Computing least common subsumers for $\mathcal{FLE}^+$

Sebastian Brandt and Anni-Yasmin Turhan *
Theoretical Computer Science, TU Dresden, Germany
Email: {brandt, turhan}@tcs.inf.tu-dresden.de

Abstract

Transitive roles are important for adequate representation of knowledge in a range of applications. In this paper we present a first algorithm to compute least common subsumers in a description logic with transitive roles.

1 Introduction

In $\mathcal{FLE}^+$ a role can be defined transitive so as to represent 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 which again consist of several devices. A natural way to represent such a machine by means of DLs would be to use some transitive *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.

Algorithms for many standard inference problems, e.g., satisfiability and subsumption, have been extended successfully to cope with transitive roles [8, 7] and have been put into practice in state of the art DL Systems. In contrast, comparatively little research exists on non-standard inferences in DLs with transitive roles [2, 1].

In this paper we present an effective algorithm to compute the least common subsumer (lcs) of $\mathcal{FLE}^+$-concept descriptions. The lcs was first mentioned as a non-standard inference problem for DLs in [6]. Intuitively, its purpose is to extract the commonalities of a set of concepts: given two $\mathcal{FLE}^+$-concept descriptions A and B the lcs of A and B is defined as the least (w.r.t. subsumption) $\mathcal{FLE}^+$-concept description subsuming A and B. It has been argued in [4] that the lcs facilitates a "bottom-up"-approach to modeling tasks: a domain expert can select a number of intuitively related concepts already existent in a KB and use the lcs operation to automatically construct a new concept representing the closest generalization of them.

DLs are based on a set of concept names (N_C), a set of role names (N_R), and a set of transitive role names N_R^T, where $N_R \cap N_R^T = \emptyset$. The DL $\mathcal{FLE}$ allows for the top-concept, conjunction, existential, and value restriction with a model-theoretic semantics defined in the usual way. In $\mathcal{FLE}^+$, transitive roles can be used in existential and value restrictions. Note that all concept descriptions in $\mathcal{FLE}^+$ are satisfiable. The lcs is formally defined as follows:

*Supported by the Deutsche Forschungsgemeinschaft, DFG Project BA 1122/4-3.

Definition 1 (lcs) *Let $C_1, \ldots, C_n$ be $\mathcal{FLE}^+$-concept descriptions. The $\mathcal{FLE}^+$-concept description C is the* least common subsumer *(lcs) of $C_1, \ldots, C_n$ 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 lcs is uniquely determined up to equivalence. In $\mathcal{FLE}$ and its sublanguages the lcs has been investigated by Baader et al. [3]. In their approach, concept descriptions are represented by *description trees*, i.e., syntax trees of a normal form in which all implicit information is made explicit. Subsumption of concept descriptions is then characterized by means of *homomorphisms* between the relevant description trees. The lcs can be computed as the *product tree* of the syntax trees of the input concepts.

In order to devise an lcs algorithm for $\mathcal{FLE}^+$, we extend this approach in three respects. Firstly, description trees are extended to *description graphs* in which the the transitivity of roles is explicitly represented. Secondly, subsumption between $\mathcal{FLE}^+$-concept descriptions is characterized by means of *simulation relations* instead of homomorphisms. Thirdly, the *graph product* is used for the actual lcs computation. The main challenge with this approach is to translate concept descriptions into description graphs and back, a comparatively simple step in the non-transitive case. All details and relevant proofs, which are largely omitted here, can be found in [5].

2 Representing $\mathcal{FLE}^+$-concept descriptions

In this section we introduce $\mathcal{FLE}^+$-description graphs to represent explicitly all aspects of $\mathcal{FLE}^+$-concept descriptions. Trees do not suffice for this purpose because, e.g., a value restrictions over a transitive role would imply an infinite path w.r.t. that role.

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*.*

There is no trivial correspondence between concept descriptions and description graphs. Hence, in order to speak of the *equivalence* of concept descriptions and description graphs we need to define a semantics for description graphs. Intuitively, a model I is a model of a description graph $\mathcal{G}$ iff there exists a mapping $\pi\colon V_\mathcal{G} \to 2^{\Delta^\mathcal{I}} \setminus \emptyset$ such that (i) a vertex with a concept A in its label is only mapped onto witnesses of A and (ii) edge transitions in the graph are reflected in the model.

However, given these semantics there exist 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 *simple description graphs* which additionally satisfy certain structural conditions (See [5] for details). In particular, simple description graphs are free of cross edges.

For the translation of an $\mathcal{FLE}^+$-concept description C into an equivalent (simple) description graph $\mathcal{G}_C$ we require C to be in $\mathcal{FLE}$-normal form [3], i.e., with flattened conjunctions and value restrictions propagated over existential restrictions. The translation starts with a root vertex v_0 with C in its label. Then, certain graph generation

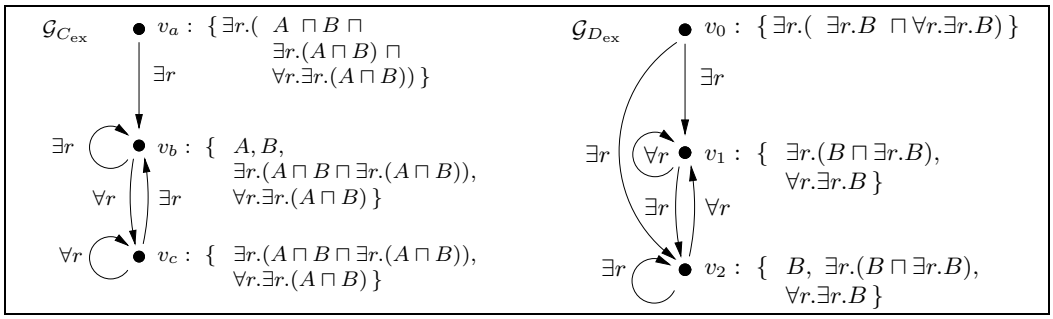

Figure 1: $\mathcal{FLE}^+$-description graphs (with complex labels)

rules (defined in detail in [5]) are applied exhaustively to produce new vertices and edges. Intuitively, these rules resemble tableaux rules in that concept descriptions occurring in the label of a vertex are used to extend the description graph "accordingly", i.e., if the label of a vertex v contains a concept of the form $\exists r.C'$ then an exists-edge $(v\, \exists r\, w)$ is added and C' is added to the label of w and so on.

To avoid generating infinitely many new vertices, every rule has a *blocking condition* testing whether or not a new vertex can be avoided by a back edge to an already existing one. If no more graph generation rules are applicable, all non-atomic concept descriptions are removed from the label sets of $\mathcal{G}$ and the graph is returned.

The following example shows the corresponding description graph of two $\mathcal{FLE}^+$-concept descriptions.

Example 3 *Let* $C_{ex} := \exists r.(B \sqcap A \sqcap \exists r.(B \sqcap A) \sqcap \forall r.(\exists r.B \sqcap A))$ *and* $D_{ex} := \exists r.(\exists r.B \sqcap \forall r.\exists r.B)$ *for* $r \in N_R^T$ *and* $A, B \in N_C$*. The corresponding* $\mathcal{FLE}^+$*-description graphs are depicted in Figure 1. The figure also shows the 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.*

For the backward translation from a (simple) description graph $\mathcal{G}$ to a concept description, the label sets of the vertices of $\mathcal{G}$ are iteratively augmented until the label of the root node contains an $\mathcal{FLE}^+$-concept description equivalent to $\mathcal{G}$. For every vertex v, its augmented label contains the same atomic labels as before but additionally has an existential restriction based on the label of every $\exists r$-successor of v. For example, if $\mathcal{G}$ contains an edge $(v\, \exists r\, w)$ and a concept C' is the only concept in the label set of w then $\exists r.C'$ is added to the label set of v in the next iteration. Forall-edges are treated similarly.

Note that this strategy is the reverse of the generation procedure for description graphs, where the label of the root vertex generated the entire description graph. Denoting by $\mathcal{G}_C$ the description graph of C and by $\text{conc}(\mathcal{G})$ the concept description obtained by the above backward translation the following theorem holds [5]:

Theorem 4 *For every* $\mathcal{FLE}^+$*-concept description* C *it holds that* $C \equiv \mathcal{G}_C$*. For every simple description graph* $\mathcal{G}$ *it holds that* $\text{conc}(\mathcal{G}) \equiv \mathcal{G}$*.*

The next step towards our lcs algorithm is to use the description graphs introduced previously to characterize subsumption of $\mathcal{FLE}^+$-concept descriptions.

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

Our aim is to characterize subsumption of $\mathcal{FLE}^+$-concept descriptions by simulation relations between the relevant description graphs.

Definition 5 (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 the following properties: (i) $(v_{02}, v_{01}) \in R$, (ii) $\ell_V(v) \cap N_C \subseteq \ell_V(v') \cap N_C$ for all $(v, v') \in R$, and (iii) if $(v\, Qr\, w) \in E_2$ and $(v, v') \in R$ then there exists a vertex $w' \in V_1$ such that $(v'\, Qr\, w') \in E_1$ and $(w, w') \in R$.*

If a simulation relation from $\mathcal{G}_2$ to $\mathcal{G}_1$ exists then $\mathcal{G}_2$ can be embedded into $\mathcal{G}_1$ in the sense that, (i) starting from the pair of root vertices, every edge in $\mathcal{G}_2$ can also be travelled in $\mathcal{G}_1$ and (ii) all atomic concepts in the label set of a vertex in $\mathcal{G}_2$ are also present in the corresponding vertices in $\mathcal{G}_1$. The following theorem provides the actual characterization of subsumption (See [5] for details):

Theorem 6 *Let C and D be $\mathcal{FLE}^+$-concept descriptions. Then, $C \sqsubseteq D$ iff $\mathcal{G}_D \rightleftharpoons \mathcal{G}_C$.*

Hence, subsumption of $\mathcal{FLE}^+$-concept descriptions can be shown by finding a simulation relation between their description graphs. To illustrate this, we return to the example introduced in the previous section.

Example 7 *Recall the concepts from Example 3. Apparently, $C_{ex} \sqsubseteq D_{ex}$. It is easy to see in Figure 1 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 and every edge which can be traveled 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 $\mathcal{G}_{D_{ex}}$.*

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

With all the information captured in an $\mathcal{FLE}$-concept description made explicit by $\mathcal{FLE}^+$-description graphs the next step is to extract commonalities of description graphs. Similar to other approaches to computing the lcs [1, 3] the graph product is employed to this end. In our case, an edge $((u_1, v_1), (u_2, v_2))$ is in the product graph $\mathcal{G} \times \mathcal{H}$ iff (u_1, u_2) is an edge in $\mathcal{G}$ and (v_1, v_2) one in $\mathcal{H}$ and both have the same edge label. Moreover, a vertex (u_1, v_1) in the product graph is labeled by the intersection of the label sets of u_1 and v_1. Note that we may restrict the product graph to the subgraph reachable from the root vertex.

The following example illustrates the notion of the product of description graphs.

Example 8 *Consider the description graphs $G_{C_{ex}}$ and $\mathcal{G}_{D_{ex}}$ from Example 7. The product $G_{C_{ex}} \times \mathcal{G}_{D_{ex}}$ is displayed in Figure 2. It is easy to see that (v_{b2}, v_{c1}) and (v_{c1}, v_{b2}) are cross edges w.r.t. a breadth-first search tree.*

The last step to computing the lcs is to translate product graphs back into concept descriptons. The difficulty here is that the graph product of simple description graphs may introduce cross edges which violate the conditions of simple description graphs.

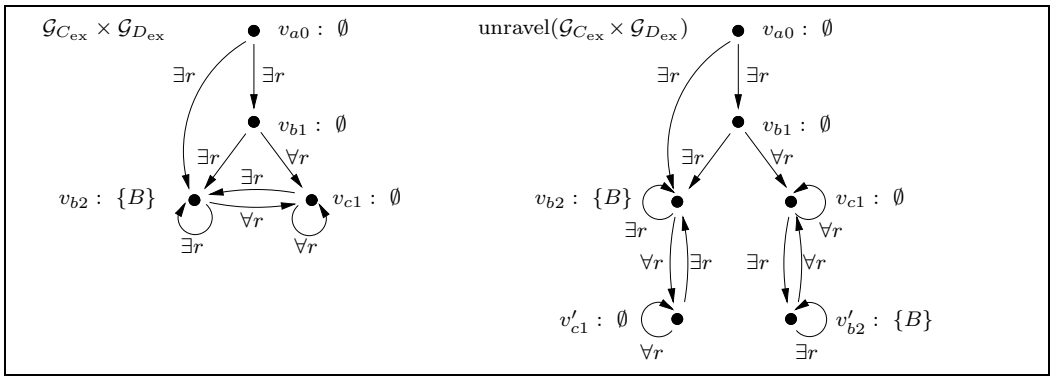

Figure 2: Product Graph and unraveled Product Graph for $\mathcal{G}_{C_{\text{ex}}}$ and $\mathcal{G}_{D_{\text{ex}}}$

Since only these are admissible for our backward translation, we need to eliminate cross edges by *unraveling* in the usual way. For our example $G_{C_{\text{ex}}} \times \mathcal{G}_{D_{\text{ex}}}$, the relevant cross-edge free graph is shown as unravel($G_{C_{\text{ex}}} \times \mathcal{G}_{D_{\text{ex}}}$) in Figure 2. In general, the following theorem [5] closes the last step towards computing the lcs of $\mathcal{FLE}^+$-concepts:

Theorem 9 *Let C, D be $\mathcal{FLE}^+$-concept descriptions and $\mathcal{G}_C, \mathcal{G}_D$ their corresponding $\mathcal{FLE}^+$-description graphs, then* $\text{conc}(\text{unravel}(\mathcal{G}_C \times \mathcal{G}_D)) \equiv \mathsf{lcs}(C, D)$.

The actual computation of the lcs of two $\mathcal{FLE}^+$-concept descriptions C, D can now be done in four steps: Firstly, compute the description graphs $\mathcal{G}_C$ and $\mathcal{G}_D$ of the normalized input concepts. Secondly, compute the unraveled product graph unravel($\mathcal{G}_C \times \mathcal{G}_D$). Thirdly, re-translate the result into a concept description. It can be shown that this procedure leads to a sound and complete algorithm for the computation of the lcs in $\mathcal{FLE}^+$. In particular, the lcs of a finite set of $\mathcal{FLE}^+$-concept descriptions always exists and is uniquely determined up to equivalence.

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. 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 DL2002*, Toulouse 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 Theor. Comp. Sci., RWTH Aachen, Germany, 1998. Abriged version in: In T. Dean, ed., *Proc. of IJCAI-99*, Stockholm, Sweden, 1999.

[4] S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proc. of KI-2001 Workshop on Applications of DLs (KIDLWS'01)*, nr. 44 in CEUR-WS, Vienna, Austria, 2001.

[5] 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.

[6] W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, ed., *Proc. of AAAI-92*, 1992.

[7] V. Haarslev and R. Möller. Expressive abox reasoning with number restrictions, role hierarchies, and transitively closed roles. In *Proc. of KR-00*, 2000.

[8] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.