Role-depth Bounded Lcs for $\mathcal{ELI}$

$\mathcal{ELI}$ allows for inverse roles. When traversing the completion graph, we may also traverse an edge we just came from backwards (using the inverse role of the role that the edge is labeled with). This is important, as you can see in the example for $\mathcal{T} = \{A \sqsubseteq \exists r.\top, B \sqsubseteq \exists r.C, C \sqsubseteq \exists r^-.A\}$. This will result in the following completion graph:



Traversing the 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.

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

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** $\displaystyle\bigsqcap_{P \in \text{common-names}} P$
4: **else**
5: **return** $\displaystyle\bigsqcap_{P \in \text{common-names}} P \sqcap \bigsqcap_{r \in N_R} \Big(\bigsqcap_{(A,r,C) \in E, (B,r,D) \in E} \exists r.\text{k-lcs-r}(C, D, (V, E, S), k-1) \Big)$
6: **end if**

However, the lcs of A and B is $\exists r.\exists r^-.A$, therefore we must allow the algorithm to go backwards from $\top$ to A using the edge $(A, r, \top)$ as $(\top, r^-, A)$ – this yields the correct lcs. However, we can not go backwards along arbitrary edges; to go from A to C using the edge (C, r^-, A) as (A, r, C) would clearly be wrong, as we don't have $A \sqsubseteq_{\mathcal{T}} \exists r.C$. The algorithm may only go back edges it already traversed.

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 node $u \in V$ and 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 6 (Concept descriptions for paths)**.** Let $\mathcal{T}$ be a normalized TBox and (V, E, S) its completion graph. Then we define for each path $l = [u_0, r_1, u_1, r_2, \ldots, r_n, u_n]$

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

Algorithm 2 computes the role-depth bounded least common subsumer for two concept descriptions C and D. Basically, this algorithm is very similar to Algorithm 1 for $\mathcal{EL}^+$, with three main differences:

– Algorithm 2 uses the whole path to the current nodes instead of the node itself.

– Whereas in algorithm 1 the nodes we visit from the current nodes are computed implicitly, algorithm 2 stores any such successor node of the nodes A and B (or better: the path to those) explicitly in the sets S_1 and S_2.

– Algorithm 2 also traverses all edges (A, r, B) from the current node A, but additionally traverses the last edge backwards, if it is the inverse of r.

Now we will prove that Algorithm 2 is correct for $\mathcal{ELI}$.

Algorithm 2 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: lcs $:= \prod_{C \in S((A_n,\phi_n)) \cap S((B_m,\psi_m))} C$
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: **end if**
8: $S_2 := \{[(B_0, \emptyset), s_1, \ldots, s_n, (B_m, \psi_m), r, (B, \psi)] \mid ((B_m, \psi_m), r, (B, \psi)) \in E\}$
9: **if** $n > 0 \wedge r = s_m^-$ **then**
10: $S_2 := S_2 \cup \{[(B_0, \emptyset), s_1, (B_1, \psi_1), s_2, \ldots, (B_{m-2}, \psi_{m-2}), s_{m-1}, (B_{m-1}, \psi_{m-1})]\}$
11: **end if**
12: lcs $:=$ lcs $\sqcap \prod_{\substack{l_1 \in S_1 \\ l_2 \in S_2}} \exists r.$ k-lcs-r$(l_1, l_2, (V, E, S), k-1)$
13: **end for**
14: **end if**
15: **return** lcs

5.2.1 Common Subsumer

First we prove that k-lcs-r for $\mathcal{ELI}$ indeed yields a common subsumer.

Lemma 7. *Let* $L = \text{k-lcs-r}(p_1, p_2, (V, E, S), k)$ *for the 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* $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} L$ *and* $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} L$.

Proof. We prove this by induction on the role-depth bound k.

Case $k = 0$**:** L is a conjunction of concept names $\bigsqcap_{C \in S((A_n,\phi_n)) \cap S((B_m,\psi_m))} C$. By Lemma 3 we know $\text{vconcept}((A_n, \phi_n)) \sqsubseteq_{\mathcal{T}} C$ for each $C \in S((A_n, \phi_n))$ and hence $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \text{vconcept}((A_n, \phi_n)) \sqsubseteq_{\mathcal{T}} \bigsqcap_{C \in S((A_n,\phi_n))} C \sqsubseteq_{\mathcal{T}} L$. Similarly, $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} \text{vconcept}((B_m, \psi_m)) \sqsubseteq_{\mathcal{T}} \bigsqcap_{C \in S((B_m,\psi_m))} C \sqsubseteq_{\mathcal{T}} L$.

Case $k > 0$**:** Assume that the result holds for the role-depth bound $k-1$. We have to show that it is also correct for k. L is a conjunction of $\bigsqcap_{C \in S((A_n,\phi_n)) \cap S((B_m,\psi_m))} C$ and existential restrictions. For the concept names $\bigsqcap_{C \in S((A_n,\phi_n)) \cap S((B_m,\psi_m))} C$ the same arguments as in the base case can be applied, yielding $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \bigsqcap_{C \in S((A_n,\phi_n)) \cap S((B_m,\psi_m))} C$. It remains to show that the conjunction of existential restrictions is a subsumer of $\text{lconcept}(p_1)$ and $\text{lconcept}(p_2)$.

All existential restrictions are of the form $\exists r.\, \text{k-lcs-r}(l_1, l_2, (V, E, S), k-1)$ where l_1 is either $[u_0, r_1, u_1, r_2, \ldots, r_n, u_n, r, u]$ for $(u_n, r, u) \in E$ or $[u_0, r_1, u_1, r_2, \ldots, r_{n-1}, u_{n-1}]$ for $r = r_n^-$. In the first case Lemma 3 for $(u_n, r, u) \in E$ yields $\text{vconcept}(u_n) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(u)$ and hence $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \text{vconcept}(u_n) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(u)$ and thus $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \exists r.(\text{vconcept}(u) \sqcap \exists r^-.\, \text{lconcept}(p_1)) = \exists r.\, \text{lconcept}(l_1)$. In the second case $\text{lconcept}(p_1) = \text{vconcept}(u_n) \sqcap \exists r_n^-.\, \text{lconcept}(l_1) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{lconcept}(l_1)$. Therefore in both cases we have $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{lconcept}(l_1)$. The same argument holds for $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{lconcept}(l_2)$. Because of the induction hypothesis we know $\text{lconcept}(l_1) \sqsubseteq_{\mathcal{T}} \text{k-lcs-r}(l_1, l_2, (V, E, S), k-1)$ and $\text{lconcept}(l_2) \sqsubseteq_{\mathcal{T}} \text{k-lcs-r}(l_1, l_2, (V, E, S), k-1)$, therefore $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{lconcept}(l_1) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{k-lcs-r}(l_1, l_2, (V, E, S), k-1)$ and $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{lconcept}(l_2) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{k-lcs-r}(l_1, l_2, (V, E, S), k-1)$ which completes the proof that $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} L$ and $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} L$.

□

Corollary 1. *Let* $\mathcal{T}$ *be a* $\mathcal{ELI}$*-TBox,* C, D $\mathcal{ELI}$*-concept descriptions w.r.t.* $\mathcal{T}$, *and* $L = \text{k-lcs}(C, D, \mathcal{T}, k)$. *Then* $C \sqsubseteq_{\mathcal{T}} L$ *and* $D \sqsubseteq_{\mathcal{T}} L$.

Proof. Let $\mathcal{T}'$ be the normalized TBox of $\mathcal{T} \cup \{A \equiv C, B \equiv D\}$, (V, E, S) be the completion graph of $\mathcal{T}'$, and $L' = \text{k-lcs-r}([(A, \emptyset)], [(B, \emptyset)], (V, E, S), k)$. Then $C \equiv_{\mathcal{T}'} A = \text{lconcept}([(A, \emptyset)]) \sqsubseteq_{\mathcal{T}'} L'$ and $D \equiv_{\mathcal{T}'} B = \text{lconcept}([(B, \emptyset)]) \sqsubseteq_{\mathcal{T}'} L'$ by Lemma 7. Together with Lemma 5, this yields that $C \sqsubseteq_{\mathcal{T}} \text{remove-normalization-names}(L') = L$ and $D \sqsubseteq_{\mathcal{T}} \text{remove-normalization-names}(L') = L$. □

5.2.2 Minimality

Now that we know that k-lcs-r yields a common subsumer, we will show that this is the least one w.r.t. $\sqsubseteq_{\mathcal{T}}$.

Lemma 8. *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 be a natural number and F a concept description with $rd(F) \leq k$. If* $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} F$ *and* $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} F$ *then* $L = \text{k-lcs-r}(p_1, p_2, (V, E, S), k) \sqsubseteq_{\mathcal{T}} F$.

Proof. We prove the claim by induction on the role-depth bound k.

Case $k = 0$**:** F must be a conjunction $F = A_1 \sqcap \ldots \sqcap A_n$. Since $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} F$ and $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} F$, we have $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} A_i$ and $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} A_i$ for all $1 \leq i \leq n$. Since p_1 and p_2 only traverse edges over $E \setminus E_{bad}$, all possible rule applications of CI5 during that path were applied, and we have $\text{vconcept}(u_n) \sqsubseteq_{\mathcal{T}} A_i$ and $\text{vconcept}(v_m) \sqsubseteq_{\mathcal{T}} A_i$. Then Lemma 4 yields $A_i \in S(u_n)$ and $A_i \in S(v_m)$ for all $1 \leq i \leq n$, i.e., $A_i \in \bigsqcap_{C \in S((A_n, \phi_n)) \cap S((B_m, \psi_m))} C$. Thus, $L \sqsubseteq_{\mathcal{T}} F$.

Case $k > 0$**:** F is a conjunction of concept names and existential restrictions. The concept names in F must appear in L by the same arguments as in the base case.

Let $\exists r.F'$ be a top-level conjunct of F. Since $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} F$ and $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} F$, we have $\text{lconcept}(p_1) \sqsubseteq_{\mathcal{T}} \exists r.F'$ and $\text{lconcept}(p_2) \sqsubseteq_{\mathcal{T}} \exists r.F'$, and, because p_1 and p_2 only traverse edges over $E \setminus E_{bad}$, there must be nodes u and v with $\text{vconcept}(u) \sqsubseteq_{\mathcal{T}} F'$, $\text{vconcept}(v) \sqsubseteq_{\mathcal{T}} F'$, such that $\text{vconcept}(u_n) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(u)$ and $\text{vconcept}(v_m) \sqsubseteq_{\mathcal{T}} \exists r.\, \text{vconcept}(v)$. Lemma 4 yields that there are u' and v' in V_{good} 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 $\text{vconcept}(u') \sqsubseteq_{\mathcal{T}} \text{vconcept}(u)$ and $\text{vconcept}(v') \sqsubseteq_{\mathcal{T}} \text{vconcept}(v)$. Therefore, there are new paths $l_1 \in S_1$ and $l_2 \in S_2$, such that $\text{lconcept}(l_1) \sqsubseteq_{\mathcal{T}} F'$ and $\text{lconcept}(l_2) \sqsubseteq_{\mathcal{T}} F'$. Since $(u_n, r, u'), (v_m, r, v') \in E \setminus E_{bad}$ if we add an edge, i.e. the paths are still *good*, the induction hypothesis yields $\text{k-lcs-r}(l_1, l_2, (V, E, S), k-1) \sqsubseteq_{\mathcal{T}} F'$, i.e. $\exists r.\, \text{k-lcs-r}(l_1, l_2, (V, E, S), k-1) \sqsubseteq_{\mathcal{T}} \exists r.F'$, and thus $L = \text{k-lcs-r}(p_1, p_2, (V, E, S), k) \sqsubseteq_{\mathcal{T}} F$.

□

Corollary 2. *Let $\mathcal{T}$ be a $\mathcal{ELI}$-TBox, C, D $\mathcal{ELI}$-concept descriptions w.r.t. $\mathcal{T}$, $L = \text{k-lcs}(C, D, \mathcal{T}, k)$ and F be a $\mathcal{ELI}$-concept description with $rd(F) \leq k$. If $C \sqsubseteq_{\mathcal{T}} F$ and $D \sqsubseteq_{\mathcal{T}} F$, then $L \sqsubseteq_{\mathcal{T}} F$.*

Proof. Let $\mathcal{T}'$ be the normalized TBox of $\mathcal{T} \cup \{A \equiv C, B \equiv D\}$, and (V, E, S) be the completion graph of $\mathcal{T}'$. Then $\text{lconcept}([(A, \emptyset)]) = A \equiv_{\mathcal{T}'} C \sqsubseteq_{\mathcal{T}'} F$ and $\text{lconcept}([(B, \emptyset)]) = B \equiv_{\mathcal{T}'} D \sqsubseteq_{\mathcal{T}'} F$ by assumption. Then Lemma 8 yields $L' = \text{k-lcs-r}([(A, \emptyset)], [(B, \emptyset)], (V, E, S), k) \sqsubseteq_{\mathcal{T}'} F$. Because L' is fully extended up to role-depth k, the application of Lemma 6 yields that $L = \text{remove-normalization-names}(L') \sqsubseteq_{\mathcal{T}} F$. □

6 Simplification

The algorithms for the least common subsumer yield highly redundant concept descriptions. These descriptions can be simplified, which makes the results much easier to understand. The general idea for the simplification is to interpret the concept description as a tree structure. Then we can simply remove subtrees which are subsumers of any sibling subtrees – they do not contain more information. For a conjunction of concept names, this means that we remove all these that subsume other concept names and only keep the least ones (regarding $\sqsubseteq_{\mathcal{T}}$).

Algorithm 3 computes the simplification of a concept description for $\mathcal{EL}^+$, Algorithm 4 determines if a concept description is a subsumer of the other one. Note that the algorithm may only 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 – this is clearly wrong.

Algorithm 3 Simplification

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: $R := \{A_i \mid 1 \leq i \leq n\} \cup \{\exists r_j.D_j \mid 1 \leq j \leq m\}$
3: **for all** $X \in R$ **do**
4: **for all** $Y \in R$ **do**
5: **if** $X \neq Y \wedge$ subsumes$(X, Y, (V, E, S), \mathcal{T})$ **then**
6: $R := R \setminus \{X\}$
7: **break**
8: **end if**
9: **end for**
10: **end for**
11: **for all** $X \in R$ **do**
12: **if** $X \equiv \exists r_j.D_j$ **then**
13: $R := (R \setminus \{\exists r_j.D_j\}) \cup \{\exists r_j.\text{simplify}(D_j, (V, E, S), \mathcal{T})\}$
14: **end if**
15: **end for**
16: **return** $\bigsqcap_{X \in R} X$

For the simplification to be correct, we only have to proof that the subsumer-procedure is sound – that is, whenever it determines that one concept subsumes another one, this must be indeed true. However, it may also return false if the first concept subsumes the other one (i.e., it does not have to be complete), in which case the subsumer does not get removed. This might yield a more redundant simplification, but it is still a correct role-depth bounded lcs.

To prove that the subsumer-procedure is sound, we use a simple case distinction, as both concept descriptions C and D might be either concept names, conjunctions

Algorithm 4 Subsumes

Procedure subsumes$(C, D, (V, E, S), \mathcal{T})$
Input: C, D: $\mathcal{EL}^+$ concept descriptions; (V, E, S): completion graph; $\mathcal{T}$: $\mathcal{EL}^+$ TBox
Output: whether C subsumes D regarding $\mathcal{T}$

1: **if** $C \in N_C$ **then**
2: **if** $C = \top$ **then**
3: **return** true
4: **else if** $D \in N_C$ **then**
5: **return** $C \in S(D)$
6: **else if** $D \equiv F_1 \sqcap \ldots \sqcap F_n$ **then**
7: **for all** $1 \leq i \leq n$ **do**
8: **if** subsumes$(C, F_i, (V, E, S), \mathcal{T})$ **then**
9: **return** true
10: **end if**
11: **end for**
12: **end if**
13: **return** false
14: **else if** $C \equiv F_1 \sqcap \ldots \sqcap F_n$ **then**
15: **for all** $1 \leq i \leq n$ **do**
16: **if** not subsumes$(F_i, D, (V, E, S), \mathcal{T})$ **then**
17: **return** false
18: **end if**
19: **end for**
20: **return** true
21: **else if** $C \equiv \exists r.F$ **then**
22: **if** $D \in N_C$ **then**
23: **return** $(C, r, F) \in E$
24: **else if** $D \equiv F_1 \sqcap \ldots \sqcap F_n$ **then**
25: **for all** $1 \leq i \leq n$ **do**
26: **if** subsumes$(C, F_i, (V, E, S), \mathcal{T})$ **then**
27: **return** true
28: **end if**
29: **end for**
30: **else if** $D \equiv \exists s.G$ **then**
31: **if** $s \sqsubseteq_{\mathcal{T}} r$ and subsumes$(F, G, (V, E, S), \mathcal{T})$ **then**
32: **return** true
33: **else**
34: **for all** $t \in N_R$ **do**
35: **if** $s \circ t \sqsubseteq_{\mathcal{T}} r$, $G \equiv \ldots \sqcap \exists t.H \sqcap \ldots$ and subsumes$(F, H, (V, E, S), \mathcal{T})$ **then**
36: **return** true
37: **end if**
38: **end for**
39: **end if**
40: **return** false
41: **end if**
42: **end if**

or existential restriction, and show that whenever the algorithm returns true, we have indeed $D \sqsubseteq_{\mathcal{T}} C$. For example, if C is a conjunction $C = F_1 \sqcap \ldots F_n$, the procedure only returns true if for all $1 \leq i \leq n$ we have that D is a subsumer of F_i which is equivalent to $D \sqsubseteq_{\mathcal{T}} C$. If both C and D are existential restrictions $C = \exists r.F$ and $D = \exists s.G$, then the procedure returns true if $s \sqsubseteq_{\mathcal{T}} r$ and F is a subsumer of G, or if there is a role t with $s \circ t \sqsubseteq_{\mathcal{T}} r$ and $G = \ldots \sqcap \exists t.H \sqcap \ldots$ where F is a subsumer of H. In both cases we clearly have $D \sqsubseteq_{\mathcal{T}} C$. All other cases are proven similarly.

7 Optimization

The k-lcs-r procedure constructs the fully expanded result description, most of which is removed again by the simplification procedure. For large ontologies with deep role hierarchies, the fully expanded result may grow quite large, slowing the algorithm down. Therefore, the general idea for optimization is to not generate the fully expanded description, but apply some of the simplifications already during the construction of the result.

The simplest optimization applies if one of the input concepts of the k-lcs-r procedure already subsumes the other one, in which case it must be the least common subsumer of the both. Therefore in k-lcs-r(A, B, S, k), if $A \in S(B)$ we can simply return A and if $B \in S(A)$ we can return B.

However, this optimization interacts with the denormalization and the role-depth bound in non-obvious ways. For example, consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.\exists r.K, B \sqsubseteq \exists r.\exists r.K, \exists r.\exists r.K \sqsubseteq \exists s.(L \sqcap M)\}$ which gets normalized to $\mathcal{T}' = \{A \sqsubseteq \exists r.X_1, X_1 \sqsubseteq \exists r.K, B \sqsubseteq \exists r.X_2, X_2 \sqsubseteq \exists r.K, \exists r.K \sqsubseteq X_3, \exists r.X_3 \sqsubseteq X_4, X_4 \sqsubseteq \exists s.X_5, X_5 \sqsubseteq L, X_5 \sqsubseteq M\}$. Running k-lcs-r without this optimization for role-depth 1 on A and B would yield $X_4 \sqcap \exists r.X_3 \sqcap \exists s.(X_5 \sqcap L \sqcap M)$, which denormalizes to $\top \sqcap \exists r.\top \sqcap \exists s.(L \sqcap M)$, while the optimization would yield $X_4 \sqcap \exists r.X_3 \sqcap \exists s.X_5$, which denormalizes to $\top \sqcap \exists r.\top \sqcap \exists s.\top$. This is clearly not the least common subsumer anymore!

Therefore, the optimization can only be applied to concept names from the original TBox, never to normalization names. Then the returned concept name will never be removed during the denormalization step and the resulting concept description is still a least common subsumer. This optimization and the next one are shown in Algorithm 5.

For another optimization consider the TBox $\mathcal{T}_n = \{A \sqsubseteq D \sqcap \exists r.C_1, B \sqsubseteq E \sqcap \exists r.C_1\} \cup \{C_1 \sqsubseteq C_i \mid i \in \{2 \ldots n\}\}$. In this case, the least common subsumer of A and B is just $\exists r.C_1$. However, the algorithm would generate the complete product set $\{C_i \mid i \in \{1, \ldots, n\}\} \times \{C_i \mid i \in \{1, \ldots, n\}\}$ and recursively call $k{-}lcs{-}r$ for each pair, just to eliminate all $\exists r.C_i$ for $i > 1$ and all $\exists r.\top$ afterwards in the simplification step. Even for this simple example, the algorithm would require time quadratic in the size of the input TBox. Clearly, evaluating the C_is for $i > 1$ is not necessary, as they all subsume C_1. The same is true for role hierarchies, where for example $\mathcal{T}'_n = \{A \sqsubseteq D \sqcap \exists r.C_1, B \sqsubseteq E \sqcap \exists r.C_1\} \cup \{r \sqsubseteq r_i \mid i \in \{2 \ldots n\}\}$ would lead to the same unnecessary quadratic runtime.

The idea for the optimization is to explicitly create the sets SA and SB of all

Algorithm 5 Computation of a role-depth bounded $\mathcal{EL}^+$-lcs.

Procedure k-lcs-o$(C, D, \mathcal{T}, k)$
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

$\mathcal{T}' :=$ normalize$(\mathcal{T} \cup \{A \equiv C, B \equiv D\})$
$(V, E, S) :=$ apply-completion-rules$(\mathcal{T}')$
$L :=$ k-lcs-r-o$(A, B, (V, E, S), k)$
if $L \equiv_{\mathcal{T}'} A$ **then**
  return C
else if $L \equiv_{\mathcal{T}'} B$ **then**
  return D
else
  return remove-normalization-names(L)
end if

Procedure k-lcs-r-o$(A, B, (V, E, S), k)$
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

if $A \in S(B)$ and A is not a normalization name **then**
  return A
else if $B \in S(A)$ and B is not a normalization name **then**
  return B
end if
common-names $:= S(A) \cap S(B)$
SA $:=$ remove-redundant$(\{(r, C) \mid (A, r, C) \in E\})$
SB $:=$ remove-redundant$(\{(s, D) \mid (B, s, D) \in E\})$
if $k = 0$ **then**
  return $\displaystyle\bigsqcap_{P \in \text{common-names}} P$
else
  return $\displaystyle\bigsqcap_{P \in \text{common-names}} P \sqcap \bigsqcap_{\substack{(r,C) \in SA \\ (s,D) \in SB \\ t \in N_R \text{ minimal with } r \sqsubseteq_{\mathcal{T}} t \wedge s \sqsubseteq_{\mathcal{T}} t}} \exists t.\, \text{k-lcs-r-o}(C, D, (V, E, S), k-1)$
end if

Procedure remove-redundant(S)
Input: S: set of role-successors
Output: remove-redundant(S): simplified set

for all $(r, C) \in S$ **do**
  for all $(s, D) \in S$ **do**
    if $(r, C) \neq (s, D)$ and $r \sqsubseteq_{\mathcal{T}} s$ and $D \in S(C)$ **then**
      $S := S \setminus \{(s, D)\}$
    end if
  end for
end for
return S

role-successors $(A, r, C) \in E$ and $(B, r, C) \in E$ for input concepts A and B and to remove all role-successors which are subsumers of other role-successors in the same set. Recursive calls to k-lcs-r-o can then be made with one successor from each set SA and SB. However, we have to be careful with role-successors with different role-names. For example, for a role-successor $(r, C) \in SA$ and a role-successor $(s, D) \in SB$, a recursive call has to be made for all minimal (w.r.t. $\sqsubseteq_\mathcal{T}$) role names t with $r \sqsubseteq_\mathcal{T} t$ and $s \sqsubseteq_\mathcal{T} t$. The following lemma shows that this optimization is correct.

Lemma 9. *The results of the* k-lcs-r *and* k-lcs-r-o *procedures are equivalent.*

Proof. Let $\mathcal{T}$ be a normalized TBox, (V, E, S) be its completion graph, A and B be two concepts names, and k be a natural number. Let further $L = \text{k-lcs-r}(A, B, (V, E, S), k)$ and $L_o = \text{k-lcs-r-o}(A, B, (V, E, S), k)$. We have to show that $L \equiv_\mathcal{T} L_o$. First notice that when the first optimization is applicable, then $A \in S(B)$ or $B \in S(A)$ and hence A resp. B is equivalent to the least common subsumer. Otherwise we will show that $L \equiv_\mathcal{T} L_o$ by induction on the role-depth bound k.

For $k = 0$ both procedures return the same concept description, i.e., $L = L_o$.

For $k > 0$, we will first show that for each role name r and all edges $(A, r, C) \in E$, $(B, r, D) \in E$, we have $L_o \sqsubseteq_\mathcal{T} \exists r.\, \text{k-lcs-r}(C, D, (V, E, S), k-1)$. If $(A, r, C) \in E$, then there exists $(s_1, C') \in SA$ with $s_1 \sqsubseteq_\mathcal{T} r$ and $C \in S(C')$. Similarly, there exists $(s_2, D') \in SB$ with $s_2 \sqsubseteq_\mathcal{T} r$ and $D \in S(D')$. Then $\exists t.\, \text{k-lcs-r-o}(C', D', (V, E, S), k-1)$ is a conjunct of L_o for all minimal $t \in N_{R,\mathcal{T}}$ with $s_1 \sqsubseteq_\mathcal{T} t$ and $s_2 \sqsubseteq_\mathcal{T} t$. Since $s_1 \sqsubseteq_\mathcal{T} r$ and $s_2 \sqsubseteq_\mathcal{T} r$, there is at least one minimal $t_0 \in N_{R,\mathcal{T}}$ with $t_0 \sqsubseteq r$ for which $\exists t_0.\, \text{k-lcs-r-o}(C', D', (V, E, S), k-1)$ is conjunct of L_o.

Together with $\text{k-lcs-r-o}(C', D', (V, E, S), k-1) \equiv_\mathcal{T} \text{k-lcs-r}(C', D', (V, E, S), k-1)$ by the induction hypothesis and $\text{k-lcs-r}(C', D', (V, E, S), k-1) \sqsubseteq_\mathcal{T} \text{k-lcs-r}(C, D, (V, E, S), k-1)$ for $C' \sqsubseteq_\mathcal{T} C$ and $D' \sqsubseteq_\mathcal{T} D$ we have $\exists t_0.\, \text{k-lcs-r-o}(C', D', (V, E, S), k-1) \sqsubseteq_\mathcal{T} \exists r.\, \text{k-lcs-r}(C, D, (V, E, S), k-1)$.

Since $L_o \sqsubseteq_\mathcal{T} \exists r.\, \text{k-lcs-r}(C, D, (V, E, S), k-1)$ for each role name r and all edges $(A, r, C) \in E$, $(B, r, D) \in E$ and also $L_o \sqsubseteq_\mathcal{T} C$ for $C \in$ common-names, we have $L_o \sqsubseteq_\mathcal{T} L$. On the other hand k-lcs-r computes all of the recursive concept descriptions (and possibly more) that k-lcs-r-o computes and hence $L \sqsubseteq_\mathcal{T} L_o$. This yields that $L \equiv_\mathcal{T} L_o$. □

8 Implementation

We implemented the role-depth bounded least common subsumer for $\mathcal{EL}^+$ in Gel. This program internally uses the reasoner jCel[3], which implements the completion-based classification algorithms for $\mathcal{EL}^+$ and $\mathcal{ELHIf}_{\mathcal{R}^+}$. The processing unit of jCel applies the completion rule step by step. After all completion rules are applied, it starts a post-processing phase, which constructs the concept hierarchy from the computed completion sets. jCel also features a plug-in for the ontology editor Protégé[4]. Protégé

[3] `http://jcel.sourceforge.net`
[4] `http://protege.stanford.edu/`

uses the OWL API[5] to store ontologies, so jCel has to translate the axioms of the ontology into an internal representation, on which the processing unit can work.

GEL expects the input to the k-lcs algorithm and the ontology in OWL API format. According to Algorithm 5, it first adds new concept names and axioms for the input concept descriptions to the ontology and then uses jCel to translate the extended ontology to the internal format. After that GEL invokes a custom processing class, which extends the jCel processor: right before the post-processing phase, after all completion sets are computed, the GEL-processor applies the recursive k-lcs-r-o procedure. The resulting least common subsumer is then simplified, translated back to OWL API format and returned. Although Algorithm 5 only allows for two input concepts, the implementation can compute the role-depth bounded lcs for arbitrary many concepts. This is done by applying the binary k-lcs to the first two concepts and then successively computing the k-lcs of the result with the next concept. In previous tests we found this to be faster then computing the n-ary k-lcs directly.

The implementation includes a Protégé plug-in. This plug-in contains an Protégé ontology view, i.e., a component where the user can input concept descriptions and select processing options such as the role-depth bound and whether to apply the simplification and optimizations. The plug-in then computes the role-depth bounded least common subsumer and shows the user the resulting concept description. Figure 1 shows a screenshot of the GEL plug-in.

GEL was implemented in Java using the Eclipse IDE. Besides jCel and the OWL API, it also uses the Protégé API for the concept description editor and for proper integration into Protégé. It should run on any system with a Java interpreter and Protégé installed.

8.1 Testing

We performed extensive testing, both with real-world ontologies and handcrafted test-ontologies. The real-world ontologies used were the Gene Ontology and a simplified variant of GALEN called NOT-GALEN. For the role-depth bounded least common subsumer, we computed the k-lcs for various input concept descriptions and role-depth bounds both with and without optimizations, and tested if the results indeed subsumed all inputs. For the large life science ontologies, we could not verify that the result was indeed the least w.r.t. subsumption; however, for the handcrafted ontologies, this was checked.

8.2 Evaluation

The role-depth bounded least common subsumer can have a size that is exponential in the role-depth bound k. However, it largely depends on the ontology if such a worst-case behavior occurs. For the Gene Ontology, the role-depth bounded least common subsumer was always constructed and simplified almost instantly. The runtime was totally dominated by the classification time for jCel.

[5] `http://owlapi.sourceforge.net`

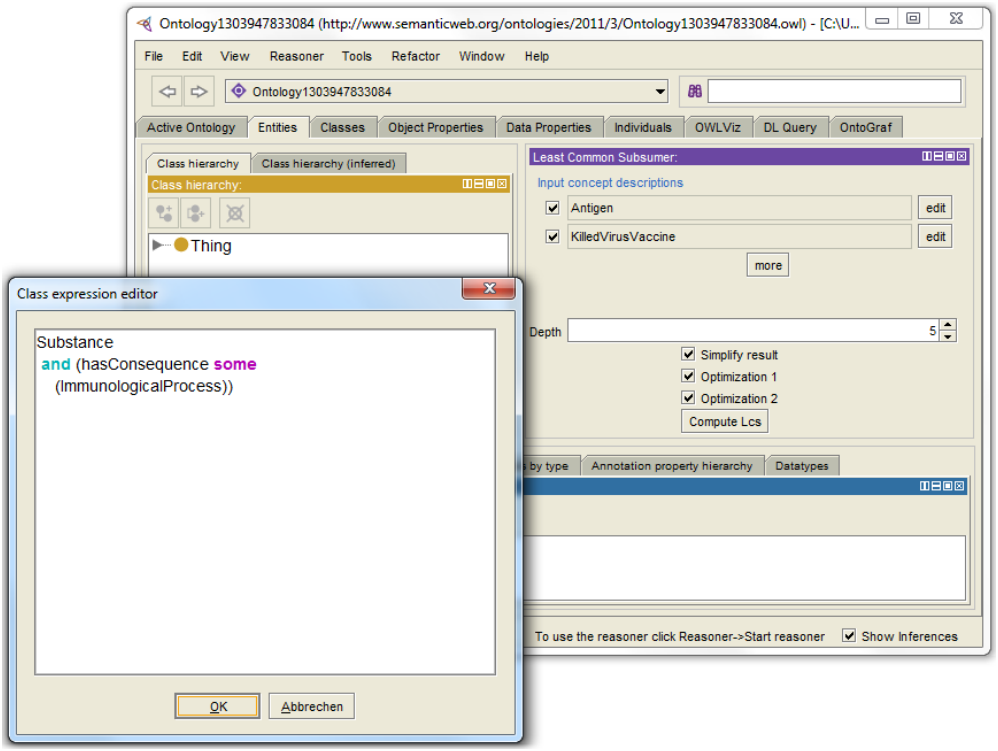


Figure 1: Screenshot of the Protégé plug-in

For NotGalen however, this was not the case. Some input concept pairs resulted in large runtimes of the program, which were mostly dominated by the construction of the result, i.e., the runtime of the k-lcs-r-procedure. Simplification of larger concepts on the other hand was faster by a factor of 10 or more. Normalization and completion of the Not-Galen ontology took around 330ms. Figure 2 shows the average k-lcs-r-construction runtime of Gel on various input pairs for different values of k.

Figure 2 also shows the effect of the optimizations on the construction time. The first optimization, which returns on of the input concepts if it subsumes the other input concept, is able to cut off the k-lcs-r-o recursion before the maximum role-depth is reached. This improves the runtime by a factor of around 2 compared to the basic k-lcs-r procedure with no optimizations. The second optimization, which removes redundant successor nodes before attempting the product construction, is able to reduce the branching factor of the recursion. In our tests, it yielded even better runtime improvements then the first optimization, on average by a factor of 30. Both optimizations yielded better speed-ups for higher role-depth bounds. Combining the optimizations yielded the best runtime for most cases, which indicates that both

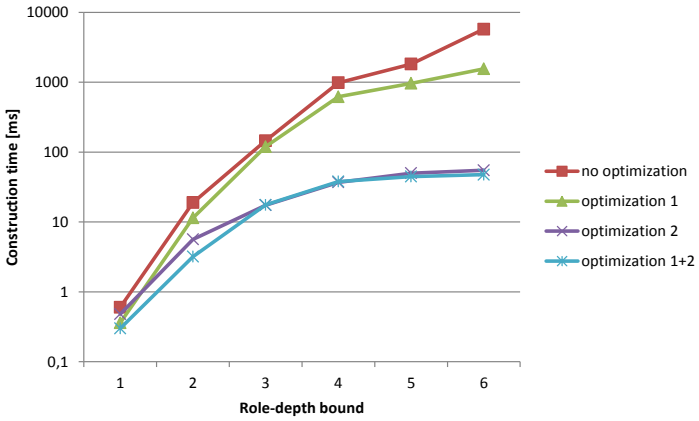


Figure 2: Average k-lcs-r-construction time in NOT-GALEN

optimizations are independent of each other.

In practice, the runtime with only the first optimization (or no optimization at all) was sometimes to large to be useful. For some input concepts, like *PepticUlcer* and *AreaOfAtrophicGastritis*, it didn't return a result within a reasonable time of one hour for a role-depth bound for as low as 4. However with both optimizations enabled, the result for a role-depth bound of 4 was computed in 1.9 seconds, which is still reasonable for a classification time of 330ms. For the given case, the role-depth bound can increased to values beyond 50, where the computation with both optimizations enabled need around a minute to complete. The reason that the second optimization is so effective on NOT-GALEN is that this ontology contains a deep role hierarchy, where without the optimization for each role also all subsuming roles would generate a recursive k-lcs-r call, which yields really large branching factors.

The runtime for the simplification is proportional to the size of the concept description before simplification (which is roughly proportional to the runtime of the k-lcs-r-construction, with a little bit of overhead for the second optimization). However Figure 3 shows that simplification reduces the concept size significantly – even with both optimizations enabled the size of the result is still reduced by a factor 16 on average. Of course, since both optimizations apply simplification steps during the construction of the k-lcs, the size-reduction for results computed without optimizations are much larger.

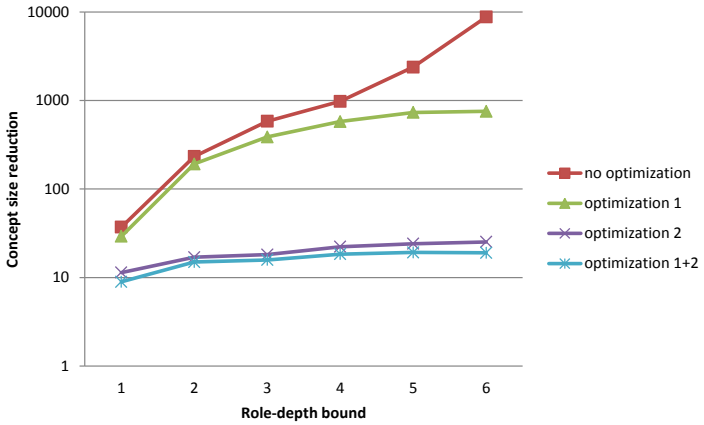


Figure 3: Average size-reduction by simplification in NOT-GALEN

9 Conclusion and Future Work

In this paper we presented the role-depth bounded least common subsumer algorithm for $\mathcal{ELI}$ and $\mathcal{EL}^+$, a simplification procedure and two optimizations for $\mathcal{EL}^+$, and discussed their implementation on the basis of existing completion-based classifiers. GEL is presented as an implementation working on jCel. This implementation is evaluated for real-life ontologies, notably NOT-GALEN.

In future work, the algorithm can to be extended to work with further extensions of $\mathcal{EL}$, especially $\mathcal{ELHI}f_{\mathcal{R}^+}$ to be useful for the full GALEN ontology, or even $\mathcal{ALE}$ with value restrictions. The algorithms can also be optimized further. One idea for optimization is incremental classification of the ontology. If one wants to compute several generalizations without changing the ontology, then the ontology is classified everytime in the current implementation, also only the definitions for the input concepts change during the computations.

Right now, when computing the k-lcs of more then two concepts, the concepts are generalized with the binary k-lcs from left to right. It might be possible to find better (or even an optimal)application order for binary lcs that yields faster computation times.

Many applications of the lcs like the bottom-up construction of knowledge bases also need a msc implementation. In [10] a completion-based algorithm for the role-depth bound msc in $\mathcal{EL}$ is introduced that is very similar to the k-lcs algorithm and that also works for $\mathcal{EL}^+$. This algorithm is also implemented in GEL, but its extension to $\mathcal{ELI}$ and further is still an open problem.

References

[1] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.

[2] F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proceedings of the Methods for Modalities Workshop (M4M-05)*, Berlin, Germany, 2005.

[3] Franz Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 319–324. Morgan Kaufman, 2003.

[4] S. Brandt. Reasoning in $\mathcal{ELH}$ w.r.t. general concept inclusion axioms. LTCS-Report LTCS-04-03, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2004. See http://lat.inf.tu-dresden.de/research/reports.html.

[5] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. López de Mantáras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004)*, pages 298–302. IOS Press, 2004.

[6] Sebastian Brandt and Anni-Yasmin Turhan. Using non-standard inferences in description logics—what does it buy me? In *In: KI Workshop on Applications of Description Logics*, 2001.

[7] Ralf Küsters Franz Baader and Ralf Molitor. Computing least common subsumer in description logics with existential restrictions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99*, pages 96–101. Morgan Kaufmann, 1998.

[8] Rafael Peñaloza and Anni-Yasmin Turhan. Completion-based computation of least common subsumers with limited role-depth for $\mathcal{EL}$ and prob-$\mathcal{EL}^{01}$. LTCS-Report LTCS-10-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2010. See http://lat.inf.tu-dresden.de/research/reports.html.

[9] Rafael Peñaloza and Anni-Yasmin Turhan. Role-depth bounded least common subsumers by completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes. In V. Haarslev, D. Toman, and G. Weddell, editors, *Proc. of the 2010 Description Logic Workshop (DL'10)*, volume 573 of *CEUR-WS*, 2010.

[10] Rafael Peñaloza and Anni-Yasmin Turhan. Towards approximative most specific concepts by completion for el with subjective probabilities. In *Proceedings of the First International Workshop on Uncertainty in Description Logics (UniDL'10)*, volume 613 of *CEUR-WS*, 2010.

[11] Q. H. Vu. Subsumption in the description logic $\mathcal{ELHI}f_{\nabla^+}$ w.r.t. general tboxes. Master's thesis, Technische Universität Dresden, 2008.