Implementing completion-based inferences for the $\mathcal{EL}$-family

Julian Mendez and Andreas Ecke and Anni-Yasmin Turhan

TU Dresden, Institute for Theoretical Computer Science

Abstract. Completion algorithms for subsumption are investigated for many extensions of the description logic $\mathcal{EL}$. While for several of them subsumption is tractable, this is no longer the case, if inverse roles are admitted. In this paper we present an optimized version of the completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ [11], which is implemented in jCel. The completion sets computed during classification are a good substrate for implementing other reasoning services such as generalizations. We report on an extension of jCel that computes role-depth bounded least common subsumers and most specific concepts based on completion sets.

1 Introduction

The lightweight Description Logic (DL) $\mathcal{EL}$ and many of its extensions enjoy the nice property that computing concept subsumption and classification of ontologies written in these Description Logics is tractable [1]. Prominent bio-medical ontologies are expressed in extensions of $\mathcal{EL}$ for which reasoning can still be done in polynomial time. The Gene ontology (GO) is an $\mathcal{ELH}$ ontology and SNOMED is written in $\mathcal{EL}^+$. 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 ontologies ExpTime-complete [2]. While the polynomial time completion algorithms work on graph structures that are static and have simple labellings, the algorithm for $\mathcal{ELI}$ requires dynamic nodes sets and uses complex labels. In [11] a completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ has been devised. Since the node set generated by this method can grow exponentially, it is important to use a good completion strategy, that determines the next node label to which a completion rules is applicable. We present in this paper an optimized version of the algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ with such a completion strategy, which is implemented in the reasoner jCel.

Recently, the completion sets computed during classification have been employed to compute (approximations for) generalization inferences such as the least common subsumer (lcs) or most specific concept (msc). The lcs generalizes a collection of concept descriptions into a single concept description that is the least w.r.t. subsumption. The msc generalizes a description of an individual into a concept description. Intuitively, the msc delivers the most specific concept description that the input individual belongs to. Both of these services are useful for the building of knowledge bases. In the bio-medical field in particular the lcs is employed to define similarity measures between concept descriptions. Since for

	Syntax	Semantics
conjunction	$C \sqcap C$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restr.	$\exists r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
role inclusion	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
functional role	$f(r)$	$\forall d_1 \in \Delta_{\mathcal{I}} : \mid \{d_2 \in \Delta_{\mathcal{I}} \mid (d_1, d_2) \in r^{\mathcal{I}}\} \mid \leq 1$
inverse role	r^-	$\{(d_1, d_2) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d_2, d_1) \in r^{\mathcal{I}}\}$
transitive role	$r \circ r \sqsubseteq r$	$\{(d_1, d_2), (d_2, d_3)\} \subseteq r^{\mathcal{I}} \rightarrow (d_1, d_3) \subseteq r^{\mathcal{I}}$

Table 1. $\mathcal{ELHI}f_{\mathcal{R}^+}$-concept and role constructors.

general $\mathcal{EL}$-TBoxes neither the lcs nor the msc need to exist, an algorithm for role-depth bounded lcs and -msc was devised in [8]. These algorithms are now implemented for $\mathcal{ELH}$ on top of jCel.

2 Preliminaries

Starting from two disjoint sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept* and *role names*, respectively, $\mathcal{ELHI}f_{\mathcal{R}^+}$*-concept descriptions* are built using concept and role constructors shown in Table 1 and the *top-concept* ($\top$). The DL $\mathcal{EL}$ is the $\mathcal{ELHI}f_{\mathcal{R}^+}$-fragment that only allows for the concept constructors conjunction and existential restrictions. $\mathcal{ELH}$ extends $\mathcal{EL}$ by role inclusion statements.

The semantics of $\mathcal{ELHI}f_{\mathcal{R}^+}$ is defined by interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names and subsets of $\Delta^{\mathcal{I}}$ to concepts. The interpretation function is extended to complex concept descriptions and roles as described in the last column of Table 1.

A TBox is a set of *concept inclusion axioms* of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation $\mathcal{I}$ *satisfies* the concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$. A concept C is *subsumed by* a concept D w.r.t. $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$) if, for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$.

Let $\mathsf{N_I}$ be a set of individual names. An $\mathcal{EL}$*-ABox* is a set of assertions of the form $C(a), r(a,b)$, where C is an $\mathcal{EL}$-concept description, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. A *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

Finally, an individual $a \in \mathsf{N_I}$ is an *instance* of a concept description C w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models C(a)$) if $\mathcal{I} \models C(a)$ for all models $\mathcal{I}$ of $\mathcal{K}$. *ABox realization* is to compute for each individual a in $\mathcal{A}$ the set of named concepts from $\mathcal{K}$ that have a as an instance.

3 Completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$

Classification of TBoxes is the computation of all subsumption relations between all named concepts of a TBox. For several extensions of $\mathcal{EL}$ classification can be performed in polynomial time [1, 2]. These classification algorithms typically proceed in three steps:

NR-1	$C \equiv D$	$\rightsquigarrow$	$C \sqsubseteq D, D \sqsubseteq C$
NR-2	$C_1 \sqcap \cdots \sqcap \hat{C} \sqcap \cdots \sqcap C_n \sqsubseteq D$	$\rightsquigarrow$	$\hat{C} \sqsubseteq A, C_1 \sqcap \cdots \sqcap A \sqcap \cdots \sqcap C_n \sqsubseteq D$
NR-3	$\exists r'.\hat{C} \sqsubseteq D$	$\rightsquigarrow$	$\hat{C} \sqsubseteq A, \exists r'.A \sqsubseteq D$
NR-4	$\hat{C} \sqsubseteq \exists r'.D$	$\rightsquigarrow$	$\hat{C} \sqsubseteq A, A \sqsubseteq \exists r'.D$
NR-5	$B \sqsubseteq \exists r'.\hat{C}$	$\rightsquigarrow$	$B \sqsubseteq \exists r'.A, A \sqsubseteq \hat{C}$
NR-6	$D \sqsubseteq C_1 \sqcap C_2$	$\rightsquigarrow$	$D \sqsubseteq C_1, D \sqsubseteq C_2$
NR-7	$C \sqsubseteq \exists r^-.D$	$\rightsquigarrow$	$C \sqsubseteq \exists u.D$, $u \sqsubseteq r^-$, $r^- \sqsubseteq u$
NR-8	$\exists r^-.C \sqsubseteq D$	$\rightsquigarrow$	$\exists u.C \sqsubseteq D$, $u \sqsubseteq r^-$, $r^- \sqsubseteq u$

where r: role; r': (inverse) role; C, C_i, D: concept descriptions; $\hat{C}$, $\hat{D}$: complex concept descriptions; B: concept name; A: *fresh* concept name; u: *fresh* role name.

Table 2. Normalization rules.

1. normalization of the TBox
2. apply completion rules to the *completion graph*
3. read off subsumptions relations from the saturated completion graph

The basic completion algorithm represents the completion graph by two kinds of *completion sets*: $S(C)$ and $S(C, r)$ for each concept name C and role name r from the TBox. The sets contain concept names from the TBox and $\top$. The sets $S(C)$ represent the labelled nodes, while the sets $S(C, r)$ represent the edges of the completion graph. The idea of the classification algorithm is that completion rules make implicit subsumption relationships explicit. In fact, the following invariants hold:

– $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
– $D \in S(C, r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.

For extensions of $\mathcal{EL}$ that also offer inverse roles, testing subsumption is not polynomial, but it is ExpTime-complete [2]. In [11] Vu has devised a completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ (and some of its sublanguages). In contrast to the basic completion algorithm, this one works on completion graphs with more complex nodes. Moreover, the set of nodes grows *dynamically* during completion. We describe now an optimized version of Vu's algorithm given in [7].

Normalization. An $\mathcal{ELHI}f_{\mathcal{R}^+}$-TBox $\mathcal{T}$ is in *normal form* if all concept inclusions have one of the following forms, where A_1, A_2, B are concept names:

$$A_1 \sqsubseteq B, \quad A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B, \quad A_1 \sqsubseteq \exists r.A_2 \quad \text{or} \quad \exists r.A_1 \sqsubseteq B.$$

Each $\mathcal{ELHI}f_{\mathcal{R}^+}$-TBox can be transformed into this normal form by applying the rules shown Table 2, where the axioms on the left-hand side are replaced by the axiom(s) on the right-hand side. The implicit information on (functional) roles is made explicit by applying the following *saturation rules* to the TBox:

$$\begin{array}{llll} r \sqsubseteq s & \rightsquigarrow \; r^- \sqsubseteq s^- & r \sqsubseteq s \;,\; s \sqsubseteq t & \rightsquigarrow \; r \sqsubseteq t \\ r \circ r \sqsubseteq r & \rightsquigarrow \; r^- \circ r^- \sqsubseteq r^- & r \sqsubseteq s \;,\; f(s) & \rightsquigarrow \; f(r) \end{array}$$

In addition, auxiliary role names are added to $\mathsf{N_R}$ to allow a mapping where each role s has an inverse role r^- such that $s \equiv r^-$. In this way, the algorithm applies the completion rules to role names and inverse roles.

Completion rules. Once the TBox is normalized and saturated, the completion sets are initialized and the completion rules are applied. Based on the two sets $\Xi := \{\exists r.A \mid r \in \mathsf{N_R}, A \in \mathsf{N_C}\}$ and $\Omega := \{(A, \psi) \mid A \in \mathsf{N_C}, \psi \subseteq \Xi\}$ the completion sets are defined as

– $V \subseteq \Omega$
– $S \subseteq \{(x, A) \mid x \in \Omega, A \in \mathsf{N_C}\}$
– $R \subseteq \{(r, x, y) \mid r \in \mathsf{N_R}, x, y \in \Omega\}$.

For the completion graph, the set V is the set of nodes, S is a node labeling and Ω is the set of edges. The elements in S are called *S-entries*, the elements in R are called *R-entries*, the elements in V are referred to as *nodes*. The completion process satisfies the following *invariants*:

– if $((A, \varphi), C) \in S$, then $(A \sqcap \bigsqcap_{E \in \varphi} E) \sqsubseteq_{\mathcal{T}} C$
– if $(r, (A, \varphi), (B, \psi)) \in R$, then $(A \sqcap \bigsqcap_{E \in \varphi} E) \sqsubseteq_{\mathcal{T}} \exists r.(B \sqcap \bigsqcap_{E \in \psi} E)$

where each E is of the form $\exists r.X$. Furthermore, after completion we have that $A \sqsubseteq_{\mathcal{T}} B$ if and only if $((A, \emptyset), B) \in S$. The algorithm initializes the sets as follows:

– $V := \{(A, \emptyset) \mid A \in \mathsf{N_C}\}$,
– $S := \{((A, \emptyset), A) \mid A \in \mathsf{N_C}\} \cup \{((A, \emptyset), \top) \mid A \in \mathsf{N_C}\}$,
– $R := \emptyset$

and applies the completion rules. The optimized completion rules for $\mathcal{ELHI}f_{\mathcal{R}^+}$ are presented in Table 3. The underlined elements are membership checks for S and R. These conditions are relevant for the strategy of completion.

As a consequence of the normal form presented here, CR-2 may have several conjuncts on the left-hand side of a normalized GCI. This simple optimization reduces the number of auxiliary symbols.

In [7] it was shown that the rules in Table 3 are equivalent to those in [11].

Completion strategy. The completion rules do not define any order of application to the S- and R entries. In fact, finding the element of the completion sets and axioms from the TBox to which a rule is applicable fast is crucial for the performance of the reasoner. The idea of *sets* is that it collects newly generated entries, which are not present in the set yet, to be tested for applicability of completion rules. This approach has already been employed in Cel, see [**?**]. In Cel each node has an associated queue with entries to be tested. This idea is now transferred to dynamic node sets and the set of completion rules for $\mathcal{ELHI}f_{\mathcal{R}^+}$.

To prepare Q the initialization of the algorithm is slightly modified:

```
CR-1 if $A \sqsubseteq B \in \mathcal{T}$, $\underline{(x,A) \in S}$ then $S' := S \cup \{(x,B)\}$

CR-2 if $A_1 \sqcap \ldots \sqcap A_i \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{T}$,
       $(x,A_1) \in S$, ..., $\underline{(x,A_i) \in S}$, ..., $(x,A_n) \in S$
     then $S' := S \cup \{(x,B)\}$

CR-3 if $A \sqsubseteq \exists r.B \in \mathcal{T}$, $\underline{(x,A) \in S}$
     then if $f(r)$
       then $v := (\top, \{\exists r^-.A\})$
         if $v \notin V$ then $V := V \cup \{v\}$,  $S' := S \cup \{(v,B)\} \cup \{(v,\top)\}$,
                              $R' := R \cup \{(r,x,v)\}$
       else $y := (B, \emptyset)$
         $R' := R \cup \{(r,x,y)\}$

CR-4 if $\exists s.A \sqsubseteq B \in \mathcal{T}$, $\underline{(r,x,y) \in R}$, $\underline{(y,A) \in S}$, $r \sqsubseteq_{\mathcal{T}} s$
     then $S' := S \cup \{(x,B)\}$

CR-5 if $s \circ s \sqsubseteq s \in \mathcal{T}$, $\underline{(r_1,x,y) \in R}$, $\underline{(r_2,y,z) \in R}$, $r_1 \sqsubseteq_{\mathcal{T}} s$, $r_2 \sqsubseteq_{\mathcal{T}} s$
     then $R' := R \cup \{(s,x,z)\}$

CR-6 if $\exists s^-.A \sqsubseteq B \in \mathcal{T}$, $r \sqsubseteq_{\mathcal{T}} s$, $\underline{(r,x,y) \in R}$, $\underline{(x,A) \in S}$, $(y,B) \notin S$, $y = (B',\psi)$
     then $v := (B', \psi \cup \{\exists r^-.A\})$
       if $v \notin V$ then $V := V \cup \{v\}$,  $S' := S \cup \{(v,k) \mid (y,k) \in S\}$
       $S' := S \cup \{(v,B)\}$,  $R' := R \cup \{(r,x,v)\}$

CR-7 if $\exists s^-.A \sqsubseteq B \in \mathcal{T}$, $\underline{(r_2,x,y) \in R}$, $x = (A',\varphi)$, $y = (B',\psi)$,
       $r \circ r \sqsubseteq r \in \mathcal{T}$, $r_1 \sqsubseteq_{\mathcal{T}} r$, $r_2 \sqsubseteq_{\mathcal{T}} r$, $\exists r_1^-.A \in \varphi$, $r \sqsubseteq_{\mathcal{T}} s$
     then $v := (B', \psi \cup \{\exists r^-.A\})$
       if $v \notin V$ then $V := V \cup \{v\}$,  $S' := S \cup \{(v,k) \mid (y,k) \in S\}$
       $S' := S \cup \{(v,B)\}$,  $R' := R \cup \{(r_2,x,v)\}$

CR-8 if $A \sqsubseteq \exists r_2^-.B \in \mathcal{T}$, $\underline{(r_1,x,y) \in R}$, $\underline{(y,A) \in S}$,  $r_1 \sqsubseteq_{\mathcal{T}} s$, $r_2 \sqsubseteq_{\mathcal{T}} s$, $f(s^-)$
     then $S' := S \cup \{(x,B)\}$

CR-9 if $\underline{(r_1,x,y) \in R}$, $(r_2,x,z) \in R$, $r_1 \sqsubseteq_{\mathcal{T}} s$, $r_2 \sqsubseteq_{\mathcal{T}} s$,
       $\underline{y = (\top,\psi)}$, $z = (\top,\varphi)$, $y \neq z$, $f(s)$
     then $v := (\top, \psi \cup \varphi)$
       if $v \notin V$ then $V := V \cup \{v\}$
       $S' := S \cup \{(v,k) \mid (y,k) \in S\} \cup \{(v,k) \mid (z,k) \in S\}$,  $R' := R \cup \{(r_1,x,v)\}$
```

Table 3. Optimized completion rules for $\mathcal{ELHI}f_{\mathcal{R}^+}$.

- $S := \emptyset$, $R := \emptyset$
- $Q := \{((A,\emptyset),A) \mid A \in \mathsf{N_C}\} \cup \{((A,\emptyset),\top) \mid A \in \mathsf{N_C}\}$

The sets S' and R' represent the next step of sets S and R, respectively. Their new elements are added to Q' in the algorithm shown in Table 4.

We say a completion rule is *sensitive* to changes in a set, if the precondition of that rule mentions that set. In Table 3 the relevant entries are underlined. For example, CR-1 is sensitive to changes in S only, CR-7 is sensitive to changes

```
1. $S, R, Q := \emptyset$
2. **for each** concept name $A$, add $((A, \emptyset), A)$ and $((A, \emptyset), \top)$ to $Q$
3. **while** $Q \neq \emptyset$
4.     take one element $e$ from $Q$ and remove it from $Q$
5.     **if** $e$ is an $S$-entry
6.         let $Q'$ be the result of applying all the $S$-rules to $e$
7.     **else if** $e$ is an $R$-entry
8.         let $Q'$ be the result of applying all the $R$-rules to $e$
9.     $Q := Q \cup ((Q' \setminus S) \setminus R)$
```
Table 4. General algorithm.

in R only, and CR-4 is sensitive to changes in S and R. According to the kind of entry they are sensitive to, the completion rules are members of the *chain of rules* the process S-entries or R-entries.

A conceptual scheme of the algorithm is presented in Table 4. The processor takes entries from Q, changes sets S and R, and informs the corresponding chain of rules of these changes. This procedure is repeated until Q is empty, i.e. no rules are applicable.

3.1 Implementation in jCel

jCel[1] is implemented in Java. The object-oriented design of the completion algorithm brings a very low coupling, since each rule can be changed separately. Thus jCel can easily be adapted to new sets of completion rules. For implementation-dependant technical details (e.g. data structures) see [7].

Besides classification for $\mathcal{ELHI}f_{\mathcal{R}^+}$-TBoxes, jCel also implements realization of $\mathcal{ELH}$-ABoxes.

3.2 Experiments with jCel

The experiments were run on a computer with two Intel(R) Core(TM)2 Duo E8500 processors running at 3.16 GHz and 4 GB of main memory.

Experiments classifying $\mathcal{ELHI}f_{\mathcal{R}^+}$ ontologies. The full version of GALEN is still one of the most challenging ontologies, since hardly any reasoner can classify it. Two GALEN ontologies were considered: the original version of GALEN (GALEN-A), and the newer version of GALEN (GALEN-B), which were used in [**?**] to test Cel. Table 6 lists their sizes in terms of concepts etc.

For GALEN-A, jCel took 1093 s and the reasoner CB less than 1 s. In case of GALEN-B, the current version of jCel could not finish classification due to lack of memory. CB classified this ontology in 5 s.

[1] The reasoner jCel and its source code is available at http://jcel.sourceforge.net.

ontology	#axioms	#norm. ax.	#concepts	#roles
GALEN-A	8140	12930	2748	413
GALEN-B	61787	95789	23143	950

Table 5. Ontologies using $\mathcal{ELHI}f_{\mathcal{R}^+}$.

ontology	logic	#axioms	#norm. ax.	#concepts	#roles
NCI	$\mathcal{EL}$	74662	47080	27652	70
GO	$\mathcal{EL}_{\mathcal{R}^+}$	49363	28900	20465	1
FMA	$\mathcal{EL}_{\mathcal{R}^+}$	150282	119570	75139	2
SNOMED CT	$\mathcal{ELH}$	962796	1127193	378569	61
NotGalen	$\mathcal{ELH}_{\mathcal{R}^+}$	7540	15089	2748	413
CELGalen	$\mathcal{ELH}_{\mathcal{R}^+}$	60637	102742	23141	950

Table 6. Ontologies using $\mathcal{ELH}_{\mathcal{R}^+}$.

ontology	entries	jCel 0.13.0	Cel Plug-in 0.5.0	quotient
NCI	346887	8.9 s	10.2 s	0.87
GO	154489	4.4 s	3.5 s	1.26
FMA	9576858	149.0 s	2388.0 s	0.06
SNOMED CT	143039451	1108.0 s	705.0 s	1.57
NotGalen	224565	2.9 s	5.2 s	0.56
CELGalen	6836237	52.0 s	134.0 s	0.39

Table 7. Compared times of classification between jCel and Cel.

Experiments in $\mathcal{ELH}_{\mathcal{R}^+}$. In Table 6 we compare the sizes of the different test ontologies to be classified with the polynomial completion algorithm (with static node set).

The execution times of jCel were compared with the Cel system. Cel is one of the fastest reasoners for reasoning in the $\mathcal{EL}$-family of DLs and is known to deliver correct results [6, 4]. The inferred concept hierarchy was identical in classifications of both reasoners. The measured run-times are shown in Table 7.

To sum up, jCel's performance is comparable to state of the art reasoners and, in case of Cel sometimes even better.

4 Completion based generalization

The classification and the realization algorithm of jCel can be employed to compute generalizations. We define these inferences now.

Definition 1. *Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{ELH}$-KB and $C_1, \ldots, C_n$ $\mathcal{ELH}$-concept descriptions and $k \in \mathbb{N}$. Then the $\mathcal{ELH}$-concept description C is the* role-depth bounded $\mathcal{ELH}$-least common subsumer *of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ and role-depth k (written k-lcs$(C_1, \ldots, C_n)$) iff*

1. *role-depth$(C) \leq k$,*
2. *$C_i \sqsubseteq_{\mathcal{T}} C$ for all $1 \leq i \leq n$, and*

3. *for each $\mathcal{ELH}$-concept description D with role-depth$(D) \leq k$ it holds that, $C_i \sqsubseteq_{\mathcal{T}} D$ for all $1 \leq i \leq n$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

Let a be an individual in $\mathcal{A}$ and again $k \in \mathbb{N}$. The $\mathcal{ELH}$-concept description C is the role-depth bounded $\mathcal{ELH}$-most specific concept *of a w.r.t. $\mathcal{K}$ and role-depth k (written k-msc(a)) iff*

1. *role-depth$(C) \leq k$,*
2. *$\mathcal{K} \models C(a)$, and*
3. *for each $\mathcal{ELH}$-concept description D with role-depth$(D) \leq k$ holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

Completion-based subsumption algorithms classify $\mathcal{ELH}$-TBoxes by explicitly deriving all subsumptions relationships between named concept and storing them in completion sets. The latter can be used to compute the *k-lcs* of concept descriptions. A completion-based realization algorithm can be used to compute the *k-msc* of an individual from its completion sets.

The algorithm for computing *k-lcs* and *k-msc* from completion sets is given in [8]. The idea for *k-lcs* algorithm is: the lcs for two $\mathcal{EL}$-concept descriptions (w.r.t. an empty TBox) can be computed as the product of their corresponding description trees [3]. However, with respect to a general TBox, we can construct the *k-lcs* of two $\mathcal{ELH}$-concept descriptions as follows:

1. assign the input concept descriptions new names
2. classify the augmented TBox
3. for the subgraph of the completion graph reachable from the nodes representing the newly introduced names by paths of length $\leq k$: do cross-product construction w.r.t. the node labels and edges.

The proof of the correctness for the *k-lcs*-algorithm for relies on the invariants discussed in Section 3.

If the completion sets for ABox realization are computed, one can compute the *k-msc* of an individual a simply by traversing the subgraph of the completion graph reachable from a by paths of length up to k and conjoining the node labels.

Since the completion sets are containing *all* subsumers of a named concept, the concept descriptions resulting from traversing subgraphs of the completion graph and collecting the node labels are very redundant. For a person editing the resulting concept description this is clearly undesirable. We devise a simplification heuristic that is similar to the (equivalent) minimal rewritings proposed in [3] for $\mathcal{EL}$-concept descriptions. For general TBoxes the Algorithm 1 yields equivalent and smaller, but not necessarily minimal concept descriptions.

Implementation of the generalization inferences in GEL. Our system GEL implements in Java the methods presented here. GEL accesses JCEL's internal data structures directly to compute the *k-lcs* or the *k-msc*. These two reasoning methods and the above described simplification are implemented in GEL in a straight-forward way.

Algorithm 1 Simplification of the resulting concept description.

Procedure simplify (C, S)
Input: C: $\mathcal{EL}$ concept description; S: set of completion sets
Output: simplify(C): a simplified concept description equivalent to C

```
 1: Let $C$ be of the form $A_1 \sqcap \ldots \sqcap A_n \sqcap \exists r_1.D_1 \sqcap \ldots \sqcap \exists r_m.D_m$ with $A_i \in N_C$
 2: $Conj := \{A_i \mid i \in \{1,\ldots,n\}\}$
 3: $ExRes := \{\exists r_j.D_j \mid j \in (1\ldots m)\}$
 4: for all $A_i, E$ with $A_i \in Conj$ and $E \in Conj \cup ExRes$ do
 5:     if $E \neq A_i$ and $E \sqsubseteq_{\mathcal{T}} A_i$ then
 6:         $R := Conj \setminus \{A_i\}$
 7:     end if
 8: end for
 9: for all $\{E, D\} \subseteq ExRes$ do
10:     if $E \neq D$ and $E \sqsubseteq_{\mathcal{T}} D$ then
11:         $R := R \cup (ExRes \setminus \{D\})$
12:     end if
13: end for
14: for all $\exists r_j.D_j \in R$ do
15:     $R := (R \setminus \{\exists r_j.D_j\}) \cup \{\exists r_j.\text{simplify}(D_j, S)\}$
16: end for
17: return $\bigsqcap_{E \in R} E$
```

Our system Gel is available as a plug-in for the ontology editor Protégé and an API for the role-depth bounded lcs and -msc is planned. The former system sonic [10] implemented the lcs and msc as well, but allowed only for acyclic, unfoldable TBoxes.

Evaluation. For the evaluation of the generalization algorithms, we used two different ontologies. The earlier mentioned NotGalen described in Table 6 is a version of the medical ontology Galen stripped-down to $\mathcal{ELH}$. This ontology does not contain individuals, but its deep concept hierarchy makes it a good test ontology for the *k-lcs*. As test concepts for the *k-lcs* we selected sibling concepts from the concept hierarchy with common ancestors other than $\top$ and with many (comparable) existential restrictions. In total, we selected 20 such concept tuples from NotGalen.

We also used the Sweet[2] ontology, the *Semantic Web for Earth and Environmental Terminology* by NASA. This ontology was converted to $\mathcal{ELH}$ by replacing all value restrictions with existential restrictions and dropping all axioms not expressible in $\mathcal{ELH}$. Sweet does contain individuals and a rich relational structure and was used as a test ontology for the *k-msc*. It has 4276 concept names and 2069 individuals. We selected those individuals from Sweet that appear in many role assertions. In total, we selected 18 individuals from Sweet.

All tests were run on an Intel(R) Core(TM) i5-2400 under Oracle Java 6SE 64bit. For each computation of the *k-lcs* or *k-msc* we measured the concept size

[2] http://sweet.jpl.nasa.gov/sweet/

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
construction time (ms)	19	572	3567	7604	289778
simplification time (ms)	< 1	3	15	40	107
expanded concept size	185	3458	15478	33667	119296
simplified concept size	5	15	27	38	42

Table 8. Average concept size and run-time for the *k-lcs* of concepts from NotGalen.

	k	2-ary lcs	3-ary lcs	4-ary lcs	5-ary lcs
construction time (ms)	1	1	7	37	114
	2	23	249	972	3752
	3	284	1954	4465	24427
	4	1801	5253	8355	45374
	5	7008	12412	114452	2665440
expanded concept size	1	178	214	199	217
	2	3608	5974	2171	2313
	3	14198	26997	12859	15300
	4	35656	34958	25793	31634
	5	104768	133089	123924	178831

Table 9. Average concept size and run-time for the *k-lcs* of n concepts from NotGalen.

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
construction time (ms)	< 1	< 1	1	2	3
simplification time (ms)	< 1	< 1	< 1	< 1	1
expanded concept size	100	275	498	918	2261
simplified concept size	8	9	10	10	11

Table 10. Average concept size and run-time for the *k-msc* of individuals from Sweet.

of the resulting concept description and after simplification and the run-time (after classification / realization) for construction of the *k-lcs* or *k-msc* and of its simplification. The Table 8 and 10 show the results for the *k-msc* and *k-lcs*, respectively. The concept construction time and expanded concept size for different numbers of input concepts to the *k-lcs* are shown in Table 9.

For the *k-lcs* the resulting run-times were quite high, whereas classification of NotGalen took only around 670 ms. Computation of the *k-msc* was always quite fast—especially compared to the realization time of 5.7 s for the Sweet ontology.

For both *k-lcs* and *k-msc* we found the expanded concept size (and thus the construction time) to grow exponentially with the role-depth bound k. The concept size of the simplified concepts, however, is growing much slower.

Interestingly, for the *k-msc* the resulting concept was the *exact* most specific concept for most individuals for a role-depth of only 2 or 3 — the resulting concept did not change for higher k. Only 3 of the 18 individuals had a msc with maximum role-depth of 5.

Table 9 shows how the run-time of the *k-lcs* grows drastically with the number of input concepts, whereas the concept size stays more or less constant. This is the case because the product construction in the *k-lcs* is more expensive for higher n. Instead of directly computing the *k-lcs* for n concepts, one can also apply the binary *k-lcs* to the first two concepts and then successively compute the *k-lcs* of the result with the next concept. Surprisingly, the accumulated construction time for this method to yield the 4-lcs of 5 concepts was in average 1043.8 ms—much faster than the direct computation time of around 45 s.

To sum up, the computation of the fully expanded concept description is very time-consuming. This is especially true for the product construction of the *k-lcs*. To apply some of the simplification steps already during the construction of the result should help the generalization algorithms to scale better.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proc. of the OWLED Workshop*, 2008.
3. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, 1999.
4. K. Dentler, R. Cornet, A. ten Teije, and N. de Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web Journal*, pages 1–17, 2011. DOI: 10.3233/SW-2011-0034.
5. Y. Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI-09)* 2009.
6. J. Mendez and B. Suntisrivaraporn. Reintroducing CEL as an OWL 2 EL reasoner. In *Proc. of the 2009 Description Logic Workshop (DL 2009)*, vol. 477 of *CEUR*, 2009.
7. J. Mendez. A classification algorithm for $\mathcal{ELIH}f_{\mathcal{R}+}$. Master's thesis, Technische Universität Dresden, 2011.
8. R. Peñaloza and A.-Y. Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In *Proc. of the 8th European Semantic Web Conf. (ESWC'11)*, LNCS. Springer, 2011.
9. B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies.* PhD thesis, Technische Universität Dresden, 2009.
10. A.-Y. Turhan and C. Kissig. Sonic — Non-standard inferences go OilEd. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-04)*, vol. 3097 of *LNCS*. Springer, 2004.
11. Q. H. Vu. Subsumption in the description logic $\mathcal{ELHIf}R^{+}$ w.r.t. general TBoxes. Master's thesis, Technische Universität Dresden, 2008.