Rewriting Concepts Using Terminologies

Franz Baader and Ralf Molitor

LTCS-Report 99-06

Rewriting Concepts Using Terminologies

Franz Baader and Ralf Molitor
LuFg Theoretical Computer Science, RWTH Aachen
email: {baader,molitor}@informatik.rwth-aachen.de

Abstract

In this work we consider the inference problem of computing (minimal) rewritings of concept descriptions using defined concepts from a terminology. We introduce a general framework for this problem. For the small description logic $\mathcal{FL}_0$, which provides us with conjunction and value restrictions, we show that the decision problem induced by the minimal rewriting problem is NP-complete.

1 Motivation

Informally, the problem of rewriting a concept given a terminology can be stated as follows: given a TBox $\mathcal{T}$ and a concept description C that does not contain concept names defined in $\mathcal{T}$, can this description be rewritten into an equivalent "better" description D by using (some of) the names defined in $\mathcal{T}$? Better may mean shorter, but one can also imagine other optimality criteria. In the formal framework of rewriting introduced in Section 2 of this paper, we will not fix such an optimality criterion, and we will allow $\mathcal{T}$, C, and D to be built over different DLs. However, when instantiating this framework in Section 3, we will assume that $\mathcal{T}$, C, and D are built over the same DL $\mathcal{FL}_0$, and we will use the size of D as optimality criterion.

In the database area, the problem of rewriting queries using views is a well-known research topic [7]. It is closely related to the problem introduced in this paper since views can be regarded as TBox definitions and queries as concepts. However, our motivation for considering this new type of inference problem in DLs is quite different from the one in the DB area. There, one wants to optimize the runtime of queries by using cached views, and thus one wants to minimize the access to source relations. Our goal is to optimize the readability of concepts, and thus minimal length of the concept D appears to be a better optimality criterion.

More precisely, our interest in the rewriting problem stems from an application in chemical process engineering [6, 11]. Within this application, we try to support the bottom-up construction of KBs by computing most specific concepts (msc) of individuals and least common subsumers (lcs) of concepts: instead of directly defining a new concept, the knowledge engineer introduces several typical examples as individuals, which are then generalized into a concept description by using the msc and the lcs operation [4, 2]. This description is offered to the knowledge engineer as a possible candidate for a definition of the concept.

Unfortunately, due to the nature of the algorithms for computing the lcs and the msc proposed in [4, 2], these algorithms yield concept descriptions that do not contain defined concept names, even if the descriptions of the individuals used concepts defined in a TBox $\mathcal{T}$. In addition, due to the inherent complexity of the lcs and the msc operation, these descriptions may be quite large. To overcome this problem, we want to employ rewriting of the computed concept description using $\mathcal{T}$ in order to obtain a shorter and better readable description.

2 A general framework for rewriting

We first introduce the general framework for *rewriting using terminologies.*

Definition 1 (Rewriting) *Let N_r be a set of role names and N_p a set of* primitive *concept names, and let $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ be three DLs. A* rewriting problem *is given by*

- *an $\mathcal{L}_1$-TBox $\mathcal{T}$ containing only role names from N_r and primitive concept names from N_p; the set of concept names defined by concept definitions in $\mathcal{T}$ is denoted by N_d;*
- *an $\mathcal{L}_2$-concept description C using only the names from N_r and N_p.*

A rewriting of C using $\mathcal{T}$ *is an $\mathcal{L}_3$-concept description D built using names from N_r and $N_p \cup N_d$ such that C and D are equivalent modulo the TBox $\mathcal{T}$, i.e., $D^{\mathcal{I}} = C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$.*

Given an appropriate ordering $\preceq$ on $\mathcal{L}_3$-concepts built using names from N_r and $N_p \cup N_d$, a rewriting D is called $\preceq$-minimal *iff there does not exist a rewriting D' such that $D' \prec D$.*

It should be noted that there are cases in which there always exists a (trivial) rewriting; e.g., whenever $\mathcal{L}_1$ is a sublanguage of $\mathcal{L}_3$. However, in these cases finding a minimal rewriting is still a non-trivial task. In the next section, we will consider the case where all three DLs are equal to $\mathcal{FL}_0$, and where the ordering is induced by the size of (a normal form of) D.

Further, note that in contrast to inference tasks like subsumption or consistency, decidability results for the rewriting problem for DLs $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ cannot be transferred to sublanguages $\mathcal{L}_i'$ of $\mathcal{L}_i$, $1 \leq i \leq 3$. For example, $C = P \sqcup \neg P$ is a rewriting in $\mathcal{ALC}$ w.r.t. the empty TBox of itself, but there does not exist a rewriting of C in $\mathcal{FL}_0$ w.r.t. the empty TBox.

A comparison to *rewriting queries using views*

In the remainder of this section, we will analyze the results of Section 3 in [7] within the framework introduced above. There are two differences between the rewriting problem considered there and the one introduced above. First, [7] is concerned with *maximally contained* rewritings, i.e., the case where one wants to determine a maximal concept D subsumed by the input concept C. It should be noted, however, that there exists a rewriting in our sense iff the maximally contained rewriting is equivalent to C. Second, [7] is concerned with *total* rewritings (i.e., D may only use defined concept names), whereas we allow for *partial* rewritings (i.e., D may still contain primitive concepts). Section 3 of [7] contains the following two results:

- For $\mathcal{L}_1 = \mathcal{L}_2 = \mathcal{ALCNR}$[1] and $\mathcal{L}_3 = \{\sqcap, \sqcup\}$, a maximally contained total rewriting is computable. Using the subsumption algorithm for $\mathcal{ALCNR}$, this can be used to decide whether there exists a total rewriting equivalent to the input concept C.

- If $\mathcal{ALCNR}$ is replaced by $\mathcal{ALN}$[2], then one can compute a maximally contained total rewriting in polynomial time, and existence of a total rewriting equivalent to C can also be decided in polynomial time.

3 Rewriting in $\mathcal{FL}_0$

Under rewriting in $\mathcal{FL}_0$ we understand the instance of the framework introduced above where (i) all three DLs are the language $\mathcal{FL}_0$; (ii) the TBox is of the usual form (i.e., acyclic and without multiple definitions); and (iii) $\mathcal{FL}_0$-conept descritpions are ordered by their size.

Note that in this case, the problem wether there exists a rewriting of C w.r.t. $\mathcal{T}$ is trivially decidable, because C is always a rewriting of itself. So we are only interested in minimal rewritings.

We first recall syntax and semantics of $\mathcal{FL}_0$ as well as an appropriate characterization of subsumption for $\mathcal{FL}_0$. Thereafter, we will show that for a given arbitrary TBox the problem of determining a minimal rewriting is NP-complete.

Syntax and semantics

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 *primitive roles*. The constructors determine the expressive power of the DL. In the sequel, we consider $\mathcal{FL}_0$*-concept descriptions* built from the constructors *conjunction* $C \sqcap D$ and *value restriction* $\forall r.C$ (see Table 1).

The semantics of a concept description is defined in terms of an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$. The domain Δ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $P \in N_C$ to a set $P^I \subseteq \Delta$ and each primitive role $r \in N_R$ to a binary relation $r^I \subseteq \Delta \times \Delta$. The extension

[1]In addition to the constructors in $\mathcal{FL}_0$, $\mathcal{ALCNR}$ allows for *disjunction* $(C \sqcup D)$, *number restrictions* $((\geq n\, r)$ and $(\leq n\, r))$, and *role conjunction* $(r_1 \sqcap r_2)$.

[2]The sublanguage $\mathcal{ALN}$ of $\mathcal{ALCNR}$ only allows for conjunction, value restrictions and number restrictions.

Construct name	Syntax	Semantics
concept name $P \in N_C$	P	$P^I \subseteq \Delta$
conjunction	$C \sqcap D$	$C^I \cap D^I$
value restr. for $r \in N_R$	$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \to y \in C^I\}$

Table 1: Syntax and semantics of concept descriptions.

of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

In this paper, we are interested in the non-standard inference task of rewriting concept descriptions using a *TBox* $\mathcal{T}$.

Definition 2 (TBox) *A* concept definition *is of the form $A \doteq C$, where $A \in N_C$ is a concept name and C is a concept description. A* TBox *is a finite set $\mathcal{T}$ of concept definitions such that every concept name occurs at most once as left-hand side of a definition and there do not exist cyclic dependencies between concept definitions.*

The concept name A is a defined concept *in the TBox $\mathcal{T}$ iff it occurs on the left-hand side of a concept definition in $\mathcal{T}$; otherwise, A is called* primitive concept.

In the sequel, N_D denotes the set of concept names which are defined in $\mathcal{T}$, and N_P denotes the set of primitive concepts, i.e., $N_P = N_C \setminus N_D$.

The interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all concept definitions $A \doteq C \in \mathcal{T}$. A TBox $\mathcal{T}$ is called $\mathcal{FL}_0$-*TBox* if all concept descriptions occuring in the concept definitions in $\mathcal{T}$ are $\mathcal{FL}_0$-concept descriptions.

Subsumption in $\mathcal{FL}_0$

Definition 3 (Subsumption and equivalence) *Let C, D be concept descriptions, and let $\mathcal{T}$ be a TBox.*

D subsumes *C (for short $C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations $\mathcal{I}$.*
C is equivalent *to D (for short $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., $C^I = D^I$ for all interpretations $\mathcal{I}$.*

D subsumes *C* w.r.t. *$\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^I \subseteq D^I$ for all models $\mathcal{I}$ of $\mathcal{T}$.*
D is equivalent to *C* w.r.t. *$\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) iff $C^I = D^I$ for all models $\mathcal{I}$ of $\mathcal{T}$.*

For the DL $\mathcal{FL}_0$, a *concept-centered normal form* has turned out to be quite convenient for various purposes [5]: any $\mathcal{FL}_0$-concept description can be written in the form $\forall L_1.P_1 \sqcap \ldots \sqcap \forall L_k.P_k$, where $P_1, \ldots, P_k$ are concept names and the L_i are finite sets of words over the alphabet of primitive roles. This normal form can be obtained by

1. distributing value restrictions over conjunctions, i.e., by exhaustively applying the rule
$$\forall r.(D \sqcap E) \longrightarrow \forall r.D \sqcap \forall r.E;$$
2. writing $\forall r_1 \dots r_n.P_i$ instead of $\forall r_1. \cdots \forall r_n.P_i$; and finally
3. collecting the words w occurring in a value restrictions ending with P_i in the set L_i.

Example 4 Consider the $\mathcal{FL}_0$-concept description

$$P \sqcap \forall r.(\forall r.P \sqcap \forall r.Q).$$

The corresponding normal form is given by

$$\forall\{\varepsilon, rr\}.P \sqcap \forall\{rr\}.Q.$$

Using this normal form, *equivalence of $\mathcal{FL}_0$-concept descriptions* can be characterized as follows.

Theorem 5 *[5] Let C, D be $\mathcal{FL}_0$-concept descriptions with normal forms $C \equiv \forall L_1.P_1 \sqcap \dots \sqcap \forall L_k.P_k$ and $D \equiv \forall M_1.P_1 \sqcap \dots \sqcap \forall M_k.P_k$. Then $C \equiv D$ iff $L_i = M_i$ for $i = 1, \dots, k$.*

As an easy consequence we get that equivalence of $\mathcal{FL}_0$-concept descriptions C, D can be decided in time polynomial in the size of C and D.

In the presence of an $\mathcal{FL}_0$-TBox, however, the equivalence problem $C \equiv_{\mathcal{T}} D$ for $\mathcal{FL}_0$ becomes co-NP-complete [10]. Note that in this case C and D may contain primitive concepts as well as defined concepts. Thus, in order to test the condition from Theorem 5, one must first unfold the concept descriptions C, D w.r.t. $\mathcal{T}$ and then test the condition $L_C(P) = L_D(P)$ for all *primitive concepts* P. As shown in [10], unfolding an $\mathcal{FL}_0$-TBox $\mathcal{T}$ may yield a TBox $\mathcal{T}'$ of size exponential in the size of $\mathcal{T}$.

In our framework, we have to decide wether a concept description D is equivalent to C w.r.t. $\mathcal{T}$ where

1. C is an $\mathcal{FL}_0$-concept description of the form
$$\forall L_1.P_1 \sqcap \dots \sqcap \forall L_k.P_k,$$
containing only primitive concepts $P_i \in N_P$, and
2. D is an $\mathcal{FL}_0$-concept description of the form
$$D \equiv \forall M_1.A_1 \sqcap \dots \sqcap \forall M_\ell.A_\ell \sqcap \forall K_1.P_1 \sqcap \dots \sqcap \forall K_k.P_k,$$
where, w.l.o.g., $\{A_1, \dots, A_\ell\} = N_D$ and $\{P_1, \dots, P_k\} = N_P$.

$\mathcal{A}_\mathcal{T}$:

$$\begin{tikzcd}
A_1 \arrow[r, "r"] \arrow[dr, "\varepsilon"] & q_1 \arrow[dr, "r"] & \\
A_2 \arrow[r, "r"] & P_1 & P_2 \\
A_3 \arrow[u, "r"] \arrow[r, "r"] & q_2 \arrow[ur, "r"'] &
\end{tikzcd}$$

Figure 1: The finite automaton corresponding to $\mathcal{T}$.

It turned out that in this special case, the equivalence $C \equiv_\mathcal{T} D$ can be decided in time polynomial in the size of C, D, and $\mathcal{T}$. This tractability result is based on the automata-theoretic characterization of equivalence for $\mathcal{FL}_0$ given in [1]. The TBox $\mathcal{T}$ is translated into a finite automaton $\mathcal{A}_\mathcal{T}$ (with ε-transitions). The concept names in $\mathcal{T}$ are the states of $\mathcal{A}_\mathcal{T}$ and the transitions in $\mathcal{A}_\mathcal{T}$ are induced by the value restrictions in $\mathcal{T}$ (see [1] for details). For a defined concept A from $\mathcal{T}$ and a primitive concept P, the language $L_{\mathcal{A}_\mathcal{T}}(A,P)$ denotes the set of all words labeling paths from A to P in $\mathcal{A}_\mathcal{T}$. The languages $L_{\mathcal{A}_\mathcal{T}}(A,P)$ represent all value restrictions that must be satisfied by instances of the concept A, i.e., A can be equivalently written as

$$\forall L_{\mathcal{A}_\mathcal{T}}(A,P_1).P_1 \sqcap \ldots \sqcap \forall L_{\mathcal{A}_\mathcal{T}}(A,P_k).P_k.$$

Example 6 Consider the $\mathcal{FL}_0$-TBox

$$\mathcal{T} \;:=\; \{A_1 \doteq P_1 \sqcap \forall r.\forall r.P_2,\; A_2 \doteq \forall r.P_1,\; A_3 \doteq \forall r.\forall r.P_2 \sqcap \forall r.A_2\}.$$

The resulting automaton $\mathcal{A}_\mathcal{T}$ is depicted in Figure 1. The states q_1, q_2 are introduced as intermediate states on the paths from A_1 respectively A_3 to P_2. We have

$$\begin{array}{ll}
L_{\mathcal{A}_\mathcal{T}}(A_1,P_1)=\{\varepsilon, rr\} & L_{\mathcal{A}_\mathcal{T}}(A_1,P_2)=\{rr\} \\
L_{\mathcal{A}_\mathcal{T}}(A_2,P_1)=\{r\} & L_{\mathcal{A}_\mathcal{T}}(A_2,P_2)=\emptyset \\
L_{\mathcal{A}_\mathcal{T}}(A_3,P_1)=\{rr\} & L_{\mathcal{A}_\mathcal{T}}(A_3,P_2)=\{rr\}
\end{array}$$

Theorem 7 *Let $\mathcal{T}$ be an $\mathcal{FL}_0$-TBox, let $N_P = \{P_1,\ldots,P_k\}$ be the primitive concepts in $\mathcal{T}$, and $N_D = \{A_1,\ldots,A_\ell$ the defined concepts from $\mathcal{T}$. Further, let*

$$\begin{array}{rcl}
C &=& \forall L_1.P_1 \sqcap \ldots \sqcap \forall L_k.P_k, \\
D &=& \forall M_1.A_1 \sqcap \ldots \sqcap \forall M_\ell.A_\ell \sqcap \forall K_1.P_1 \sqcap \ldots \sqcap \forall K_k.P_k
\end{array}$$

be two $\mathcal{FL}_0$-concept descriptions in normal form.

1. *It holds that $C \equiv_\mathcal{T} D$ iff $L_C(P_i) = \bigcup_{1\le j\le \ell} M_j \cdot L_{\mathcal{A}_\mathcal{T}}(A_j,P_i) \cup K_i$ for all $1 \le i \le k$.*

2. *Deciding wether $C \equiv_\mathcal{T} D$ takes time polynomial in the size of C, D and $\mathcal{T}$.*

Proof: The characterization of equivalence is a direct consequence of the results in [1].

It remains to prove the complexity result. In order to decide $C \equiv_{\mathcal{T}} D$, we have to decide $L_C(P_i) = \bigcup_{1\leq j\leq \ell} M_j \cdot L_{\mathcal{A}_\mathcal{T}}(A_j, P_i) \cup K_i$ for all $1 \leq i \leq k$. Obviously, it is sufficient to show that we can test validity of the equation for a fixed primitive concept P_i in polynomial time.

By the results in [3] we get that for each $\mathcal{FL}_0$-concept description of the form $\forall L_C(P_i).P_i$ there exists a *deterministic* finite automaton $\mathcal{A}_C$ such that (a) the size of $\mathcal{A}_C$ is polynomial in the size of $L_C(P_i)$; and (b) $L(\mathcal{A}_C) = L_C(P_i)$.

In order to complete the proof, we now define a (non-deterministic) automaton $\mathcal{A}$ of size polynomial in the size of D and $\mathcal{T}$ with $L(\mathcal{A}) = L := \bigcup_{1\leq j\leq \ell} M_j \cdot L_{\mathcal{A}_\mathcal{T}}(A_j, P_i) \cup K_i$.

Let $\mathcal{A}_\mathcal{T} = (Q_\mathcal{T}, N_R, \Delta_\mathcal{T})$ be the automaton corresponding to $\mathcal{T}$ with ε-transitions. We first define an automaton $\mathcal{A}^* := (Q, N_R, q_0, \Delta, F)$ with ε-transitions as follows. Let q_0 be a new state, i.e., $q_0 \notin Q_\mathcal{T}$. Let $F := \{P_i\}$. Q and Δ are obtained from $Q_\mathcal{T}$ respectively $\Delta_\mathcal{T}$ as follows. For each j, if $\varepsilon \in M_j$, then introduce the transition (q_0, ε, A_j); for each word $r_1 \ldots r_n \in M_j$, $n \geq 1$, introduce $n-1$ new states $q_1, \ldots, q_{n-1}$ and transitions $(q_{\nu-1}, r_\nu, q_\nu)$ for $1 \leq \nu < n$, and (q_{n-1}, r_n, A_j). Further, if $\varepsilon \in K_i$, then introduce the transition (q_0, ε, P_i); for each word $r_1 \ldots r_n$, $n \geq 1$, introduce $n-1$ new states $q_1, \ldots, q_{n-1}$ and transitions $(q_{\nu-1}, r_\nu, q_\nu)$ for $1 \leq \nu < n$, and (q_{n-1}, r_n, P_i).

Now, let $\mathcal{A}$ be the non-deterministic finite automaton without ε-transitions obtained from $\mathcal{A}^*$. The automaton $\mathcal{A}^*$ has size polynomial in the size of M_j, $1 \leq j \leq \ell$; K_i; and $\mathcal{A}_\mathcal{T}$. So, the size of $\mathcal{A}^*$, and hence of $\mathcal{A}$, is polynomial in the size of D and $\mathcal{T}$. Further, it is not hard to see that $L(\mathcal{A}^*) = L$, and thus, $L(\mathcal{A}) = L$.

Using the automata $\mathcal{A}$ and $\mathcal{A}_C$, we can decide validity of the equation $L_C(P_i) = \bigcup_{1\leq j\leq \ell} M_j \cdot L_{\mathcal{A}_\mathcal{T}}(A_j, P_i) \cup K_i$ as follows:

1. Testing $L_C(P_i) \subseteq L(\mathcal{A})$ can be done in polynomial time by just testing $w \in L(\mathcal{A})$ for all $w \in L_C(P_i)$.

2. Conversely, testing $L(\mathcal{A}_C) \supseteq L(\mathcal{A})$ is reduced to testing $\overline{L(\mathcal{A}_C)} \cap L(\mathcal{A}) = \emptyset$. Since $\mathcal{A}_C$ is a deterministic automaton, an automaton $\mathcal{B}$ with $L(\mathcal{B}) = \overline{L(\mathcal{A}_C)} \cap L(\mathcal{A})$ can be determined in polynomial time. For a finite automaton $\mathcal{B}$ the emptiness problem can be decided in time polynomial in the size of $\mathcal{B}$.

This shows that equivalence of C and D w.r.t. $\mathcal{T}$ can be decided in polynomial time. □

Rewriting as a formal language problem

With the help of the automata-theoretic characterization of subsumption, the *rewriting problem in $\mathcal{FL}_0$* can be translated into a *formal language problem.* Let $(C, \mathcal{T})$ be an instance of the $\mathcal{FL}_0$ rewriting problem, let $P_1, \ldots, P_k$ be the available primitive concepts, and let $A_1, \ldots, A_\ell$ be the concept names defined in

$\mathcal{T}$. We assume that C has the normal form $C \equiv \forall L_1.P_1 \sqcap \ldots \sqcap \forall L_k.P_k$, and that the finite automaton $\mathcal{A}_\mathcal{T}$ corresponding to $\mathcal{T}$ is defined as described before, i.e., we have $A_j \equiv \forall L_{\mathcal{A}_\mathcal{T}}(A_j, P_1).P_1 \sqcap \ldots \sqcap \forall L_{\mathcal{A}_\mathcal{T}}(A_j, P_k).P_k$. Then, the $\mathcal{FL}_0$-concept description D is a rewriting of C using the TBox $\mathcal{T}$ iff its normal form is of the form

$$D \equiv \forall M_1.A_1 \sqcap \ldots \sqcap \forall M_\ell.A_\ell \sqcap \forall K_1.P_1 \sqcap \ldots \sqcap \forall K_k.P_k,$$

where the assignment $X_1 := M_1, \ldots, X_\ell := M_\ell, Y_1 := K_1, \ldots, Y_k := K_k$ solves the system of formal language equations

$$L_i \quad = \quad X_1 \cdot L_{\mathcal{A}_\mathcal{T}}(A_1, P_i) \cup \ldots \cup X_\ell \cdot L_{\mathcal{A}_\mathcal{T}}(A_\ell, P_i) \cup Y_i \quad (1 \leq i \leq k). \qquad (1)$$

Example 8 (Example 4 and 6 continued) Consider the $\mathcal{FL}_0$-concept description C defined in Example 4 and the $\mathcal{FL}_0$-TBox $\mathcal{T}$ introduced in Example 6 again.

The rewriting problem $(C, \mathcal{T})$ translates into the system of formal language equations

$$\begin{aligned} \{\varepsilon, rr\} &= X_1 \cdot \{\varepsilon\} \cup X_2 \cdot \{r\} \cup X_3 \cdot \{rr\} \cup Y_1, \\ \{rr\} &= X_1 \cdot \{rr\} \cup X_2 \cdot \emptyset \cup X_3 \cdot \{rr\} \cup Y_2. \end{aligned}$$

It is easy to see that the assignment $X_1 := \{\varepsilon\}$, $X_2 := \{r\}$ and $X_3 := Y_1 := Y_2 := \emptyset$ solves this system. This solution yields the rewriting $D := A_1 \sqcap \forall r.A_2$.

There are also other solutions of the system; e.g., $X_1 := X_2 := X_3 := \emptyset$ and $Y_1 := \{\varepsilon, rr\}$, $Y_2 := \{rr\}$ is also a solution, which yields the trivial rewriting $D' := C$.

If we compare the two rewritings D and D' given in Example 8, then intuitively, D is a better rewriting than D' since it is shorter, and thus better to read and comprehend. This leads us to the definition of minimal rewritings. There are different partial orderings on $\mathcal{FL}_0$-concept descriptions that are appropriate for measuring the quality of a rewriting w.r.t. size and readability.

Definition 9 (Partial orderings on $\mathcal{FL}_0$-concept descriptions) *Let C, D be $\mathcal{FL}_0$-concept descriptions. Further, let $C' = \forall L_1.A_1 \sqcap \ldots \sqcap \forall L_m.A_m \sqcap \forall M_1.P_1 \sqcap \ldots \sqcap \forall M_n.P_n$ be the normal form of C where the A_j denote the defined concepts and the P_i the primitive concepts in C.*

First, we define the size of a set L of words as $\|L\| := \sum_{w \in L}(|w| + 1)$ where $|w|$ denotes the length of the word w.[3]

Now, we define different sizes $\|C\|_1 \in \mathbb{N}$ and $\|C\|_2 \in \mathbb{N} \times \mathbb{N}$ as follows:

1. $\|C\|_1 := \|C'\|_1 := \sum_{1 \leq i \leq n} \|L_i\|$
2. $\|C\|_2 := \|C'\|_2 := (\sum_{1 \leq i \leq n} \|M_i\|, \sum_{1 \leq i \leq m} \|L_i\|)$

Further, we define

[3]The $+1$ for each $w \in L$ corresponds to the size of the concept name occuring in the value restriction $\forall L.P$ in C'.

1. *$C \leq_1 D$ iff $\|C\|_1 \leq \|D\|_1$ where $\leq$ denotes the linear ordering on $\mathbb{N}$, and*
2. *$C \leq_2 D$ iff $\|C\|_2 \preceq \|D\|_2$ where $\preceq$ denotes the lexicographic ordering on $\mathbb{N} \times \mathbb{N}$.*

If one is interested in a rewriting that is as small as possible, then one must compare concept descriptions by $\leq_1$. Comparing concept descriptions by $\leq_2$ assures that the number of occurences of primitive concepts in a minimal rewriting is minimal and hence the reuse of defined concepts is maximal. In other words, whenever there exists a total rewriting, the minimal rewriting is total (because for a total rewriting D, the first component in $\|D\|_2$ is 0).

The following example shows that the two orderings $\leq_1$ and $\leq_2$ yield different minimal rewritings D_1, D_2. Here, *different* means that the concept descriptions D_1 and D_2 are *not* equivalent (though they are still equivalent w.r.t. $\mathcal{T}$).

Example 10 (Example 4 and 6 continued) Let C and $\mathcal{T}$ be defined as in Example 4 and Example 6, respectively.

Now, consider the $\mathcal{FL}_0$-concept descriptions $D_1 := P \sqcap A_3$ and $D_2 := A_1 \sqcap \forall r.A_2$.

It is not hard to see that D_i is a rewriting of C using $\mathcal{T}$ that is minimal w.r.t. $\leq_i$ $i = 1, 2$.

But D_1 is not a minimal rewriting w.r.t. $\leq_2$, because $\|D_2\|_2 = (0, 2) \preceq (1, 1) = \|D_1\|_2$.

Also, D_2 is not a minimal rewriting w.r.t. $\leq_1$, because $\|D_1\|_1 = 2 \leq 3 = \|D_2\|_1$.

We now show that the decision problem induced by the minimal rewriting problem is NP-complete (for both orderings introduced in Definition 9). NP-hardness is shown using a straight forward reduction of the NP-complete problem SETCOVER [9].

Theorem 11 *Let C be an $\mathcal{FL}_0$-concept description and $\mathcal{T}$ an $\mathcal{FL}_0$-TBox. Further, let $\kappa \in \mathbb{N}$ and $(\kappa_1, \kappa_2) \in \mathbb{N} \times \mathbb{N}$.*

Deciding wether there exists a rewriting D of C using $\mathcal{T}$ with size $\|D\|_1 \leq \kappa$ respectively $\|D\|_2 \preceq (\kappa_1, \kappa_2)$ is NP-complete.

Outline of the proof We will prove Theorem 11 as follows: First, we reduce the problem of solving the k formal language equations (1) to solving a single formal language equation (cf. Lemma 12). Then, we will determine a maximal solution of the single equation (cf. Lemma 13). Unfortunately, determining the maximal solution may take time exponential in the size of the TBox. Thus, in order to obtain a non-deterministic *polynomial* algorithm, we introduce a set of possible solutions that (a) contains all solutions (in particular, the maximal solution), and (b) allows for a polynomial, non-deterministic step for choosing a possible solution. For such a possible solution, deciding wether it is a solution of size $\leq \kappa$ resp. $\preceq (\kappa_1, \kappa_2)$ takes time polynomial in the size of C and $\mathcal{T}$. Hence, we obtain an NP-decision algorithm.

In order to show that the problem is NP-hard, we will give a polynomial reduction of the NP-complete problem SETCOVER to the minimal rewriting decision problem.

Proof that the problem is in NP For a language L and a word w we define $L \cdot w := \{vw \mid v \in L\}$ and $L \cdot w^{-1} := \{v \mid vw \in L\}$. For two languages L_1, L_2 we define $L_1 \cdot L_2 := \{vw \mid v \in L_1 \text{ and } w \in L_2\}$.

The system of formal language equations (1) can be transformed equivalently into a single equation as follows: Let $R_1, \ldots, R_k$ be new role names not occuring in C and $\mathcal{T}$. For convenience, let

$$\begin{aligned} S_0 &:= \bigcup_{1\le i\le n} L_C(P_i) \cdot R_i \text{ and} \\ S_j &:= \bigcup_{1\le i\le n} L_{A_j}(P_i) \cdot R_i \text{ for } 1 \le j \le m. \end{aligned}$$

Using these abbreviations, we can rewrite the system of equations (1) by the single equation

$$S_0 = \bigcup_{1\le j\le \ell} (X_j \cdot S_j) \cup \bigcup_{1\le i\le k} (Y_i \cdot R_i). \tag{2}$$

Lemma 12 *The assignment $X_1 := M_1, \ldots, X_\ell := M_\ell, Y_1 := K_1, \ldots, Y_k := K_k$ solves the system of formal language equations (1) iff it solves the formal language equation (2).*

Proof: "$\Longrightarrow$" Let $(M_j)_{1\le j\le m}, (K_i)_{1\le i\le k}$ be a solution of (1). We have

$$L_C(P_i) = \bigcup_{1\le j\le m} M_j \cdot L_{A_j}(P_i) \cup K_i$$

for all $1 \le i \le k$. This implies

$$L_C(P_i) \cdot R_i = (\bigcup_{1\le j\le m} M_j \cdot L_{A_j}(P_i) \cup K_i) \cdot R_i$$

for all $1 \le i \le k$ and hence

$$\bigcup_{1\le i\le k} L_C(P_i) \cdot R_i = \bigcup_{1\le i\le k} (\bigcup_{1\le j\le m} M_j \cdot L_{A_j}(P_i)) \cdot R_i \cup \bigcup_{1\le i\le k} K_i \cdot R_i.$$

It remains to show

$$\bigcup_{1\le i\le k} (\bigcup_{1\le j\le m} M_j \cdot L_{A_j}(P_i)) \cdot R_i = \bigcup_{1\le j\le m} M_j \cdot \bigcup_{1\le i\le k} L_{A_j}(P_i)) \cdot R_i.$$

"$\subseteq$" Let $w \in \bigcup_{1\le i\le k} (\bigcup_{1\le j\le m} M_j \cdot L_{A_j}(P_i)) \cdot R_i$. There exist i_0, j_0 and $v_1 \in M_{j_0}$, $v_2 \in L_{A_{j_0}}(P_{i_0})$ such that $w = v_1 v_2 R_{i_0}$. This implies

$$\begin{aligned} w &\in M_{j_0} \cdot L_{A_{j_0}}(P_{i_0}) \cdot R_{i_o} \\ &\subseteq M_{j_0} \cdot \bigcup_{1\le i\le k} L_{A_j}(P_i)) \cdot R_i \\ &\subseteq \bigcup_{1\le j\le m} M_j \cdot \bigcup_{1\le i\le k} L_{A_j}(P_i)) \cdot R_i. \end{aligned}$$

"$\supseteq$" Let $w \in \bigcup_{1\leq j\leq m} M_j \cdot \bigcup_{1\leq i\leq k} L_{A_j}(P_i)) \cdot R_i$. There exist i_0, j_0 and $v_1 \in M_{j_0}$, $v_2 \in L_{A_{j_0}}(P_{i_0})$ such that $w = v_1 v_2 R_{i_0}$. This implies

$$\begin{array}{rcl} w & \in & M_{j_0} \cdot L_{A_{j_0}}(P_{i_0}) \cdot R_{i_o} \\ & \subseteq & (\bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_{i_0})) \cdot R_{i_0} \\ & \subseteq & \bigcup_{1\leq i\leq k}(\bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_i)) \cdot R_i. \end{array}$$

"$\Longleftarrow$" We show that if $(M_j)_j$, $(K_i)_i$ is *not* a solution of (1), then it is also *not* a solution of (2).

Assume $(M_j)_j$, $(K_i)_i$ is not a solution of (1), i.e., there exists $1 \leq i_0 \leq k$ such that $L_{i_0} \neq \bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_{i_0})$. We have to consider two cases:

1. $w \in L_{i_0}$ and $w \notin \bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_{i_0}) \cup K_{i_0}$. Then we have $wR_{i_0} \in \bigcup_{1\leq i\leq n} L_C(P_i) \cdot R_i$, but

$$\begin{array}{rcl} w & \notin & \bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_{i_0}) \cup K_{i_0} \\ \Longrightarrow wR_{i_0} & \notin & \bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_{i_0}) \cup K_{i_0}) \cdot R_{i_0} \\ \Longrightarrow wR_{i_0} & \notin & \bigcup_{1\leq i\leq k}(\bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_i) \cup K_i) \cdot R_i, \end{array}$$

   because $R_{i_0} \neq R_i$ for $i \neq i_0$ and $R_i \notin \Sigma$ for all i. In the first part of the proof, we have already shown that $\bigcup_{1\leq i\leq k}(\bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_i)) \cdot R_i = \bigcup_{1\leq j\leq m} M_j \cdot \bigcup_{1\leq i\leq k} L_{A_j}(P_i)) \cdot R_i$. This implies that $(M_j)_j$, $(K_i)_i$ is also not a solution of (2).

2. $w \notin L_{i_0}$ and $w \in \bigcup_{1\leq j\leq m} M_j \cdot L_{A_j}(P_{i_0}) \cup K_{i_0}$. As before, it follows $wR_{i_0} \in \bigcup_{1\leq j\leq m} M_j \bigcup_{1\leq i\leq k}(L_{A_j}(P_i) \cup K_i) \cdot R_i)$, but $wR_{i_0} \notin L_{i_0}R_{i_0}$. Further, $wR_{i_0} \notin L_{i_0}R_{i_0}$, because $R_{i_0} \neq R_i$ for $i \neq i_0$ and $R_i \notin \Sigma$ for all i. Hence, $wR_{i_0} \notin \bigcup_{1\leq i\leq k} L_i \cdot R_i$, and $(M_j)_j$, $(K_i)_i$ is not a solution of (2). □

Lemma 13 *For a given $\mathcal{FL}_0$-concept description C in normal form and an $\mathcal{FL}_0$-TBox $\mathcal{T}$ let S_0 and S_j, $1 \leq j \leq \ell$ be defined as before. Further, let*

$$\begin{array}{rcl} M_j & := & \bigcap_{w\in S_j} (S_0 \cdot w^{-1}) \text{ for } 1 \leq j \leq \ell, \text{ and} \\ K_i & := & L_C(P_i) \text{ for } 1 \leq i \leq k. \end{array}$$

The assignment $X_j := M_j$ for $1 \leq j \leq \ell$ and $Y_i := K_i$ for $1 \leq i \leq k$

1. *satisfies the formal language equation (2), and*

2. *for each solution $M'_1, \ldots, M'_\ell, K'_1, \ldots, K'_k$ of (2) we have $M'_j \subseteq M_j$ for all $1 \leq j \leq \ell$ and $K'_i \subseteq K_i$ for all $1 \leq i \leq k$.*

Proof:

1. By definition of M_j, $1 \leq j \leq \ell$, and K_i, $1 \leq i \leq k$, we have
$$S_0 \supseteq \bigcup_{1 \leq j \leq \ell} (M_j \cdot S_j) \cup \bigcup_{1 \leq i \leq k} (K_i \cdot R_i).$$
By definition of K_i, $1 \leq i \leq k$, we have
$$\begin{aligned} S_0 &\subseteq \bigcup_{1 \leq i \leq k} (K_i \cdot R_i) \\ &\subseteq \bigcup_{1 \leq j \leq \ell} (M_j \cdot S_j) \cup \bigcup_{1 \leq i \leq k} (K_i \cdot R_i). \end{aligned}$$

2. Assume that there exists w' such that $w' \in M'_j \backslash M_j$ for some $j \in \{1, \ldots, \ell\}$. We have $M_j = \bigcap_{w \in S_j} S_0 \cdot w^{-1}$. So, $w' \notin M_j$ implies that there exists $w \in S_j$ such that $w' \notin S_0 \cdot w^{-1}$. Let $w = w_1 R_i$, $R_i \in \{R_1, \ldots, R_k\}$. We get $w' w_1 R_i \notin S_0$. This is a contradiction to $S_0 = \bigcup_{1 \leq j \leq \ell} (M_j \cdot S_j) \cup \bigcup_{1 \leq i \leq k} (K_i \cdot R_i)$.

   Finally, assume that $K'_i \not\subseteq K_i$ for some $i \in \{1, \ldots, k\}$. As before, this yields a contradiction to $S_0 = \bigcup_{1 \leq j \leq \ell} (M_j \cdot S_j) \cup \bigcup_{1 \leq i \leq k} (K_i \cdot R_i)$. □

Note that in order to determine the maximal solution for a given concept description C and a given TBox $\mathcal{T}$, the sets $L_{\mathcal{A}_\mathcal{T}}(A_j, P_i)$ must be explicitly computed, i.e., the TBox must be unfolded. Since the unfolded TBox may be of size exponential in the size of $\mathcal{T}$ [10], the resulting languages may be of exponential size.

So since we are interested in a non-deterministic *polynomial* algorithm, we cannot use the maximal solution as a starting point for guessing a possible solution. In order to ensure that this first step takes time polynomial in the size of C and $\mathcal{T}$, we have to choose an appropriate set of possible solutions. In the second step, we then have to test wether the possible solution really is a solution of size $\leq k$ ($\preceq (k_1, k_2)$, respectively).

Consider the maximal solution again. We have

$$M_j \subseteq \{v \in N_R^* \mid \text{ exists } w \in N_R^* : vw \in S_0\} \quad (1 \leq j \leq \ell), \tag{3}$$
$$K_i \subseteq \{v \in N_R^* \mid \text{ exists } w \in N_R^* : vw \in S_0\} \quad (1 \leq i \leq k). \tag{4}$$

For convenience, let

$$\mathcal{X} := \{v \in N_R^* \mid \text{ exists } w \in N_R^* : vw \in S_0\}$$

be the set of all prefixes of words in S_0. Obviously, the size of $\mathcal{X}$ is polynomial in the size of S_0 and hence in the size of C. Thus, we can guess a possible solution $(M'_1, \ldots, M'_\ell, K'_1, \ldots, K'_k)$ where $M'_j, K'_i \subseteq \mathcal{X}$ for all $1 \leq j \leq \ell$ and $1 \leq i \leq k$ in polynomial time by just deciding for each M'_j (K'_i) and each $w \in \mathcal{X}$ wether $w \in M'_j$ ($w \in K'_i$) or not. By Lemma 13 and (3) we get that each solution can be obtained this way.

The *non-deterministic* decision algorithm works as follows:

1. Guess the sets $M'_j \subseteq \mathcal{X}$, $K'_i \subseteq \mathcal{X}$, i.e., determine a possible solution.

2. Test wether the $\mathcal{FL}_0$-concept description induced by M'_j, K'_i is equivalent to C w.r.t. $\mathcal{T}$.

3. Test wether the $\mathcal{FL}_0$-concept description induced by M'_j, K'_i has size $\leq \kappa$ (resp. $\preceq (\kappa_1, \kappa_2)$).

Return "yes", which means that there exists a rewriting with size $\leq \kappa$ ($\preceq (\kappa_1, \kappa_2)$), if there exists a successful computation for the steps 1–3, i.e., there exist sets M'_j, K'_i such that the induced $\mathcal{FL}_0$-concept description is equivalent to C w.r.t. $\mathcal{T}$ and has size $\leq \kappa$ ($\leq (\kappa_1, \kappa_2)$). Otherwise, return "no".

Obviously, this algorithm answers "yes" iff there exists a rewriting D of C using $\mathcal{T}$ with $\|D\| \leq \kappa$ ($\leq (\kappa_1, \kappa_2)$). It remains to show that the steps 1–3 can be executed in polynomial time.

The first step can be executed in polynomial time, because m, k, and $\mathcal{X}$ are polynomial in the size of C and $\mathcal{T}$. By Theorem 7 we get that $C \equiv_{\mathcal{T}} D$ can be decided in time polynomial in the size of C and $\mathcal{T}$. Finally, the size of the induced $\mathcal{FL}_0$-concept description D is polynomial in the size of C and $\mathcal{T}$, so $\|D\|_1 \leq \kappa$ ($\|D\|_2 \preceq (\kappa_1, \kappa_2)$) can also be decided in polynomial time.

Proof of NP-hardness We will use a reduction of the NP-complete problem SETCOVER. An instance of the SETCOVER problem is of the following form [9]:

Given: A finite set $\mathcal{U} = \{u_1, \ldots, u_n\}$, a family $\mathcal{F}$ of subsets of $\mathcal{U}$, $\mathcal{F} = \{F_i \subseteq \mathcal{U} \mid 1 \leq i \leq m\}$, and a number $k \in \mathbb{N}$.

Question: Does there exist a subset $\{F_{i_1}, \ldots, F_{i_k}\}$ of $\mathcal{F}$ of size $\leq k$ such that $F_{i_1} \cup \ldots \cup F_{i_k} = \mathcal{U}$?

Obviously, we can restrict our attention to instances of the problem where (a) at least $\mathcal{F}$ yields a covering of $\mathcal{U}$, i.e., $F_1 \cup \ldots \cup F_n = \mathcal{U}$, and (b) $k \leq n$.

For such an instance $(\mathcal{U}, \mathcal{F}, k)$ of the SETCOVER problem, we consider $\mathcal{U}$ as a set of role names and define an $\mathcal{FL}_0$-concept description by $C_{\mathcal{U}} := \forall \mathcal{U}.P$. Further, we define an $\mathcal{FL}_0$-TBox by $\mathcal{T}_{\mathcal{F}} := \{A_j \doteq \forall F_j.P \mid 1 \leq j \leq m\}$, where P denotes a single primitive concept. Obviously, C and $\mathcal{T}$ are polynomial in the size of $(\mathcal{U}, \mathcal{F}, k)$.

We will now show that the following equivalences hold:

1. There exists a minimal rewriting D of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$ with $\|D\|_1 \leq k$ iff there exists a covering of U with k sets $F_{i_1}, \ldots, F_{i_k}$ from $\mathcal{F}$.

2. There exists a minimal rewriting D of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$ with $\|D\|_2 \preceq (0, k)$ iff there exists a covering of $\mathcal{U}$ with k sets $F_{i_1}, \ldots, F_{i_k}$ from $\mathcal{F}$.

The maximal rewriting of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$ is of the form $D = \forall\{\varepsilon\}.A_1 \sqcap \ldots \sqcap \forall\{\varepsilon\}.A_m \sqcap \forall \mathcal{U}.P$. Hence, each rewriting of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$ is of the form $D' =$

$\forall M_1.A_1 \sqcap \ldots \sqcap \forall M_m.A_m \sqcap \forall K.P$ such that $M_j = \emptyset$ or $M_j = \{\varepsilon\}$, $K \subseteq \mathcal{U}$, and $M_1 \cup \ldots \cup M_m \cup K = \mathcal{U}$.

Assume that $F_{i_1}, \ldots, F_{i_k}$ is a covering of $\mathcal{U}$ of size k. Without loss of generality let $i_j = j$ for $1 \leq j \leq k$. Then $D' := A_1 \sqcap \ldots \sqcap A_k$ is a rewriting of $C_{\mathcal{U}}$, because $L_{C_{\mathcal{U}}}(P) = \mathcal{U} = F_1 \cup \ldots \cup F_k = L_{A_1}(P) \cup \ldots \cup L_{A_k}(P)$. Further, we have $\|D'\|_1 = k$ and $\|D'\|_2 = (0, k)$.

Conversely, assume that D is a rewriting of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$, $D = \bigsqcap\limits_{1 \leq i \leq m} \forall M_i.A_i \sqcap \forall K.P$.

1. Let D be minimal w.r.t. $\|\cdot\|_1$ and $\|D\|_1 \leq k$. We show that $M = \emptyset$. This implies that D is of the form $A_{i_1} \sqcap \ldots \sqcap A_{i_k}$ and hence, $\{F_{i_1}, \ldots, F_{i_k}\}$ is a covering of $\mathcal{U}$ of size $\leq k$.

   Assume $K \neq \emptyset$. Let $w \in K$. Then $w \notin \bigcup_{M_i = \{\varepsilon\}} F_i$. Otherwise, $D' := \bigsqcap_{1 \leq i \leq m} \forall M_i.A_i \sqcap \forall(K \setminus \{w\}).P$ would also be a rewriting of $C_{\mathcal{U}}$ with $\|D'\|_1 = \|D\|_1 - 2 < \|D\|_1$ in contradiction to the minimality of D.

   Now, let $E := \{F_i \in \mathcal{F} \mid M_i = \emptyset\}$. Then, for all $F_j \in E$ we have $w \notin F_j$. Otherwise, $D' := \bigsqcap_{1 \leq i \leq m} \forall M_i.A_i \sqcap A_j \sqcap \forall(K \setminus \{w\}).P$ would also be a rewriting of $C_{\mathcal{U}}$ with $\|D'\|_1 = \|D\|_1 + 1 - 2 < \|D\|_1$ in contradiction to the minimality of D.

   So, $w \notin \bigcup_{M_i = \{\varepsilon\}} F_i \cup \bigcup_{M_i = \emptyset} F_i = \bigcup_{1 \leq i \leq m} F_i$. Since $w \in L_D(P)$ and by assumption $\bigcup_{1 \leq i \leq m} F_i = \mathcal{U} = L_C(P)$, this yields a contradiction to $C \equiv_{\mathcal{T}} D$.

   Thus, we get $K = \emptyset$ and hence, $D = A_{i_1} \sqcap \ldots \sqcap A_{i_k}$.

2. Let D be minimal w.r.t. $\|\cdot\|_2$ and $\|D\|_2 \preceq (0, k)$.

   By definition of $\|\cdot\|_2$ we get that $K = \emptyset$. Hence, $D = A_{i_1} \sqcap \ldots \sqcap A_{i_k}$, and $\{F_{i_1}, \ldots, F_{i_k}\}$ is a covering of $\mathcal{U}$ of size $\leq k$.

This completes the proof of NP-hardness of the decision problem induced by the minimal rewriting problem, and hence the proof of Theorem 11. □

It should be noted that, in the formulation of Theorem 11, we do not assume that the TBox $\mathcal{T}$ is unfolded. Since it is well-known [10] that the equivalence problem w.r.t. (not unfolded) $\mathcal{FL}_0$-TBoxes is a co-NP-complete problem, one might conjecture that this is the source of complexity for the rewriting problem. This is not true, however: on the one hand, the reduction of SETCOVER to the rewriting problem generates unfolded TBoxes; on the other hand, our NP-algorithm is based on the fact that testing whether a candidate rewriting D is equivalent to C can be realized in polynomial time, even if $\mathcal{T}$ is not assumed to be unfolded.

Consider the proof of NP-hardness again. The reduction of an instance $(\mathcal{U}, \mathcal{F}, k)$ of the NP-complete problem SETCOVER is based on representing (i) the set $\mathcal{U}$ by the concept description $C_{\mathcal{U}}$ that has to be rewritten, and (ii) the family of subsets $\mathcal{F}$ by defined concepts in the TBox $\mathcal{T}_{\mathcal{F}}$. Consequently, in the resulting minimal rewriting problem, the size of the concept description as well as the TBox variates in the size of $(\mathcal{U}, \mathcal{F}, k)$. Moreover, if we assume the TBox to be fixed, then this reduction cannot be applied.

So far, we could not determine the exact complexity of the minimal rewriting decision problem *for a fixed TBox.* Obviously, the NP-algorithm also works in the case of a fixed TBox, i.e., the problem is in NP. But we could neither prove that in this restricted case the problem becomes tractable, nor that it is still NP-hard.

Minimal rewritings w.r.t. the role based normal form

In Theorem 11 we consider $\mathcal{FL}_0$-concept descriptions in the concept-based normal form. For a given $\mathcal{FL}_0$-concept description C the corresponding concept-based normal form C_c can be obtained by exhaustively applying the following rule:

$$\forall r.(D \sqcap E) \longrightarrow \forall r.D \sqcap \forall r.E.$$

If we apply the above rule from right to left, we obtain a *role-based normal form* C_r of C (as known from the structural subsumption algorithm used in the CLASSIC-system [8]). The *size* of a concept description in role-based normal form is given by the sum of the number of $\forall$-constructors and the number of occurences of concept names.[4]

For example, the role-based normal form of the $\mathcal{FL}_0$-concept C from our example is given as $C_r := P \sqcap \forall r.\forall r.(P \sqcap Q)$, and $\|C_r\| = 5$.

A comparison of the concept-based normal form C_c and the role-based normal form C_r reveals that common prefixes occuring in languages $L_{C_c}(P)$ and $L_{C_c}(Q)$ are shared in C_r. In our example, the subconcepts $\forall r.\forall r.P$ and $\forall r.\forall r.Q$ yield the word rr in $L_{C_c}(P)$ and $L_{C_c}(Q)$ and hence, the $\forall$-constructors are counted twice in $\|C_c\|$. In C_r, however, there is only one subconcept beginning with $\forall r.\forall r.\cdot$ and hence, the $\forall$-constructors are counted only once in $\|C_r\|$. In general, it is $\|C_r\| \leq \|C_c\|$ for all $\mathcal{FL}_0$-concept descriptions C.

For the minimal rewriting problem, this means that we have to distinguish between concept-based minimal rewritings, i.e., minimal rewritings in a concept-based normal form, and those in role-based normal form. More precisely, it turned out that the role-based normal form of a concept-based minimal rewriting in general is *not* a role-based minimal rewriting and vice versa.

Example 14 Consider the concept description

$$C := \forall r.\forall r.\forall r.(P_1 \sqcap P_2 \sqcap P_3 \sqcap P_4 \sqcap \forall r.(P_1 \sqcap P_2)),$$

and the TBox

$$\begin{aligned} \mathcal{T} \quad := \quad & \{A_1 \doteq \forall r.\forall r.P_1, A_2 \doteq \forall r.\forall r.P_2, A_3 \doteq P_3 \sqcap P_4, \\ & A_4 \doteq P_1 \sqcap P_2 \sqcap \forall r.P_1, A_5 \doteq \forall r.P_2, \\ & A_6 \doteq \forall r.\forall r.\forall r.(P_1 \sqcap P_2)\}. \end{aligned}$$

[4] Here, we only consider the size of the concept term. Note that one can also define the size of a concept description in role-based normal form that distinguishes between primitive and defined concepts.

The unique concept-based minimal rewriting is given as

$$D_c := \forall\{r\}A_1 \sqcap \forall\{r\}.A_2 \sqcap \forall\{r\}.A_6 \sqcap \forall\{rrr\}.A_3,$$

and the unique role-based minimal rewriting is given as

$$D_r := \forall r.\forall r.\forall r.(A_3 \sqcap A_4 \sqcap A_5).$$

Now, the role-based normal form of D_c is given as

$$D_{cr} := \forall r.(A_1 \sqcap A_2 \sqcap A_6 \sqcap \forall r.\forall r.A_3),$$

and the concept-based normal form of D_r is given as

$$D_{rc} := \forall\{rrr\}.A_3 \sqcap \forall\{rrr\}.A_4 \sqcap \forall\{rrr\}.A_5.$$

Obviously, we have $\|D_c\| = 10 < 12 = \|D_{rc}\|$ and $\|D_r\| = 6 < 7 = \|D_{cr}\|$.

The above example shows that computing a *role-based minimal rewriting* cannot be done by just computing the role-based normal form of a concept-based minimal rewriting. But for the induced decision problem, we can re-use the decision algorithm from the concept-based case.

Formally, the decision problem induced by the minimal rewriting problem for $\mathcal{FL}_0$-concept descriptions in role-based normal form is given as follows:

Given: an $\mathcal{FL}_0$-concept description C, an $\mathcal{FL}_0$-TBox $\mathcal{T}$, and a number $\kappa \in \mathbb{N}$.
Question: Does there exist a rewriting D in role-based normal form of C using $\mathcal{T}$ of size $\|D\| \leq \kappa$?

This problem can be decided in non-deterministic polynomial time as follows: As shown in the proof of Theorem 11, the concept-based normal form of every rewriting of C using $\mathcal{T}$ is of the form

$$\forall M_1'.A_1 \sqcap \ldots \sqcap \forall M_m'.A_m \sqcap \forall K_1'.P_1 \sqcap \ldots \sqcap \forall K_k'.P_k,$$

where M_j' resp. K_i' are subsets of the sets M_j, K_i occuring in the maximal solution (see Lemma 13). So, in order to decide wether there exists a rewriting in *role-based normal form* of size $\leq \kappa$, we only have to modify the test on the size of the possible solution:

1. Guess the sets $M_j' \subseteq \mathcal{X}$, $K_i' \subseteq \mathcal{X}$, i.e., determine a possible solution (see proof of Theorem 11).
2. Test wether the $\mathcal{FL}_0$-concept description induced by M_j', K_i' is equivalent to C (in polynomial time, see Theorem 7).
3. Compute the role-based normal form D of the $\mathcal{FL}_0$-concept description induced by M_j', K_i', and test wether $\|D\| \leq \kappa$.

Return “yes”, which means that there exists a rewriting in role-based normal form with size $\leq \kappa$, if there exists a successful computation of the steps 1–3, i.e., there exist sets M'_j, K'_i such that the induced $\mathcal{FL}_0$-concept description is equivalent to C w.r.t. $\mathcal{T}$ and has size $\leq \kappa$. Otherwise, return “no”.

Obviously, this algorithm answers “yes” iff there exists a rewriting D in role-based normal form of C using $\mathcal{T}$ with $\|D\| \leq \kappa$. Since the steps 1–3 can be executed in polynomial time, this shows that the problem can be decided in non-deterministic polynomial time.

4 Conclusion and future work

In this work, we first introduced a general framework for rewritng concept descriptions using a TBox. For the problem of computing a minimal rewriting in $\mathcal{FL}_0$ of an $\mathcal{FL}_0$-concept description using an $\mathcal{FL}_0$-TBox, we obtained the following complexity result: For a given TBox, the minimal rewriting decision problem is NP-complete.

In future work, we will consider the (minimal) rewriting problem for more expressive DLs. Motivated by the results in [2] and [4], we will extend the results to (i) DLs that allow for number restrictions, and (ii) DLs that allow for existential restrictions.

References

[1] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematic and Artificial Intelligence*, 18:175–219, 1996.

[2] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In O. Herzog and A. Günter, editors, *Proceedings of the 22nd Annual German Conference on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140, Bremen, Germany, 1998. Springer–Verlag.

[3] F. Baader, R. Küsters, and R. Molitor. Structural subsumption considered from an automata theoretic point of view. In *Proceedings of the 1998 International Workshop on Description Logics (DL'98)*, Trento, Italy, 1998.

[4] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence 1999 (IJCAI'99)*, 1999. To appear.

[5] F. Baader and P. Narendran. Unification of concept terms in description logics. In H. Prade, editor, *Proceedings of the 13th European Conference on*

Artificial Intelligence (ECAI-98), pages 331–335. John Wiley & Sons Ltd, 1998.

[6] F. Baader and U. Sattler. Knowledge representation in process engineering. In *Proceedings of the 1996 International Workshop on Description Logic (DL'96)*, 1996.

[7] C. Beeri, A.Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, Tucson, Arizona, 1997.

[8] A. Borgida and P. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[10] B. Nebel. Terminological reasoning is inherently intractable. *Journal on Artificial Intelligence*, 43(2):235–249, 1990.

[11] U. Sattler. *Terminological knowledge representation systems in a process engineering application.* PhD thesis, RWTH Aachen, 1998.