A New n-ary Existential Quantifier in Description Logics

Franz Baader, Carsten Lutz, Eldar Karabaev
TU Dresden, Germany

Manfred Theißen
RWTH Aachen
Germany

Abstract

Motivated by a chemical process engineering application, we introduce a new concept constructor in Description Logics (DLs), an n-ary variant of the existential restriction constructor, which generalizes both the usual existential restrictions and so-called qualified number restrictions. We show that the new constructor can be expressed in $\mathcal{ALCQ}$, the extension of the basic DL $\mathcal{ALC}$ by qualified number restrictions. However, this representation results in an exponential blow-up. By giving direct algorithms for $\mathcal{ALC}$ extended with the new constructor, we can show that the complexity of reasoning in this new DL is actually not harder than the one of reasoning in $\mathcal{ALCQ}$. Moreover, in our chemical process engineering application, a restricted DL that provides only the new constructor together with conjunction, and satisfies an additional restriction on the occurrence of roles names, is sufficient. For this DL, the subsumption problem is polynomial.

1 Introduction

For the inference services of a DL system to be feasible, the underlying inference problems (like the subsumption problem) must at least be decidable, and preferably of low complexity. This is only possible if the expressiveness of the DL employed by the system is restricted in an appropriate way. Because of this restriction of the expressive power of DLs, various application-driven language extensions have been proposed in the literature (see, e.g., [2, 8, 19, 14]), some of which have been integrated into state-of-the-art DL systems [13, 11].

The present paper considers a new concept constructor that is motivated by a process engineering application [20]. This constructor is an n-ary variant of the usual existential restriction operator available in most DLs. To motivate the need for this new constructor, assume that we want to describe a chemical plant that has a reactor with a main reaction, and *in addition* a reactor with a main and a side reaction. Also assume that the concepts *Reactor_with_main_reaction* and *Reactor_with_main_and_side_reaction* are defined such that the first concept subsumes the second one. We could try to model this chemical plant with the help of the usual existential restriction operator as

$$\begin{aligned} &\mathit{Plant} \sqcap \exists \mathit{has\_part}.\mathit{Reactor\_with\_main\_reaction} \sqcap \\ &\qquad \exists \mathit{has\_part}.\mathit{Reactor\_with\_main\_and\_side\_reaction}. \end{aligned}$$

However, because of the subsumption relationship between the two reactor concepts, this concept is equivalent to

$$\mathit{Plant} \sqcap \exists \mathit{has\_part}.\mathit{Reactor\_with\_main\_and\_side\_reaction},$$

and thus does *not* capture the intended meaning of a plant having *two* reactors, one with a main reaction and the other with a main and a side reaction. To overcome this problem, we consider a new concept constructor of the form $\exists r.(C_1, \dots, C_n)$, with the intended meaning that it describes all individuals having n *different* r-successors $d_1, \dots, d_n$ such that d_i belongs to C_i $(i = 1, \dots, n)$. Given this constructor, our concept can correctly be described as

$$\begin{aligned} &\mathit{Plant} \sqcap \exists \mathit{has\_part}.(\mathit{Reactor\_with\_main\_reaction}, \\ &\qquad\qquad \mathit{Reactor\_with\_main\_and\_side\_reaction}). \end{aligned}$$

The situation differs from other application-driven language extensions in that the new constructor can actually be expressed using constructors available in the DL $\mathcal{ALCQ}$, which can be handled by state-of-the-art DL systems (Section 3). Thus, the new constructor can be seen as syntactic sugar; nevertheless, it makes sense to introduce it explicitly since this speeds up reasoning. In fact, expressing the new constructor with the ones available in $\mathcal{ALCQ}$ results in an exponential blow-up. In addition, the translation introduces many "expensive" constructors (disjunction and qualified number restrictions). For this reason, even highly optimized DL systems like RACER [11] cannot handle the translated concepts in a satisfactory way. In contrast, the direct introduction of the new constructor into $\mathcal{ALCQ}$ does not increase the complexity of reasoning (Section 4). Moreover, in the process engineering application [20] mentioned above, a rather inexpressive DL that provides only the new constructor together with conjunction is sufficient. In addition, only concept descriptions are used where in each conjunction there is at most one n-ary existential restriction for each role. For this restricted DL, the subsumption problem turns out to be polynomial (Section 5).

2 The DL $\mathcal{ALCQ}$

Concept descriptions are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. The

Name	Syntax	Semantics
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
at-least qualified number restriction	$\geqslant n\, r.C$	$\{x \mid card(\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}) \geq n\}$

Table 1: Syntax and semantics of $\mathcal{ALCQ}$.

constructors determine the expressive power of the DL. In this section, we restrict the attention to the DL $\mathcal{ALCQ}$, whose concept descriptions are formed using the constructors shown in Table 1. Using these constructors, several other constructors can be defined as abbreviations:

- $C \sqcup D := \neg(\neg C \sqcap \neg D)$ (disjunction),
- $\top := A \sqcup \neg A$ for a concept name A (top-concept),
- $\exists r.C := \geqslant 1\, r.C$ (existential restriction),
- $\forall r.C := \neg \exists r.\neg C$ (value restriction),
- $\leqslant n\, r.C := \neg(\geqslant (n+1)\, r.C)$ (at-most restriction).

The semantics of $\mathcal{ALCQ}$-concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each role $r \in N_R$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1. Here, the function *card* yields the cardinality of the given set.

A *general* $\mathcal{ALCQ}$*-TBox* is a finite set of general concept inclusions (GCIs) $C \sqsubseteq D$ where C, D are $\mathcal{ALCQ}$-concept descriptions. The interpretation $\mathcal{I}$ is a model of the general $\mathcal{ALCQ}$-TBox $\mathcal{T}$ iff it satisfies all its GCIs, i.e., if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all GCIs $C \sqsubseteq D$ in $\mathcal{T}$.

We use $C \equiv D$ as an abbreviation of the two GCIs $C \sqsubseteq D$, $D \sqsubseteq C$. An *acyclic* $\mathcal{ALCQ}$*-TBox* is a finite set of *concept definitions* of the form $A \equiv C$ (where A is a concept name and C an $\mathcal{ALCQ}$-concept description) that does not contain multiple definitions or cyclic dependencies between the definitions. Concept names occurring on the left-hand side of a concept definition are called *defined* whereas the others are called *primitive*.

Given two $\mathcal{ALCQ}$-concept descriptions C, D we say that C *is subsumed by* D *w.r.t. the general TBox* $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. Subsumption w.r.t. an acyclic TBox and subsumption between concept descriptions (where $\mathcal{T}$ is empty) are special cases of this definition. In the latter case we write $C \sqsubseteq D$ in place of $C \sqsubseteq_\emptyset D$. The concept description C is *satisfiable* (w.r.t. the general TBox $\mathcal{T}$) iff there is an interpretation $\mathcal{I}$ (a model $\mathcal{I}$ of $\mathcal{T}$) such that $C^\mathcal{I} \neq \emptyset$.

The complexity of the subsumption problem in $\mathcal{ALCQ}$ depends on the presence of GCIs. Subsumption of $\mathcal{ALCQ}$-concept descriptions (with or without acyclic TBoxes) is PSpace-complete and subsumption w.r.t. a general $\mathcal{ALCQ}$-TBox is ExpTime-complete [21].[1] These results hold both for unary and binary coding of the numbers in number restriction, but in this paper we restrict the attention to unary coding (where the size of the number n is counted as n rather than $\log n$).

3 The new constructor

The general syntax of the new constructor is

$$\exists r.(C_1, \ldots, C_n)$$

where $r \in N_R$, $n \geq 1$, and $C_1, \ldots, C_n$ are concept descriptions. We call this expression an *n-ary existential restriction*. Its semantics is defined as

$$\exists r.(C_1, \ldots, C_n)^\mathcal{I} := \{x \mid \exists y_1, \ldots, y_n.\ (x, y_1) \in r^\mathcal{I} \wedge \ldots \wedge (x, y_n) \in r^\mathcal{I} \wedge \\ y_1 \in C_1^\mathcal{I} \wedge \ldots \wedge y_n \in C_n^\mathcal{I} \wedge \bigwedge_{1 \leq i < j \leq n} y_i \neq y_j\}.$$

We call the DL whose concept descriptions are formed using the constructors conjunction, negation, and n-ary existential restriction $\mathcal{EL}^{(n)}\mathcal{C}$. It is an immediate consequence of the semantics of n-ary existential restrictions that the at-least restriction $\geqslant n\, r.C$ of $\mathcal{ALCQ}$ can be expressed by the n-ary existential restriction $\exists r.(C, \ldots, C)$.[2] Consequently, all of $\mathcal{ALCQ}$ can be expressed within $\mathcal{EL}^{(n)}\mathcal{C}$.

Conversely, can we express n-ary existential restrictions within $\mathcal{ALCQ}$? We have seen in the introduction that, in general, $\exists r.(C_1, \ldots, C_n)$ cannot be replaced by the conjunction $\exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ since this conjunction does not ensure the existence of n *different* r-successors. However, $\mathcal{ALCQ}$ provides us with the more expressive qualified number restriction constructor. Let us first consider the case $n = 2$. We claim that $\exists r.(C_1, C_2)$ can be expressed by the $\mathcal{ALCQ}$-concept description

$$D := (\geqslant 1\, r.C_1) \sqcap (\geqslant 1\, r.C_2) \sqcap (\geqslant 2\, r.(C_1 \sqcup C_2)).$$

It is clear that any individual belonging to $\exists r.(C_1, C_2)$ also belongs to D. Conversely, assume that x belongs to D. Then x has two distinct r-successors y_1, y_2,

[1] In [21], acyclic TBoxes are not considered, but it is easy to show that the usual approach for handling acyclic TBoxes without using exponential space [16] extends to $\mathcal{ALCQ}$ (see [5]).

[2] Since we assume unary coding of numbers in number restrictions, this translation is linear. Otherwise, it would be exponential.

both belonging to $C_1 \sqcup C_2$. If one of them belongs to C_1 and the other to C_2, then we are done. Otherwise, we have two cases: (i) both belong to $C_1 \sqcap \neg C_2$, or (ii) both belong to $\neg C_1 \sqcap C_2$. We restrict our attention to the first case (since the second is symmetric). Due to the conjunct $\geqslant 1\, r.C_2$ in D, x has an r-successor in C_2, which is different from y_1 since y_1 does not belong to C_2. Consequently, there are two distinct r-successors of x, one belonging to C_1 and the other belonging to C_2, which shows that x belongs to $\exists r.(C_1, C_2)$.

This result can be extended to arbitrary n.

Theorem 1 *The n-ary existential restriction constructor can be expressed within $\mathcal{ALCQ}$, and thus $\mathcal{ALCQ}$ and $\mathcal{EL}^{(n)}\mathcal{C}$ have the same expressive power.*

To prove this theorem we show that $\exists r.(C_1, \ldots, C_n)$ can be expressed by the $\mathcal{ALCQ}$-concept description

$$D_n := \bigsqcap_{\{i_1,\ldots,i_k\}\subseteq\{1,\ldots,n\}} (\geqslant k\, r.(C_{i_1} \sqcup \ldots \sqcup C_{i_k})).$$

It is again clear that any individual belonging to the concept $\exists r.(C_1, \ldots, C_n)$ also belongs to D_n. The other direction is an easy consequence of Hall's theorem [12]. Let $F = (S_1, \ldots, S_n)$ be a finite family of sets. This family has a *system of distinct representatives (SDR)* iff there are n distinct elements $s_1, \ldots, s_n$ such that $s_i \in S_i$ $(i = 1, \ldots, n)$.

Theorem 2 (Hall) *The family $F = (S_1, \ldots, S_n)$ has an SDR iff card$(S_{i_1} \cup \ldots \cup S_{i_k}) \geq k$ for all $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, where $i_1, \ldots, i_k$ are distinct.*

Now, assume that the individual x belongs to D_n. For $i = 1, \ldots, n$, let S_i be the set of r-successors of x that belong to C_i. By the definition of D_n, the family $(S_1, \ldots, S_n)$ satisfies the condition of Hall's theorem, and thus it has an SDR. This SDR obviously shows that x belongs to $\exists r.(C_1, \ldots, C_n)$.

The proof of Theorem 1 shows that the subsumption problem in $\mathcal{EL}^{(n)}\mathcal{C}$ can be reduced to the subsumption problem in $\mathcal{ALCQ}$, and thus DL systems like Racer that can handle $\mathcal{ALCQ}$ can in principle be used to compute subsumption in $\mathcal{EL}^{(n)}\mathcal{C}$. However, the translation from $\mathcal{EL}^{(n)}\mathcal{C}$ into $\mathcal{ALCQ}$ is obviously exponential. In addition, the constructs it introduces (disjunctions and qualified number restrictions) are hard to handle for tableau-based subsumption algorithms like the one used by Racer. In fact, faced with the $\mathcal{ALCQ}$-translations of the $\mathcal{EL}^{(n)}\mathcal{C}$-concept descriptions

$$\begin{aligned} C &:= \exists r.(A_1 \sqcap B_1, A_2 \sqcap B_2, A_3 \sqcap B_3, A_4 \sqcap B_4), \\ D &:= \exists r.(A_1, A_2, A_3, A_4), \end{aligned}$$

it takes RACER[3] 57 minutes to find out that $C \sqsubseteq D$. For the 5-ary variant of this example, RACER did not finish its computation within 4 hours.

This problem can be due either to the inherently higher complexity of reasoning in $\mathcal{EL}^{(n)}\mathcal{C}$, or to the translation. We will see in the next section that the latter is the culprit.

4 Complexity of reasoning in $\mathcal{EL}^{(n)}\mathcal{C}$

The exponential translation of $\mathcal{EL}^{(n)}\mathcal{C}$-concepts into $\mathcal{ALCQ}$-concepts together with the known complexity of the subsumption problem in $\mathcal{ALCQ}$ (PSPACE for subsumption of concept descriptions and EXPTIME for subsumption w.r.t. a general TBox) yields the following complexity upper-bounds for the subsumption problem in $\mathcal{EL}^{(n)}\mathcal{C}$: EXPSPACE for subsumption of concept descriptions and 2EXPTIME for subsumption w.r.t. a general TBox. The next theorem shows that these upper-bounds are not optimal.

Theorem 3 *The subsumption problem in $\mathcal{EL}^{(n)}\mathcal{C}$ is* PSPACE-*complete for subsumption between concept descriptions and* EXPTIME-*complete for subsumption w.r.t. a general TBox.*

The hardness results are an immediate consequence of the corresponding hardness results [10] for the subsumption problem in $\mathcal{ALC}$ (which allows for conjunction, negation, and existential restrictions). The complexity upper-bounds can be shown by relatively simple adaptations of well-known algorithmic approaches used in modal logics to show similar results (see [4] for more details). To show the PSPACE-upper bound, one can adapt the "witness algorithm" (also called **K**-worlds algorithm) commonly used in modal logics to show that satisfiability in the modal logic **K** is in PSPACE (see, e.g., [6]). The EXPTIME-upper bound can be proved by an adaptation of Pratt's "elimination of Hintikka sets" approach to show that satisfiability in propositional dynamic logic (PDL) is in EXPTIME (see also [6]).

5 A tractable sublanguage

In the chemical process engineering application mentioned above [20], the full expressive power of $\mathcal{EL}^{(n)}\mathcal{C}$ is actually not needed. This application is concerned with supporting the construction of mathematical models of process systems by storing building blocks for such models in a class hierarchy. In order to retrieve building blocks, one can then either browse the hierarchy or formulate

[3]RACER Version 1.7.23; on a Pentium 4 machine, 2 Ghz, 2 GB memory; under Redhat Linux.

query classes. In both cases, the existence of efficient algorithms for computing subsumption between class descriptions is an important prerequisite.

The frame-like formalism for describing classes of such building blocks introduced in [20] can be expressed in the *sublanguage* $\mathcal{EL}^{(n)}$ *of* $\mathcal{EL}^{(n)}\mathcal{C}$, which allows for conjunction, n-ary existential restrictions, and the top concept. Moreover, since in each frame a given slot-name can be used only once, it is sufficient to consider *restricted* $\mathcal{EL}^{(n)}$*-concept descriptions* where in each conjunction there is at most one n-ary existential restriction for each role: an $\mathcal{EL}^{(n)}$-concept description is *restricted* iff it is of the form

$$A_1 \sqcap \ldots \sqcap A_n \sqcap \exists r_1.(B_{1,1}, \ldots, B_{1,\ell_1}) \sqcap \ldots \sqcap \exists r_m.(B_{m,1}, \ldots, B_{m,\ell_m}),$$

where $A_1, \ldots, A_n$ are concept names, $r_1, \ldots, r_m$ are *distinct* role names, and $B_{1,1}, \ldots, B_{m,\ell_m}$ are restricted $\mathcal{EL}^{(n)}$-concept descriptions. For example, the $\mathcal{EL}^{(n)}$-concept description $\exists r.(A, \exists r.(B, C)) \sqcap \exists s.(A, A)$ is restricted whereas the description $\exists r.(A, \exists r.(B, C)) \sqcap \exists r.(A, A)$ is not.

As in the case of $\mathcal{EL}$ [3], the corresponding DL with unary existential restrictions, restricted $\mathcal{EL}^{(n)}$-concept descriptions can be translated into $\mathcal{EL}^{(n)}$-*description trees*, where the nodes are labeled with sets of concept names and the edges are labeled with role names. For example, the restricted $\mathcal{EL}^{(n)}$-concept descriptions

$$A \sqcap \exists r.(A, B \sqcap \exists r.(B, A), \exists r.(A, A \sqcap B)) \quad \text{and} \quad A \sqcap \exists r.(A, B, \exists r.(A, A))$$

yield the description trees depicted in Fig. 1. Given a restricted $\mathcal{EL}^{(n)}$-concept

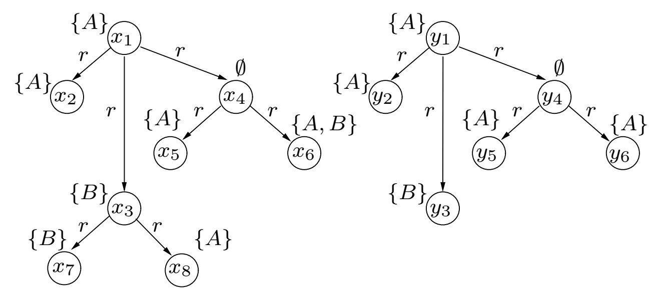

Figure 1: Two $\mathcal{EL}^{(n)}$-description trees.

description C, we denote the corresponding description tree by T_C. Formally, this tree is described by a tuple $T_C = (V, E, v_0, \ell)$, where V is the finite set of nodes, $E \subseteq V \times N_R \times V$ is the set of N_R-labeled edges, $v_0 \in V$ is the root, and $\ell : V \to 2^{N_C}$ is the node labeling function.

In [3], it was shown that subsumption between $\mathcal{EL}$-concept descriptions corresponds to the existence of a homomorphism between the corresponding description trees. In $\mathcal{EL}^{(n)}$ we must additionally require that the homomorphism is injective.

Definition 4 Given two $\mathcal{EL}^{(n)}$-description trees $T_1 = (V_1, E_1, v_{0,1}, \ell_1)$ and $T_2 = (V_2, E_2, v_{0,2}, \ell_2)$, a *homomorphism* $\varphi : T_1 \to T_2$ is a mapping $\varphi : V_1 \longrightarrow V_2$ s.t.

- $\varphi(v_{0,1}) = v_{0,2}$,
- $\ell_1(v) \subseteq \ell_2(\varphi(v))$ for all $v \in V_1$, and
- $(\varphi(v), r, \varphi(w)) \in E_2$ for all $(v, r, w) \in E_1$.

This homomorphism is an *embedding* iff the mapping $\varphi : V_1 \longrightarrow V_2$ is injective.

For example, mapping y_i to x_i for $i = 1, \ldots, 6$ yields an embedding from the description tree on the right-hand side of Fig. 1 to the description tree on the left-hand side. If we changed the label of x_6 to $\{B\}$, then there would still exist a homomorphism between the two trees (mapping both y_5 and y_6 onto x_5), but not an embedding. The following theorem can be shown similarly to the proof of the corresponding result for $\mathcal{EL}$ [3].

Theorem 5 *Let C, D be restricted $\mathcal{EL}^{(n)}$-concept descriptions and T_C, T_D the corresponding description trees. Then $C \sqsubseteq D$ iff there exists an embedding from T_D into T_C.*

To show that subsumption between restricted $\mathcal{EL}^{(n)}$-concept descriptions is a polynomial-time problem, it remains to be shown that the existence of an embedding can be decided in polynomial time. First, let us recall the well-known bottom-up approach for testing for the existence of a homomorphism [18, 3].

Let $T_1 = (V_1, E_1, v_{0,1}, \ell_1)$ and $T_2 = (V_2, E_2, v_{0,2}, \ell_2)$ be two $\mathcal{EL}^{(n)}$-description trees, and assume that we want to check whether there is a homomorphism from T_1 to T_2. The idea underlying the polynomial time test is to compute, for each $v \in V_1$, the set $\delta(v)$ of all nodes $w \in V_2$ such that there is a homomorphism from the subtree of T_1 with root v to the subtree of T_2 with root w. Once these sets δ are computed for all nodes of T_1, we can simply check whether $v_{0,2}$ belongs to $\delta(v_{0,1})$. The sets $\delta(v)$ are computed in a bottom-up fashion, where a node is treated only after all its successor nodes have been considered:[4]

1. If v is a leaf of T_1, then $\delta(v)$ simply consists of all the nodes $w \in V_2$ such that $\ell_1(v) \subseteq \ell_2(w)$.
2. Let v be a node of T_1 and let $(v, r_1, v_1), \ldots, (v, r_k, v_k)$ be all the edges in E_1 with first component v. Since we work bottom up, we know that the sets $\delta(v_1), \ldots, \delta(v_k)$ have already been computed. The set $\delta(v)$ consists of all the nodes $w \in V_2$ such that

[4] For example, one can use a postorder tree walk [9] of the nodes of T_1 to realize this.

(a) $\ell_1(v) \subseteq \ell_2(w)$ and

(b) for each $i, 1 \leq i \leq k$ there exists a node $w_i \in \delta(v_i)$ such that $(w, r_i, w_i) \in E_2$.

It is easy to show that this indeed yields a polynomial-time algorithm for checking the existence of a *homomorphism* between two $\mathcal{EL}^{(n)}$-description trees.

If we want to test for the existence of an *embedding*, we must modify Step 2 of this algorithm. In fact, we must ensure that distinct r-successors of v can be mapped to distinct r-successors of w. This can be achieved as follows:

2′. Let v be a node of T_1, and for each role r let $(v, r, v_{1,r}), \ldots, (v, r, v_{k_r,r})$ be the edges in E_1 with first component v and label r. Since we work bottom up, we know that the sets $\delta(v_{1,r}), \ldots, \delta(v_{k_r,r})$ have already been computed. The set $\delta(v)$ consists of all the nodes $w \in V_2$ satisfying the following two properties:

(a) $\ell_1(v) \subseteq \ell_2(w)$,

(b) for all roles r, the family $F_r(w) := (S_{1,r}(w), \ldots,\ S_{k_r,r}(w))$ has an SDR, where the members of this family are defined as

$$S_{i,r}(w) := \{w' \in \delta(v_{i,r}) \mid (w, r, w') \in E_2\}.$$

Obviously, the existence of an SDR for $F_r(w)$ allows us to map the r-successors of v to *distinct* r-successors of w, and thus construct an embedding. For this algorithm to be polynomial, it remains to be shown that the existence of an SDR can be decided in polynomial time. Note that Hall's characterization of the existence of an SDR obviously does not yield a polynomial-time procedure. However, checking for the existence of an SDR is basically the same as solving the maximum bipartite matching problem, which can be done in polynomial time since it can be reduced to a network flow problem [9].

To be more precise, let $(L \cup R, E)$ be a bipartite graph, i.e., $L \cap R = \emptyset$ and $E \subseteq L \times R$. A *matching* is a subset M of E such that each node in $L \cup R$ occurs at most once in M. This matching is called *maximum* iff there is no other matching having a larger cardinality. As shown in [9], such a maximum matching can be computed in time polynomial in the cardinality of V and E.

Let $F = (S_1, \ldots, S_n)$ be a finite family of finite sets, and let $L := \{1, \ldots, n\}$ and $R = S_1 \cup \ldots \cup S_n$.[5] We define the set of edges of the bipartite graph $G_F = (L \cup R, E)$ as follows:

$$E := \{(i, s) \mid s \in S_i\}.$$

[5] Without loss of generality we can assume that $L \cap R = \emptyset$.

It is easy to see that the family F has an SDR iff the corresponding bipartite graph G_F has a maximum matching of cardinality n. In fact, $(1, s_1), \ldots, (n, s_n)$ is a maximum matching iff $s_1, \ldots, s_n$ is an SDR.

Thus, we have shown that the existence of an embedding can be decided in polynomial time. Together with Theorem 5, this yields the following tractability result:

Corollary 6 *Subsumption between restricted $\mathcal{EL}^{(n)}$-concept descriptions can be decided in polynomial time.*

A first implementation of this polynomial-time algorithm behaves much better than the translation approach on the example concept descriptions C, D from Section 3 and their obvious extensions to larger n. For small n, the subsumption relationship is found immediately (i.e., with no measurable run-time), and even for $n = 100$, the runtime (of our unoptimized implementation) is just 1 second.

It is not hard to show that this polynomiality result can be extended to the case of *restricted acyclic $\mathcal{EL}^{(n)}$-TBoxes*, where an acyclic $\mathcal{EL}^{(n)}$-TBox is restricted if its expansion (i.e., the TBox obtained by exhaustively replacing defined concepts by their definition) contains only restricted $\mathcal{EL}^{(n)}$-concept descriptions (see [4] for more details).

6 Related and future work

Polynomiality of the subsumption problem in $\mathcal{EL}$ was shown in [3] as a by-product of the characterization of subsumption via the existence of homomorphisms between the corresponding description trees. This result can also be obtained as a consequence of the fact that the containment problem $Q_1 \subseteq Q_2$ for conjunctive queries is polynomial if Q_2 is acyclic [22, 17]. Since it is easy to see that $\mathcal{EL}^{(n)}$-concept descriptions can be expressed by acyclic conjunctive queries with disequations [15], one might conjecture that polynomiality of subsumption in $\mathcal{EL}^{(n)}$ follows from the corresponding result for acyclic conjunctive queries with disequations. This is not true, however. In fact, the containment problem for conjunctive queries becomes considerably harder if disequations (i.e., atoms of the form $x \neq y$ for variables x, y) are allowed to occur in the conjunctive queries. For general conjunctive queries with disequations, the containment problem is Π_2^p-complete rather than NP-complete as in the case of conjunctive queries without disequations. Surprisingly, the problem remains Π_2^p-complete if Q_2 is restricted to being acyclic [15]. And even if both queries contain only disequations (and no database predicates), it is not hard to show by a reduction of the complement of the graph homomorphism problem that the containment problem is coNP-hard. Thus, the polynomiality result shown in the present paper does *not* follow from known results for containment of conjunctive queries with disequations.

In [7], it was shown that subsumption in $\mathcal{EL}$ remains polynomial even in the presence of GCIs, and this result was recently extended to a DL extending $\mathcal{EL}$ by several other interesting constructors [1]. Unfortunately, the results in [1] imply that subsumption in $\mathcal{EL}^{(n)}$ becomes ExpTime-hard in the presence of GCIs.

The most interesting topics for future research are, on the one hand, to show that the exponential translation from $\mathcal{EL}^{(n)}\mathcal{C}$ into $\mathcal{ALCQ}$ given in Section 3 is optimal, i.e., to prove that there is no polynomial translation. On the other hand, the exact complexity of subsumption between *unrestricted* $\mathcal{EL}^{(n)}$-concept descriptions is not yet known. The best complexity upper-bound that we currently have is coNP, i.e., there is an NP-algorithm for testing non-subsumption. We conjecture that the problem is also coNP-hard, but we have not yet found an appropriate reduction from a coNP-complete problem.

References

[1] F. Baader and S. Brandt and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, 2005.

[2] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German Workshop on Artificial Intelligence (GWAI'92)*, volume 671 of *LNCS*, 1992. Springer-Verlag.

[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] F. Baader, C. Lutz, E. Karabaev, and M. Theißen. A new n-ary existential quantifier in description logics. In *Proc. of the 28th German Conference on Artificial Intelligence (KI'05)*, 2005. To appear. A long version has appeared as LTCS-Report 05-08. See http://lat.inf.tu-dresden.de/research/reports.html.

[5] F. Baader, M. Milicic, C. Lutz, U. Sattler, and F. Wolter. Integrating description logics and action formalisms for reasoning about web services LTCS-Report 05-02, TU Dresden, Germany, 2005. See http://lat.inf.tu-dresden.de/research/reports.html.

[6] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[7] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. 16th Eur. Conf. on Artificial Intelligence (ECAI'04)*, 2004.

[8] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, 1994.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[10] F. Donini. Complexity of reasoning. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider (editors), *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[11] V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[12] P. Hall. On representatives of subsets. *The Journal of the London Mathematical Society*, 10:26–30, 1935.

[13] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.

[14] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *J. of the IGPL*, 8(3):239–264, 2000.

[15] P. G. Kolaitis, D. M. Martin, and M. N. Thakur. On the complexity of the containment problem for conjunctive queries with built-in predicates. In *Proc. of the 17th ACM Symp. on Principles of Database Systems (PODS'98)*, 1998.

[16] Carsten Lutz. Complexity of terminological reasoning revisited. In *Proc. 6th Int. Conf. on Logic for Programming and Automated Reasoning*, volume 1705 of *LNAI*. Springer-Verlag, 1999.

[17] X. Qian. Query folding. In *Proc. of the 12th IEEE Int. Conf. on Data Engineering (ICDE'96)*, 1996.

[18] S. W. Reyner. An analysis of a good algorithm for the subtree problem. *SIAM J. on Computing*, 6(4):730–732, 1977.

[19] U. Sattler. A concept language extended with different kinds of transitive roles. In *Proc. of the 20th German Annual Conf. on Artificial Intelligence (KI'96)*, volume 1137 of *LNAI*. Springer-Verlag, 1996.

[20] M. Theißen and L. von Wedel. The need for an n-ary existential quantifier in description logics. In *Proc. of the KI-04 Workshop on Applications of Description Logics*. http://CEUR-WS.org/Vol-115/, 2004.

[21] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, Department of Computer Science, RWTH Aachen, Germany, 2001.

[22] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of the 7th Int. Conf. on Very Large Data Bases (VLDB'81)*, 1981.