Extensions of Non-standard Inferences to Description Logics with transitive Roles

Sebastian Brandt[1], Anni-Yasmin Turhan[1], and Ralf Küsters[2] *

[1] Institute for Theoretical Computer Science, TU Dresden, Germany
email: *lastname*@tcs.inf.tu-dresden.de
[2] Theoretical Computer Science, CAU Kiel, Germany
email: kuesters@ti.informatik.uni-kiel.de

Abstract. Description Logics (DLs) are a family of knowledge representation formalisms used for terminological reasoning. They have a wide range of applications such as medical knowledge-bases, or the semantic web. Research on DLs has been focused on the development of sound and complete inference algorithms to decide satisfiability and subsumption for increasingly expressive DLs. Non-standard inferences are a group of relatively new inference services which provide reasoning support for the building, maintaining, and deployment of DL knowledge-bases. So far, non-standard inferences are not available for very expressive DLs. In this paper we present first results on non-standard inferences for DLs with transitive roles. As a basis, we give a structural characterization of subsumption for DLs where existential and value restrictions can be imposed on transitive roles. We propose sound and complete algorithms to compute the least common subsumer (lcs).

1 Introduction and Motivation

Description Logics (DLs) are a family of formalisms used to represent terminological knowledge of a given application domain in a structured and well-defined way. The basic notions of DLs are *concept-descriptions* and *roles*, representing unary predicates and binary relations, respectively. Atomic concepts and concept descriptions represent sets of individuals, whereas roles represent binary relations between individuals [5]. The main characteristic of a DL is the set of concept constructors by which complex concept descriptions can be built from atomic concepts and roles. In the present paper, we are concerned with the DL $\mathcal{FLE}^+$ which provides the constructors conjunction ($C \sqcap D$), existential restriction ($\exists r.C$), value restriction ($\forall r.C$), and the top concept ($\top$).

In $\mathcal{FLE}^+$, a role can be defined transitive. In this case it represents the transitive closure of a binary relation. Transitive roles appear naturally in many application domains, such as medicine and process engineering [1]. Consider, for instance, a machine that comprises several components each of which again consists of several devices. A natural way to represent such a machine by means of

* This work has been supported by the Deutsche Forschungsgemeinschaft, DFG Project BA 1122/4-3.

DLs would be to use some *has-part* role to reflect its compositional structure. It would be natural here to implicitly regard every part of a component also as a part of the whole. To this end, a DL with transitive roles is necessary.

Inference problems for DLs are divided into so-called standard and non-standard ones. Well known standard inference problems are satisfiability and subsumption of concept descriptions. These are well investigated for a great range of DLs. For many of them, sound and complete decision procedures could be devised and lower and upper bounds for the computational complexity have been found [12]. Many standard inference algorithms have been successfully extended to cope with transitive roles [14, 13] and are put into practice in state of the art DL Systems.

Prominent non-standard inferences are matching, the least common subsumer (lcs), the most specific concept (msc), and, more recently, approximation. Non-standard inferences resulted from the experience with real-world DL-knowledge bases (KBs), where standard inference algorithms sometimes did not suffice for building and maintaining purposes. For example, the problem of how to structure the application domain by means of concept definitions may not be clear at the beginning of the modeling task. Moreover, the expressive power of the DL under consideration sometimes makes it difficult to come up with a faithful formal definition of the concept originally intended. To alleviate these difficulties it is expedient to employ non-standard inferences [8].

The lcs was first mentioned as an inference problem for DLs in [11]. Given two concept descriptions A and B in a description logic $\mathcal{L}$, the lcs of A and B is defined as the least (w.r.t. subsumption) concept description in $\mathcal{L}$ subsuming A and B. It has been argued in [8] that the lcs facilitates a "bottom-up"-approach to the above mentioned modeling task: a domain expert can select a number of intuitively related concept descriptions already existent in a KB and use the lcs operation to automatically construct a new concept description representing the closest generalization of them.

Matching in DLs was first proposed in [7]. A matching problem (modulo subsumption) consists of a concept description C and a concept *pattern* D, i.e., a concept description with variables. Matching D against C means finding a substitution of variables in D by concept descriptions such that C is subsumed by the instantiated concept pattern D. Among other applications, matching can be employed for queries in KBs: a domain expert unable to specify uniquely the concept he is looking for in a KB can use a concept pattern to retrieve all those concepts in the KB for which a matcher exists. The structural constraints expressible by patterns exceed the capabilities of simple "wildcards" familiar from ordinary search engines [8].

Approximation was first mentioned as a new inference problem in [4]. The upper (lower) approximation of a concept description C_1 from a DL $\mathcal{L}_1$ is defined as the least (greatest) concept description in another DL $\mathcal{L}_2$ which subsumes (is subsumed by) C_1. Approximation can be used to make non-standard inferences accessible to more expressive DLs by transferring a given inference problem to a less expressive DL where at least an approximate solution can be computed.

Table 1. Syntax and semantics of $\mathcal{FLE}^+$-concept descriptions.

Construct name	Syntax	Semantics
top-concept	$\top$	$\Delta_\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
existential restrictions	$\exists r.C$	$\{x \in \Delta_\mathcal{I} \mid \exists y : (x,y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
value restrictions	$\forall r.C$	$\{x \in \Delta_\mathcal{I} \mid \forall y : (x,y) \in r^\mathcal{I} \rightarrow y \in C^\mathcal{I}\}$
transitive roles	r^+	$\bigcup_{1<n} (r^\mathcal{I})^n$

Another application of approximation lies in user-friendly DL-systems offering a simplified frame-based view on KBs defined in a more expressive background DL [6]. Here approximation can be used to compute simple frame-based representations of otherwise overwhelmingly complicated concept descriptions.

In contrast to standard inference problems, comparatively little research exists on non-standard inferences in DLs with transitive roles [2]. If existential restrictions can be expressed in a DL then the inferences matching and approximation are defined by means of the lcs operation. This central role of the lcs for non-standard inferences has lead us to make this inference problem the first to be extended to $\mathcal{FLE}^+$. Experience with other DLs has shown that to find an lcs algorithm is the crucial step towards algorithms for other non-standard inferences such as matching and approximation. For this reason the lcs in $\mathcal{FLE}^+$ may be regarded as the foundation of several other non-standard inferences in $\mathcal{FLE}^+$. After introducing some basic notions and notation, our first step towards the lcs will be a characterization of subsumption for $\mathcal{FLE}^+$-concept descriptions by means of so-called *description graphs*. We shall see that for two $\mathcal{FLE}^+$-concept descriptions A and B, subsumption ($A \sqsubseteq B$) holds if and only if there exists a simulation relation from the description graph of B into the one of A. The lcs inference of A and B is then defined as the graph product of the respective description graphs.

As a result, we shall see that the lcs of a finite set of $\mathcal{FLE}^+$-concept descriptions always exists and is uniquely determined up to equivalence. Moreover, an effective algorithm for the computation of the lcs will be provided.

All technical details and relevant proofs can be found in our technical report [9]. Moreover, the problem of the lcs computation in two sublanguages of $\mathcal{FLE}^+$, namely $\mathcal{FL}_0^+$ and $\mathcal{EL}^+$, is also addressed in detail.

2 Preliminaries

DLs are based on the following sets of names: N_C is the set of concept names, and N_R is the set of non-transitive roles, and N_R^T is the set of transitive roles, where $N_R \cap N_R^T = \emptyset$. Concept descriptions are inductively defined starting from the set of concept names and use the concept constructors shown in Table 1. The DL $\mathcal{FLE}$ offers the top-concept, conjunction, existential, and value restrictions, as displayed in Table 1. In $\mathcal{FLE}^+$, transitive roles can be used in existential and value restrictions.

As usual, the semantics of a concept description is defined in terms of an *interpretation* $\mathcal{I} = (\Delta, \cdot^I)$. The domain Δ of $\mathcal{I}$ is a non-empty set and the interpretation function $\cdot^I$ maps each concept name $A \in N_C$ to a set $A^I \subseteq \Delta$ and each role name $r \in N_R \cup N_R^T$ to a binary relation $r^I \subseteq \Delta \times \Delta$. The extension of $\cdot^I$ to arbitrary concept descriptions is defined inductively, as shown in the second column of Table 1. Note that all concept descriptions in the above mentioned DLs are satisfiable.

One of the most important traditional inference services provided by DL systems is computing the subsumption hierarchy. The concept description C is *subsumed* by the description D ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations $\mathcal{I}$; C and D are *equivalent* ($C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$.

In this paper we focus on the *non-standard inference* of computing the *least common subsumer (lcs).*

Definition 1 (lcs). *Given $\mathcal{L}$-concept descriptions $C_1, \ldots, C_n$, for some description logic $\mathcal{L}$, the $\mathcal{L}$-concept description C is the* least common subsumer *(lcs) of $C_1, \ldots, C_n$ ($C = \mathsf{lcs}(C_1, \ldots, C_n)$ for short) iff (i) $C_i \sqsubseteq C$ for all $1 \leq i \leq n$, and (ii) C is the least concept description with this property, i.e., if C' satisfies $C_i \sqsubseteq C'$ for all $1 \leq i \leq n$, then $C \sqsubseteq C'$.*

The idea behind the lcs inference is to extract the commonalities of the input concepts. The lcs is uniquely determined up to equivalence. Therefore it is justified to speak about "the" lcs instead of "an" lcs.

3 Least common subsumer for $\mathcal{FLE}^+$

The lcs has already been investigated for sub-logics of $\mathcal{FLE}^+$. The work of Baader et al. [4, 3] investigates the computation of the lcs in $\mathcal{FLE}$ and its sublanguages.

As long as a sublanguage of $\mathcal{FLE}$ does not allow for both existential and value restrictions it is comparatively easy to adapt the existing lcs algorithms to transitive roles. In [9] this is done both for $\mathcal{FL}_0$, admitting only conjunction and value restrictions, as well as for $\mathcal{EL}$, where only conjunction and existential restrictions are admitted. For $\mathcal{EL}^+$, it is possible to translate a concept C into an equivalent one in $\mathcal{EL}$. Thus, all the additional restrictions imposed by transitive roles in C are made explicit. This simple approach, however, does not work for $\mathcal{FLE}^+$-concept descriptions, as the following example illustrates.

Example 1. Consider the $\mathcal{FLE}^+$-concept description $C_{\text{ex}} := (\forall r.\exists r.A) \sqcap \exists r.A$, where r is transitive. To explicitly satisfy the (transitive) value restriction, we need to propagate $\forall r.\exists r.A$ to the existential restriction. This yields $(\forall r.\exists r.A) \sqcap \exists r.(A \sqcap \exists r.A \sqcap \forall r.\exists r.A)$ which equals $(\forall r.\exists r.A) \sqcap \exists r.(A \sqcap C_{\text{ex}})$. Obviously, an attempt of exhaustive propagation would not terminate.

Hence, our first aim is to find a finite representation of $\mathcal{FLE}^+$-concept descriptions in which the transitivity of roles is made explicit. Such a representation is introduced by the following section.

3.1 Description Graphs

In this section we will not only introduce description graphs as a syntactic construct but also provide a model-theoretic semantics for them which makes it easier to examine the equivalence between a concept description and a description graphs directly.

Definition 2 (description graph). *Let $\mathcal{G} := (V, E, v_0, \ell_V, \ell_E)$ be a rooted, directed, and connected graph with labeling functions for vertices and edges. The labeling function ℓ_V assigns a set of concept descriptions to every vertex in V and ℓ_E assigns a label of the form Qr to every edge in E, where $Q \in \{\forall, \exists\}$ and $r \in N_R \cup N_R^T$. An edge labeled $\forall r$ is called* forall-edge*, an edge labeled $\exists r$* exists-edge*. If every vertex v in $\mathcal{G}$ has at most one outgoing forall-edge per role then it is called a* description graph*.*

For the sake of simplicity, we use the notation $(v\, Qr\, w) \in E$ to express that (i) $(v, w) \in E$ and (ii) $\ell_E(v, w) = \{Qr\}$. Note that description graphs can be cyclic. Like concept descriptions, description graphs are interpreted w.r.t. a model-theoretic semantics to be introduced next.

Definition 3 (semantics of description graphs). *Let $\mathcal{G} := (V, E, v_0, \ell_V, \ell_E)$ be a description graph and let $\mathcal{I} := (\Delta, \cdot^I)$ be an interpretation. A mapping $\pi\colon V \to 2^{\Delta^\mathcal{I}} \setminus \emptyset$ is called a* model mapping *iff for all $v, w \in V$ it holds that:*

- *$\pi(v) \subseteq C^\mathcal{I}$ for all $C \in \ell(v)$;*
- *if $(v\, \exists r\, w) \in E$ for $r \in N_R$ and $x \in \pi(v)$ then there exists some $y \in \Delta^\mathcal{I}$ with $(x, y) \in r^\mathcal{I}$ and $y \in \pi(w)$;*
- *if $(v\, \exists r\, w) \in E$ for $r \in N_R^T$ and $x \in \pi(v)$ then there exists some $y \in \Delta^\mathcal{I}$ with $(x, y) \in (r^\mathcal{I})^+$ and $y \in \pi(w)$;*
- *if $(v\, \forall r\, w) \in E$ for $r \in N_R$ and $x \in \pi(v)$ then $(x, y) \in r^\mathcal{I}$ implies $y \in \pi(w)$.*

For a given $x \in \Delta^\mathcal{I}$, define $\mathcal{I}, x \models \mathcal{G}$ iff there is a model mapping π with $x \in \pi(v_0)$. The semantics of $\mathcal{G}$ w.r.t. $\mathcal{I}$ is defined as $\mathcal{G}^\mathcal{I} := \{x \in \Delta^\mathcal{I} \mid \mathcal{I}, x \models \mathcal{G}_C\}$.

There is a similarity between the semantics of description graphs and that of concept descriptions as defined in Section 2. A (transitive) $\exists r$-edge $(v\, \exists r\, w)$ like an existential restriction implies a corresponding r-edge (r-path) for all witnesses $x \in \pi(v)$ in the model. Similarly, every $\forall r$-edge $(v\, \forall r\, w)$ imposes restrictions on every witness in the model reachable via an r-edge from some $x \in \pi(v)$.

Regarded as a description graph the syntax tree of every $\mathcal{FLE}$-concept description C is equivalent to C. This, however, is not generally true of $\mathcal{FLE}^+$-concept descriptions. Moreover, there are description graphs for which no equivalent $\mathcal{FLE}^+$-concept description exists. Ultimately, however, we are interested in description graphs guaranteed to represent concept descriptions. To this end, we introduce six conditions to restrict description graphs further, leading to the notion of simple description graphs. As a prerequisite, we need to specify the notion of a simulation relation for description graphs.

Definition 4 (simulation relation). *For $i \in \{1,2\}$, let $\mathcal{G}_i := (V_i, E_i, v_{0i}, \ell_{V_i}, \ell_{E_i})$ be description graphs. Then, $\mathcal{G}_2 \rightleftharpoons \mathcal{G}_1$ iff there exists a relation $R \subseteq V_2 \times V_1$ with:*

1. $(v_{02}, v_{01}) \in R$
2. $\ell_V(v) \cap N_C \subseteq \ell_V(v') \cap N_C$ *for all* $(v, v') \in R$.
3. *If* $(v\, Q r\, w) \in E_2$ *and* $(v, v') \in R$ *then there exists a vertex* $w' \in V_1$ *such that* $(v'\, Q r\, w') \in E_1$ *and* $(w, w') \in R$.

For vertices $v_1 \in V_1$ and $v_2 \in V_2$, denote by $\mathcal{G}_2(v_2) \rightleftharpoons \mathcal{G}_1(v_1)$ the fact that a simulation relation R exists between the subgraph of $\mathcal{G}_2$ reachable from v_2 and the subgraph of $\mathcal{G}_1$ reachable from v_1. In particular, this implies $(v_2, v_1) \in R$.

Definition 5 (simple description graph). *Let $\mathcal{G} := (V, E, v_0, \ell_V, \ell_E)$ be a description graph. $\mathcal{G}$ is a* simple description graph *iff the following properties hold.*

1. *W.r.t. a breadth-first search tree, $\mathcal{G}$ has no forall-forward edges and no cross edges. Every exists-forward edge only connects vertices connected by a path of exists-tree edges w.r.t. one transitive role.*
2. *If $(v_0\, Q_0 r_0\, v_1 \ldots v_{n-1}\, Q_{n-1} r_{n-1}\, v_0)$ is a cycle in E with pairwise distinct vertices then there exists one transitive role r with $r_i = r$ for all i.*
3. *If $(v_0\, Q_0 r\, v_1 \ldots v_{n-1}\, Q_{n-1} r\, v_0)$ is a cycle in E with pairwise distinct vertices and $r \in N_R^T$ then v_0 has a $\forall r$-successor.*
4. *If $\{(u\, \forall r\, v), (u\, \exists r\, w)\} \subseteq E$ then $\mathcal{G}(v) \rightleftharpoons \mathcal{G}(w)$. If $r \in N_R^T$ then there exists a vertex w' such that $(w\, \forall r\, w') \in E$ and $\mathcal{G}(v) \rightleftharpoons \mathcal{G}(w')$.*
5. *If $(u\, \forall r\, v) \in E$ with $r \in N_R^T$ then there exists a vertex v' such that $(v\, \forall r\, v') \in E$ and $\mathcal{G}(v) \rightleftharpoons \mathcal{G}(v')$.*
6. *If $B \in \ell_V(v)$ then $\mathcal{G}_B \rightleftharpoons \mathcal{G}(v)$ for every vertex $v \in V$.*

The idea behind the above definition to is imitate the propagation of existential and value restrictions in the graph structure. For instance, Condition 4 ensures that no subgraph representing an existential restriction may be more general that a corresponding subgraph representing a value restriction. Hence, a value restriction must be propagated over all existential restrictions. Condition 5 similarly ensures that value restrictions over transitive roles are propagated to deeper role levels, as $\forall r.A$ implies $\forall r.(A \sqcap (\forall r.A))$ and so on. Conditions 2 and 3 ensure that cycles cannot occur arbitrarily. By means of Condition 6, complex concept descriptions are already represented in the structure of the description graph. The first condition excludes a number of irregularities which would make the proofs over description graphs more intricate.

The following lemma can be shown for all description graphs.

Lemma 1. *For description graphs $\mathcal{G}$ and $\mathcal{H}$ it holds that $\mathcal{H} \rightleftharpoons \mathcal{G}$ implies $\mathcal{G} \sqsubseteq \mathcal{H}$.*

Having defined syntax and semantics of description graphs in general the next step is to translate $\mathcal{FLE}^+$-concept descriptions into equivalent description graphs.

3.2 Translation of $\mathcal{FLE}^+$-concept descriptions into $\mathcal{FLE}^+$-description graphs

To show that every $\mathcal{FLE}^+$-concept description has a corresponding $\mathcal{FLE}^+$-description graph we devise a suitable translation function. As a technical prerequisite, we require a normal form for $\mathcal{FLE}$-concept descriptions, as introduced in [3]. The purpose of this normal form is merely to flatten conjunctions, to make the top-concept explicit, and to propagate value restrictions over existential restrictions. The problem of implicit information induced by transitive roles remains untouched here.

Definition 6 ($\mathcal{FLE}$ normalization rules). *Let E, F be two $\mathcal{FLE}^+$-concept descriptions and $r \in N_R \cup N_R^T$ a primitive role. The* $\mathcal{FLE}$-normalization rules *are defined as follows*

1)	$\forall r.\top \longrightarrow \top$	3)	$\forall r.E \sqcap \forall r.F \longrightarrow \forall r.(E \sqcap F)$
2)	$E \sqcap \top \longrightarrow E$	4)	$\forall r.E \sqcap \exists r.F \longrightarrow \forall r.E \sqcap \exists r.(E \sqcap F)$
		5)	$E \sqcap (F \sqcap G) \longrightarrow E \sqcap F \sqcap G.$

A concept description is in $\mathcal{FLE}$-normal form *if the $\mathcal{FLE}$-normalization rules have been applied to it exhaustively.*

Each of the above normalization rules preserve equivalence and should be read modulo commutativity of conjunction. It has been shown in [3] that exhaustive application of these rules may produce concept descriptions of size exponential in the size of the original. During the translation of an $\mathcal{FLE}^+$-concept description into an $\mathcal{FLE}^+$-description tree the $\mathcal{FLE}$-normalization rules need to be applied only to the out most role-level of the $\mathcal{FLE}^+$-concept at a time.

The following definition provides the framework of the translation of $\mathcal{FLE}^+$-concept descriptions into description graphs. For a given concept description C we start with an empty description graph $\mathcal{G}$ consisting only of a root vertex v_0 with C in its label. Then we exhaustively apply graph generation rules (defined in detail in Figure 1) producing new vertices and edges. In this process we distinguish three kinds of edges. The set $E^{\mathcal{D}}$ contains the edges of the underlying spanning tree, in E^+ are the forward-edges induced by transitivity, and in $E^{\circlearrowleft}$ are self-loops or edges that connect a vertex with an ancestor vertex in w.r.t. the spanning tree. As soon as no production rules are applicable, all non-atomic concept descriptions are removed from the label sets of $\mathcal{G}$ and the graph is returned.

For the actual definition, a shorthand notation needs to be introduced first. For a set $\{C_1, \ldots, C_n\}$ of $\mathcal{FLE}^+$-concept descriptions, let $\{C_1, \ldots, C_n\}^*$ denote the corresponding set in which (i) the $\mathcal{FLE}^+$ normalization rules defined above have been applied exhaustively on the top most role-level of every C_i and (ii) every C_i is split into its conjuncts. Observe that there is at most one value restriction per role r in $\{C_1, \ldots, C_n\}^*$.

Definition 7 ($\mathcal{FLE}^+$-description graph). *Let C be a $\mathcal{FLE}^+$-concept description. The* $\mathcal{FLE}^+$-description graph $\mathcal{G}_C$ *is obtained by the following procedure:*

$\mathbf{R}_{\exists}$: If $(\exists r.C') \in \ell_V(v)$, $(\forall r.C'') \notin \ell_V(v)$ for some C', C'', and
there is no $v'' \in V : (v, \exists r, v'') \in E^{\mathcal{D}} \cup E^{\circlearrowleft} \ \wedge \ \{C'\}^* = \ell_V(v'')$,
then if there is $v_i \in V : v_i$ appears in $\rho(v) \wedge \ell_V(v_i) = \{C'\}^*$,
then $E^{\circlearrowleft} := E^{\circlearrowleft} \cup \{(v, \exists r, v_i)\}$,
else $V := V \cup \{v'\}$, $E^{\mathcal{D}} := E^{\mathcal{D}} \cup \{(v, \exists r, v')\}$, $\ell_V(v') := \{C'\}^*$.

$\mathbf{R}_{\exists\forall}$: If $r \in N_R$, and $\{(\exists r.C'), (\forall r.C'')\} \subseteq \ell_V(v)$ for some C', C'', and
there is no $v'' \in V : (v, \exists r, v'') \in E^{\mathcal{D}} \cup E^{\circlearrowleft} \ \wedge \ \{C'\}^* = \ell_V(v'')$,
then if there is $v_i \in V : v_i$ appears in $\rho(v) \wedge \ell_V(v_i) = \{C'\}^*$
then $E^{\circlearrowleft} := E^{\circlearrowleft} \cup \{(v, \exists r, v_i)\}$,
else $V := V \cup \{v'\}$, $E^{\mathcal{D}} := E^{\mathcal{D}} \cup \{(v, \exists r, v')\}$, $\ell_V(v') := \{C'\}^*$.

$\mathbf{R}_{\exists\forall^+}$: If $r \in N_R^T$, and $\{(\exists r.C'), (\forall r.C'')\} \subseteq \ell_V(v)$ for some C', C'', and
there is no $v'' \in V : (v, \exists r, v'') \in E^{\mathcal{D}} \cup E^{\circlearrowleft} \wedge \{C', \forall r.C''\} = \ell_V(v'')$,
then if there is $v_i \in V : v_i$ appears in $\rho(v) \wedge \ell_V(v_i) = \{C', \forall r.C''\}^*$
then $E^{\circlearrowleft} := E^{\circlearrowleft} \cup \{(v, \exists r, v_i)\}$,
else $V := V \cup \{v'\}$, $E^{\mathcal{D}} := E^{\mathcal{D}} \cup \{(v, \exists r, v')\}$, $\ell_V(v') := \{C', \forall r.C''\}^*$.

$\mathbf{R}_{\forall}$: If $r \in N_R$, and $(\forall r.C') \in \ell_V(v)$ for some C', and
there is no $v'' \in V : (v, \forall r, v'') \in E^{\mathcal{D}} \cup E^{\circlearrowleft}$
then if there is $v_i \in V : v_i$ appears in $\rho(v) \wedge \ell_V(v_i) = \{C'\}^*$
then $E^{\circlearrowleft} := E^{\circlearrowleft} \cup \{(v, \forall r, v_i)\}$,
else $V := V \cup \{v'\}$, $E^{\mathcal{D}} := E^{\mathcal{D}} \cup \{(v, \forall r, v')\}$, $\ell_V(v') := \{C'\}^*$.

$\mathbf{R}_{\forall^+}$: If $r \in N_R^T$, and $(\forall r.C') \in \ell_V(v)$ for some C', and
there is no $v'' \in V : (v, \forall r, v'') \in E^{\mathcal{D}} \cup E^{\circlearrowleft}$
then if there is $v_i \in V : v_i$ appears in $\rho(v) \wedge \ell_V(v_i) = \{C', \forall r.C'\}^*$
then $E^{\circlearrowleft} := E^{\circlearrowleft} \cup \{(v, \forall r, v_i)\}$,
else $V := V \cup \{v'\}$, $E^{\mathcal{D}} := E^{\mathcal{D}} \cup \{(v, \forall r, v')\}$, $\ell_V(v') := \{C', \forall r.C'\}^*$.

$\mathbf{R}_{E^+}$: If $r \in N_R^T$, and $\{(v, \exists r, v'), (v', \exists r, v'')\} \in E^{\mathcal{D}}$ and $(v, \exists r, v'') \notin E^+$
then $E^+ := E^+ \cup \{(v, \exists r, v'')\}$

Fig. 1. $\mathcal{FLE}^+$-Description Graph Generation Rules.

1. *Initialize the sets* $V := \{v_0\}$, $\ell_V = \ell_V(v_0) = \{C\}^*$, *and* $E := E^+ := E^{\mathcal{D}} := E^{\circlearrowleft} := \emptyset$.
2. *Apply the* $\mathcal{FLE}^+$*-description graph generation rules from Figure 1 exhaustively to obtain* $\mathcal{G}'_C := (V, E, v_0, \ell_V, \ell_E)$, *where* $E = E^{\mathcal{D}} \cup E^{\circlearrowleft} \cup E^+$.
3. *Reduce the label sets of vertices:* $\forall v \in V$: $\ell'_V(v) := \ell_V(v) \cap N_C$.
4. *return* $\mathcal{G}_C := (V, E, v_0, \ell'_V, \ell_E)$.

All non-atomic concept descriptions in the label sets of the vertices of $\mathcal{G}$ are discarded afterwards because their information (as we shall see) is then represented by the structure of the graph. It remains to define the generation rules used in Step 2 of the above definition.

Figure 1 shows the generation rules referred to in Definition 7. For every v, $\rho(v)$ denotes the (unique) path from v_0 to v w.r.t. tree edges. Intuitively, the idea of the rules is to use the concept descriptions occurring in the label set of a vertex v to extend the description graph "accordingly", i.e., if an existential

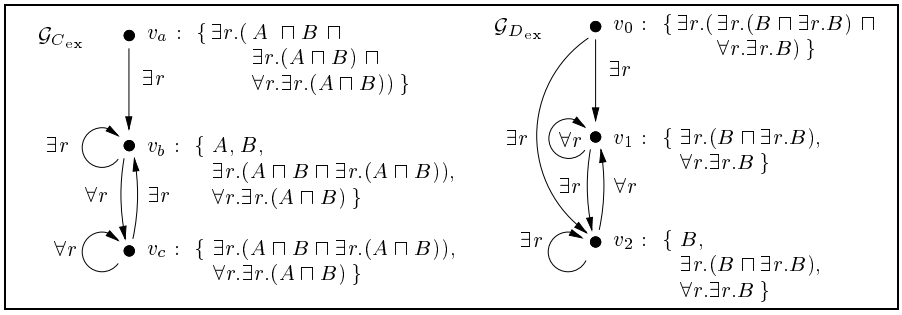

Fig. 2. $\mathcal{FLE}^+$-description graphs

restriction $\exists r.C$ occurs in $\ell_V(v)$ then a vertex w must be introduced (or probably only found) such that (i) w is connected to v by an exists-edge and (ii) a concept equivalent to C occurs in $\ell_V(w)$. Moreover, a value restriction $\forall r.D$ probably also occurring in $\ell_V(v)D$ must be propagated to $\ell(w)$.

Starting at a given vertex v, the rules $\mathbf{R}_\exists$, $\mathbf{R}_{\exists\forall}$, and $\mathbf{R}_{\exists\forall^+}$ all produce new exists-edges, possibly to a newly generated vertex. $\mathbf{R}_\exists$ applies if only an existential restriction is present in $\ell_V(v)$, $\mathbf{R}_{\exists\forall}$ applies if an additional value restriction (w.r.t. the same non-transitive role) is present, and $\mathbf{R}_{\exists\forall^+}$ covers the analogous transitive case. Similarly, $\mathbf{R}_\forall$ and $\mathbf{R}_{\forall^+}$ address the case where only a value restriction (non-transitive or transitive) is present. Rule $\mathbf{R}_{\exists^+}$ never introduces new vertices but only adds forward-edges over exists-paths w.r.t. one transitive role.

To avoid generating infinitely many new vertices, every generation rule has a *blocking condition*[3] testing for every vertex v whether or not a new vertex w can be avoided by a back edge to an already existing ancestor vertex u. This is the case if the ancestor u has the same label set as the new vertex w would get, i.e., $\ell_V(u) = \ell_V(w)$. The vertex u is regarded as ancestor of v iff u lies on a (the) tree-path from the root vertex to v. Note that the condition $\ell_V(u) = \ell_V(w)$ determines u uniquely and that $v = w$ is not excepted. The following example shows the corresponding $\mathcal{FLE}^+$-description graph of two simple $\mathcal{FLE}^+$-concepts.

Example 2. Let $C_{ex} := \exists r.(B \sqcap \exists r.B \sqcap \forall r.\exists r.B)$ and $D_{ex} := \exists r.(\exists r.B \sqcap \forall r.\exists r.B)$ for a transitive role r and an atomic concept B. The corresponding $\mathcal{FLE}^+$-description graphs are depicted in Figure 2. The figure also shows the normalized label sets of every vertex. Note that the non-atomic concept descriptions in the label sets are used only during the generation of the description graphs.

It remains to be shown that the resulting $\mathcal{FLE}^+$-description graphs are in fact equivalent to the original concept descriptions. It is shown in [9] that the following theorem holds.

Lemma 2. *Let C be an $\mathcal{FLE}^+$-concept descriptions, then (1) $C \equiv \mathcal{G}_C$ and (2) $\mathcal{G}_C$ is a simple description graph.*

[3] Blocking strategies originally have been introduced in the DL context in [10] for a tableaux-based satisfiability tester for expressive DLs.

As a result, we know how to encode the information represented by $\mathcal{FLE}^+$-concept descriptions in $\mathcal{FLE}^+$-description graphs. Our next step is to find a way to translate description graphs back to concept descriptions.

3.3 Translation of simple description graphs into $\mathcal{FLE}^+$-concept descriptions

It has already been mentioned in Section 3.1 that description graphs exist without an equivalent $\mathcal{FLE}^+$-concept description, see [9]. We shall see that it suffices to restrict our backward translation procedure to the class of simple description graphs introduced in the previous section.

For the backward translation procedure we may not rely on complex concept descriptions in the label sets of the graphs in question. On the contrary, the idea is to re-build complex concept descriptions in the label sets while preserving equivalence to the original description graph. This process is continued until the desired concept description occurs in the root label. Note that this strategy is just the reverse of the generation procedure of $\mathcal{FLE}^+$-description graphs, where the label of the root vertex generates the entire description graph.

To formalize the notion of re-building complex labels we devise an operation which modifies a given description graph by altering its label function. Intuitively, the function acc "accumulates" complex concept descriptions in the label sets of the vertices.

Definition 8. *Let $\mathcal{G} := (V, E, v_0, \ell_V, \ell_E)$ be a description graph and $|E| := n$. Then,* $\mathrm{acc}(\mathcal{G}) := (V, E, v_0, \ell'_V, \ell_E)$ *where ℓ'_V is defined as follows. For every $v \in V$,*

$$\ell'_V(v) := (\ell_V(v) \cap N_C\,) \cup \bigcup_{r \in N_R \cup N_R^T} \left(\bigcup_{(v\,\exists r\,w) \in E} \exists r. \bigsqcap \ell_V(w) \right.$$

$$\left. \cup \bigcup_{(v\,\forall r\,w) \in E} \left(\forall r. \bigsqcap (\ell_V(w) \setminus \{\forall r.\top\}) \sqcap \bigsqcap_{(w\,\exists r\,w') \in E} \exists r. \bigsqcap \ell_V(w') \right) \right).$$

Define $\mathrm{conc}(\mathcal{G}) := \bigsqcap \ell_V(v'_0)$*, where v'_0 denotes the root vertex of* $\mathrm{acc}^n(\mathcal{G})$.

For every vertex v, the modified label function ℓ'_V contains the same atomic labels as before but additionally has an existential restriction based on the label of every $\exists r$-successor of v. Forall-edges are treated similarly only that existential edges starting from vertices reachable by forall-edges are also taken into account. Observe that $\mathrm{acc}(\mathcal{G})$ is still a simple description graph.

To illustrate the effect of the function acc, consider the a simple description graph $\mathcal{G}$ with only one vertex v_0 with a label $\ell_V(v_0) = \{A\}$ and edges $E := \{(v_0, \exists r, v_0), (v_0\,\forall r\,v_0)\}$. In the graph $\mathrm{acc}(\mathcal{G})$ the root vertex has the label $\{A, \exists r.A, \forall r.(A \sqcap \exists r.A)\}$. Applying acc again we obtain the root label of $\mathrm{acc}^2(\mathcal{G})$ which equals $\{A, \exists r.(A \sqcap \exists r.A \sqcap \forall r.(A \sqcap \exists r.A)), \forall r.(A \sqcap \exists r.A \sqcap \forall r.(A \sqcap \exists r.A) \sqcap \exists r.(A \sqcap \exists r.A \sqcap \forall r.(A \sqcap \exists r.A)))\}$.

It suffices to show that applying the function acc at most $|E|$ times produces a root label such that the conjunction of all contained concepts is equivalent to $\mathcal{G}$. Hence, we obtain the following theorem.

Theorem 1. *For every simple description graph* $\mathcal{G} = (V, E, v_0, \ell_V, \ell_E)$ *it holds that* $\mathrm{conc}(\mathcal{G}) \equiv \mathcal{G}$.

The idea of the proof is to show the equivalence $\mathrm{conc}(\mathcal{G}) \equiv \mathcal{G}$ in three steps. Firstly, we show for every $\mathcal{G}$ that a single application of acc preserves equivalence, i.e., $\mathcal{G} \equiv \mathrm{acc}(\mathcal{G})$. This immediately implies $\mathcal{G} \equiv \mathrm{acc}^{|E|}(\mathcal{G})$. Secondly, due to the semantics of description graphs it is also easy to see that every concept description in the root label of $\mathrm{acc}^{|E|}(\mathcal{G})$ subsumes $\mathrm{acc}^{|E|}(\mathcal{G})$. Hence, $\mathrm{acc}^{|E|}(\mathcal{G}) \sqsubseteq \mathrm{conc}(\mathcal{G})$. Thirdly, we can show that every model of $\mathrm{conc}(\mathcal{G})$ is also a model of $\mathrm{acc}^{|E|}(\mathcal{G})$. See [9] for the full proof.

Now the necessary means are provided to translate $\mathcal{FLE}^+$-concept descriptions (back and forth) into a representation where the transitivity of roles is made explicit. To define the lcs operation w.r.t. description graphs we first need a complete characterization of subsumption in this representation.

3.4 Characterization of subsumption in $\mathcal{FLE}^+$

In this section, the description graphs introduced previously are used to characterize subsumption of $\mathcal{FLE}^+$-concept descriptions.

Theorem 2. *Let* C, D *be* $\mathcal{FLE}^+$*-concept descriptions, then* $C \sqsubseteq D$ *iff* $\mathcal{G}_D \overset{\sim}{\to} \mathcal{G}_C$.

To show the 'if'-direction, one can use a canonical model I_C of C obtained from $\mathcal{G}_C$ by renaming the labels of all edges $(v\, Q r\, w)$ in E_C to $(v\, r\, w)$. The fact that (i) I_C actually is a model of C and (ii) that by subsumption every model of C is also one of D can then be used to construct a simulation relation R. This is done iteratively while traversing $\mathcal{G}_D$ in depth-first order starting from the root vertex. See [9] for the full proof. However, the proof of the 'only if'-direction is easily obtained as a consequence of Lemma 2 and two results shown in the previous sections, namely Lemma 1 and Theorem 1. To illustrate the above result, we return to the example introduced in the previous section.

Example 3. Recall the concepts from Example 2. The only difference between C_{ex} and D_{ex} is the atomic concept B in the outermost existential restriction of C_{ex}. Hence, $C_{ex} \sqsubseteq D_{ex}$. It is easy to see that $R := \{(v_0, v_a), (v_1, v_c), (v_2, v_b)\}$ is in fact a simulation relation from $\mathcal{G}_{D_{ex}}$ into $\mathcal{G}_{C_{ex}}$. For all pairs it holds that the label set of the first vertex is a subset of that of the second one. Moreover, every edge starting from the first vertex can also be traveled from the second one, reaching again a pair in R. Note that this property does not hold without the transitive edge $(v_0\, \exists r\, v_2)$ in $\mathcal{G}_{D_{ex}}$.

3.5 Computation of the lcs in $\mathcal{FLE}^+$

With all the information captured in a $\mathcal{FLE}$-concept description made explicit by $\mathcal{FLE}^+$-description graphs the next step is to extract the commonalities of the description graphs. Similar to other approaches to computing the lcs [1, 4] the graph product is employed to this end. In a description graph $\mathcal{G}$ the *depth* of a vertex v is defined as the distance to the root vertex w.r.t. tree edges.

Definition 9 (Product of $\mathcal{FLE}^+$-description graphs). *The* product $\mathcal{G}_C \times \mathcal{G}_D$ *of two $\mathcal{FLE}^+$-description graphs $\mathcal{G}_A = (V_A, E_A, v_{0A}, \ell_{V_A}, \ell_{E_A})$ for $A \in \{C, D\}$ is defined by induction on the depth of the $\mathcal{FLE}^+$-description graphs. The vertex (v_{0C}, v_{0D}) labeled with $\ell_{V_C}(v_{0C}) \cap \ell_{V_D}(v_{0D})$ is the root vertex of $\mathcal{G}_C \times \mathcal{G}_D$. For each pair $(v_C, v_D), v_C \in V_C, v_D \in V_D$ s.t. v_C is a Qr-successor of v_{0C} in $\mathcal{G}_C$ and for v_D is a Qr-successor of v_{0D} in $\mathcal{G}_D$, we obtain a Qr-successor (v_C, v_D) of (v_{0C}, v_{0D}) in $\mathcal{G}_C \times \mathcal{G}_D$. The vertex (v_C, v_D) is the root vertex of the inductively defined product of $\mathcal{G}_C \times \mathcal{G}_D$. The graph $\mathcal{H} = \mathcal{G}_C \times \mathcal{G}_D$ is called the* product graph.

The resulting product graph $\mathcal{G}_C \times \mathcal{G}_D$ is rooted, connected, and directed. Since all vertices in $\mathcal{G}_C$ and $\mathcal{G}_D$ have at most one outgoing all-edge, every vertex in the product graph has at most one outgoing all-edge. Thus, product graphs are description graphs. Note that by construction of the product graph there trivially exists a simulation $Z\colon \mathcal{G}_C \times \mathcal{G}_D \rightleftharpoons \mathcal{G}_C$ and between $Z'\colon \mathcal{G}_C \times \mathcal{G}_D \rightleftharpoons \mathcal{G}_D$.

Example 4. Let us return to the concept descriptions C_{ex} and D_{ex} from Example 2. The product of their $\mathcal{FLE}^+$-description graphs is displayed in Figure 3. The edges between v_{b2} and v_{c1} are cross edges.

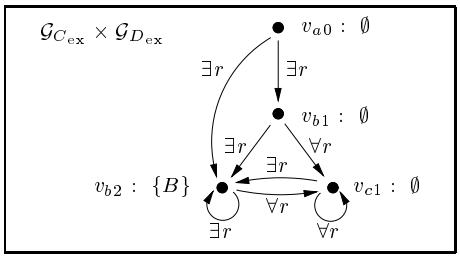

Fig. 3: Product Graph for $\mathcal{G}_{C_{\text{ex}}}$ and $\mathcal{G}_{D_{\text{ex}}}$

Once the product graph is obtained, we need to transform this representation into a $\mathcal{FLE}^+$-concept description. In order to apply the conc function introduced in Section 3, we have to check whether the obtained graph is a simple description graph. Unfortunately, this is not necessarily the case since the product graph may contain cross edges (w.r.t. a breadth-first spanning tree). Cross edges violate Condition 1 from Definition 5. We therefore have to eliminate them before applying the function conc to read out a concept description from a product graph. The elimination of cross edges is performed by an unraveling function introduced in Definition 10. The idea is to introduce a new vertex in $\mathcal{G}$' for each path over distinct vertices in the original graph starting from the root vertex and then transform every cross edge (v, w) into a new tree edge (v, w') by redirecting it to a copy of the subgraph reachable from w. For the formal definition, an auxiliary function eliminate-cross-edges is introduced first.

Definition 10 (unravel-function). *Let $\mathcal{G} := (V, E, v_0, \ell_V, \ell_E)$ be an $\mathcal{FLE}$-description graph. For every non-empty path $p := (v_0 \dots v_n)$ in V, let* $\text{Tail}(p) := v_n$.

Denote by $p \cdot q$ the concatenation of two such paths. Let

$$\text{Final-Path}(\mathcal{G}) := \bigcup_{1 \leq i \leq |V|} \{(v_0 v_1 \ldots v_i) \in V^i \mid (v_j\, Q r\, v_{j+1}) \in E, v_j \neq v_k \text{ for } j \neq k\}.$$

Define the function eliminate-cross-edges *by*
eliminate-cross-edges$(\mathcal{G}) :=$ (Final-Path$(\mathcal{G}), E', \ell'_V, \ell'_E$), *where*

$$\begin{aligned} E' &:= \{(\langle p \rangle, \langle p \cdot v \rangle) \in V' \times V' \mid (\text{Tail}(p)\, Q r\, v) \in E\} \cup \\ &\quad \{(\langle p \cdot v \cdot q \rangle, \langle p \cdot v \rangle) \in V' \times V' \mid (\text{Tail}(q)\, Q r\, v) \in E\} \\ \ell'_V(p) &:= \ell_V(\text{Tail}(p)) \\ \ell'_E(p\, Q r\, q) &:= \ell_E(\text{Tail}(p)\, Q r\, \text{Tail}(q)) \end{aligned}$$

The set Final-Path$(\mathcal{G})$ contains vertices of the underlying spanning tree of $\mathcal{G}$. For a given input graph $\mathcal{G}$, the result unravel(G) is constructed in three steps: firstly, remove forward edges from $\mathcal{G}$; secondly, apply the function eliminate-cross-edges on the resulting graph, and; thirdly, augment the resulting graph by the transitive-closure over all exists-edges. It can be shown that the graph obtained by the function unravel is equivalent to the original one.

Lemma 3. *Let C, D be $\mathcal{FLE}^+$-concept descriptions, then, (1)* unravel$(\mathcal{G}_C \times \mathcal{G}_D) \rightleftarrows \mathcal{G}_C \times \mathcal{G}_D$ *and* $\mathcal{G}_C \times \mathcal{G}_D \rightleftarrows$ unravel$(\mathcal{G}_C \times \mathcal{G}_D)$ *and (2)* unravel$(\mathcal{G}_C \times \mathcal{G}_D) \rightleftarrows \mathcal{G}_C$.

The underlying idea of the proof of (1) in this Lemma is to construct a simulation by extending the identity relation on $\mathcal{G}_C \times \mathcal{G}_D$ to the desired simulations by mapping the copied parts of the unraveled graph obtained from the unravel function to (or from resp.) the same vertices as their originals. For the exact proof refer to [9]. In this Lemma (2) is an immediate consequence of (1), since there always exists a simulation $Z' : \mathcal{G}_C \times \mathcal{G}_D \rightleftarrows \mathcal{G}_C$ and simulations are closed under concatenation.

Lemma 4. *Let C, D be $\mathcal{FLE}^+$-concept descriptions, then* unravel$(\mathcal{G}_C \times \mathcal{G}_D)$ *is a $\mathcal{FLE}^+$- description graph.*

Again, for the exact proof of this lemma refer to [9]. Since the graph obtained by the function unravel is a simple description graph, Theorem 1 is applicable and the concept description corresponding to the unraveled graph can be obtained by the conc function. We are now ready to prove the main theorem of this paper.

Theorem 3. *Let C, D be $\mathcal{FLE}^+$-concept descriptions then* conc(unravel$(\mathcal{G}_C \times \mathcal{G}_D)) \equiv \mathsf{lcs}(C, D)$.

Proof. Let $L =$ conc(unravel$(\mathcal{G}_C \times \mathcal{G}_D)$). We have to show that (1) $C \sqsubseteq L$ and $D \sqsubseteq L$ and (2) if there exist another $\mathcal{FLE}^+$-concept E with $E \sqsubseteq L$, $C \sqsubseteq E$, and $D \sqsubseteq E$ then $L \sqsubseteq E$.

Proof of (1): It is sufficient to show $C \sqsubseteq L$. Lemma 3 implies that there exists a simulation Z: unravel$(\mathcal{G}_C \times \mathcal{G}_D) \rightleftarrows \mathcal{G}_C$. Applying Lemma 4 to the unraveled graph and by the definition of $\mathcal{G}_C$ we know that unravel$(\mathcal{G}_C \times \mathcal{G}_D)$ and $\mathcal{G}_C$ are both

simple description graphs. Thus by Lemma 1 it holds that $\mathcal{G}_C \sqsubseteq \text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)$. From Theorem 1 it follows that $\text{unravel}(\mathcal{G}_C \times \mathcal{G}_D) \equiv \text{conc}(\text{unravel}(\mathcal{G}_C \times \mathcal{G}_D))$ and since $\mathcal{G}_C$ is a $\mathcal{FLE}^+$-description graph Lemma 2 can be applied and we can conclude that $\mathcal{G}_C \equiv C \sqsubseteq \text{conc}(\text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)) \equiv \text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)$.

Proof of (2): By contradiction: assume $E \sqsubseteq L$, $C \sqsubseteq E$, $D \sqsubseteq E$ and $L \not\sqsubseteq E$. Let $\mathcal{G}_A := (V_A, E_A, v_0^A, \ell_V^A, \ell_E^A)$ where $A \in \{C, D, E, L\}$. From $C \sqsubseteq E$, $D \sqsubseteq E$ and Theorem 2 follows that there exist simulations $Z_C : \mathcal{G}_E \rightleftharpoons \mathcal{G}_C$ and $Z_D : \mathcal{G}_E \rightleftharpoons \mathcal{G}_D$. Thus it holds by definition of simulations: $\forall v \in V_E$:

- $\forall v_F \in V_F$: If $v_F \in Z_F(v)$ then $\ell_V^E(v) \subseteq \ell_V^F(v_F)$, and
- $\forall (v\, Q r\, w) \in E_E$ there exist $v_F, w_F \in V_F$ s.t. $\{v_F\} \in Z_F(v), \{w_F\} \in Z_F(w)$ and $(v_F\, Q r\, w_F) \in E_F$,

where $F \in \{C, D\}$. From the existence of both simulation relations and from Definition 9 follows that for all $v \in V_E$:

- If $v_C \in Z_C(v)$ and $v_D \in Z_D(v)$ for $v_C \in V_C, (v_C\, Q r\, w_C) \in E_C$ and for $v_D \in V_D, (v_D\, Q r\, w_D) \in E_D$ then there exist the vertices $\{(v_C, v_D), (w_C, w_D)\} \in V_{\mathcal{G}_C \times \mathcal{G}_D}$ and $((v_C, v_D)\, Q r\, (w_C, w_D)) \in E_{\mathcal{G}_C \times \mathcal{G}_D}$.
- Since $\ell_V^E(v) \subseteq \ell_V^C(v_C) \cup \ell_V^D(v_D) = \ell_V^{\mathcal{G}_C \times \mathcal{G}_D}((v_C, v_D))$

Thus there exists a simulation relation $Z_L : \mathcal{G}_E \rightleftharpoons \mathcal{G}_C \times \mathcal{G}_D$, where $Z_L(v) = \{(v'v'') \in V_{\mathcal{G}_C \times \mathcal{G}_D} \mid v' \in Z_C(v), v'' \in Z_D(v)\}$. By Lemma 3 there also must exist a simulation $Z_L' : \mathcal{G}_E \rightleftharpoons \text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)$. Since $\mathcal{G}_E$ and $\text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)$ are simple description graphs, Lemma 1 implies $\mathcal{G}_E \sqsubseteq \text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)$. From this we obtain with Lemma 2 and Lemma 4, that $\mathcal{G}_E \equiv E \sqsubseteq \text{conc}(\text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)) \equiv \text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)$. This is a contradiction to our initial assumption. Thus we can conclude that $\text{conc}(\text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)) \equiv \mathsf{lcs}(C, D)$.

In case the n-ary lcs is to be computed from a set of concepts, the product of all corresponding $\mathcal{FLE}^+$-description graphs should be computed first and then the unravel and the conc function should be applied only once.

4 Conclusion and Outlook

We have introduced a sound and complete algorithm for the computation of the lcs in the DL $\mathcal{FLE}^+$. In particular, the lcs of a finite set of $\mathcal{FLE}^+$-concept descriptions always exists and is uniquely determined up to equivalence. As a key utility for the lcs computation we have proposed description graphs as a finite representation of $\mathcal{FLE}^+$-concept descriptions in which all restrictions additionally imposed by transitive roles are made explicit. The lcs could thus be defined by means of the graph product of the description graphs of the input concepts.

It is easy to see that the lcs algorithm can be optimized in several ways to produce smaller output concept descriptions. Firstly, the blocking conditions used to generate description graphs out of concept descriptions so far only allow for blocking w.r.t. ancestors. This might be replaced by a more general blocking strategy capable of blocking between arbitrary vertices. Secondly, it seems

expedient to reduce redundancies possibly produced by the function conc. In particular, it is not always necessary to apply the acc-function once for every edge in the description graph. A thorough investigation of the computational complexity of the lcs computation in $\mathcal{FLE}^+$ remains future work. Nevertheless, already for then non-transitive language $\mathcal{FLE}$ it is known that the lcs may be exponentially large in the input size.

References

1. F. Baader. Least common subsumers, most specific concepts, and role-value-maps in a description logic with existential restrictions and terminological cycles. LTCS-Report LTCS-02-07, Chair f. Automata Theory, Inst. f. Theor. Comp. Sci. TU Dresden, Germany, 2002.
2. F. Baader and R. Küsters. Unification in a description logic with inconsistency and transitive closure of roles. In I. Horrocks and S. Tessaris, eds., *Proc. of the 2002 International Workshop on Description Logics*, Toulouse, France, 2002.
3. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. LTCS-Report LTCS-98-09, LuFG Theoretical Comp. Sci. RWTH Aachen, Germany, 1998.
4. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In T. Dean, ed., *Proc. of IJCAI-99*, p. 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann.
5. F. Baader and P. Narendran. Unification of concept terms in description logics. In H. Prade, ed., *Proc. of ECAI-98*, p. 331–335. John Wiley & Sons Ltd, 1998.
6. S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. In *Proc of KI-01*.
7. A. Borgida and D. L. McGuinness. Asking queries about frames. In Luigia C. Aiello, John Doyle, and Stuart C. Shapiro, eds., *Proc. of KR-96*, p. 340–349, Cambridge, MA, 1996. Morgan Kaufmann.
8. S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proc. of KIDLWS'01*, nr 44 in CEUR-WS, Vienna, Austria, 2001. RWTH Aachen.
9. S. Brandt, A.-Y. Turhan, and R. Küsters. Foundations of non-standard inferences for description logics with transitive roles. LTCS-Report 03-02, Chair f. Automata Theory, Inst. f. Theor. Comp. Sci. TU Dresden, Germany, 2003.
10. Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.
11. W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, ed., *Proc. of AAAI-92*, San Jose, CA, 1992. AAAI Press.
12. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, ed., *Foundation of Knowledge Representation*. CSLI Publication, Cambridge University Press, 1996.
13. V. Haarslev and R. Möller. Expressive abox reasoning with number restrictions, role hierarchies, and transitively closed roles. In *Proc. of KR-00*, 2000.
14. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles. *J. of Logic and Computation*, 9(3):385–410, 1999.