Rewriting Concepts Using Terminologies

Franz Baader, Ralf Küsters, and Ralf Molitor
Theoretical Computer Science, RWTH Aachen
Ahornstraße 55, 52074 Aachen, Germany
email: {baader,kuesters,molitor}@informatik.rwth-aachen.de

Abstract

The problem of rewriting a concept given a terminology can informally be stated as follows: given a terminology $\mathcal{T}$ (i.e., a set of concept definitions) and a concept description C that does not contain concept names defined in $\mathcal{T}$, can this description be rewritten into a "related better" description E by using (some of) the names defined in $\mathcal{T}$?

In this paper, we first introduce a general framework for the rewriting problem in description logics, and then concentrate on one specific instance of the framework, namely the minimal rewriting problem (where "better" means shorter, and "related" means equivalent). We investigate the complexity of the decision problem induced by the minimal rewriting problem for the languages $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$, and then introduce an algorithm for computing (minimal) rewritings for the language $\mathcal{ALE}$. (In the full paper, a similar algorithm is also developed for $\mathcal{ALN}$.) Finally, we sketch other interesting instances of the framework.

1 Motivation

In description logics (DLs), the standard inference problems, like the subsumption and the instance problem, are now well-investigated. More recently, new types of inference problems have been introduced and investigated, like matching (Borgida and McGuinness, 1996; Baader et al., 1999a; Baader and Küsters, 2000) and computing the least common subsumer (Cohen et al., 1992; Cohen and Hirsh, 1994; Baader and Küsters, 1998; Baader et al., 1999b). In contrast to the standard inferences, algorithms that solve these nonstandard problems produce *concept descriptions as output,* which are then returned to the user for inspection. For example, in an application in chemical process engineering (Baader and Sattler, 1996; Sattler, 1998) we try to support the bottom-up construction of knowledge bases 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 (Baader and Küsters, 1998; Baader et al., 1999b). This description is then offered to the knowledge engineer as a possible candidate for a definition of the concept.

In such a framework, it is important that the returned description is as readable and comprehensible as possible. Unfortunately, the descriptions that are produced by the known algorithms for solving the nonstandard inference problems in general do not satisfy this requirement. The reason is that – like most algorithms for the standard inference problems – these algorithms work on unfolded descriptions, i.e., concept descriptions that do not contain names defined in the underlying terminology (TBox). Consequently, the descriptions that they produce also do not use defined names, which makes them large and hard to read and comprehend. One possibility to overcome this problem would be to modify the known algorithms for the nonstandard inference problems such that they can take defined names into account. In order to avoid having to modify all these algorithms separately, we propose not to change the algorithms themselves, but to add rewriting as a post-processing step to them.

Informally, the problem of rewriting a concept given a terminology can be stated as follows: given a TBox $\mathcal{T}$ (i.e., a set of concept definitions) and a concept description C that does not contain concept names defined in $\mathcal{T}$, can this description be rewrit-

ten into an "related better" description E by using (some of) the names defined in $\mathcal{T}$? In this paper, related will mean equivalent, and better will mean shorter (but one can also imagine other optimality criteria). For example, if $\mathcal{T}$ contains the definition $\mathsf{Parent} \doteq \mathsf{Human} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Human}$, then the concept description $\mathsf{Human} \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Human} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Human})$ can be rewritten into the two smaller descriptions $\mathsf{Human} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Parent}$ and $\mathsf{Parent} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Parent}$, which are both equivalent to the original description.

The formal framework for rewriting that will be introduced in Section 3 encompasses this type of rewriting (called the minimal rewriting problem in the following), but also has other interesting instances (see Section 7). In Section 4 we investigate the complexity of the decision problem induced by the minimal rewriting problem for the DLs $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$. This will show that (unless P = NP) minimal rewritings cannot be computed in polynomial time, even for DLs with a polynomial subsumption problem. Section 5 then sketches an algorithm for computing all minimal rewritings for the DL $\mathcal{ALE}$. (A similar algorithm exists for $\mathcal{ALN}$ (Baader et al., 1999c).) In Section 6, we describe a more efficient rewriting algorithm, which computes one (possibly non-minimal) rewriting using a greedy heuristics. Due to space limitations, we cannot give complete proofs of all the results presented in this paper. All details can, however, be found in the full paper (Baader et al., 1999c).

2 Preliminaries

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*. In this work, we consider concept descriptions built from the constructors shown in Table 1. The concept descriptions in the description logics $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$ are built using certain subsets of these constructors, as shown in the last four columns of Table 1. When talking about an arbitrary DL, we will usually employ the letter $\mathcal{L}$ (possibly with subscript).

The semantics of 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 $P \in N_C$ to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

The terminology of an application domain can be represented in a so-called TBox. A *TBox* $\mathcal{T}$ is a finite set of *concept definitions* of the form $A \doteq C$, where $A \in N_C$ is a concept name and C is a concept description. The interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all concept definitions $A \doteq C$ in $\mathcal{T}$.

The concept name A is a *defined name* 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 name*. The concept description C in $A \doteq C$ is called the *defining concept of* A. For a given DL $\mathcal{L}$, we talk about $\mathcal{L}$-concept descriptions and $\mathcal{L}$-TBoxes, if all constructors occurring in the concept descriptions and concept definitions belong to $\mathcal{L}$. Throughout the paper, we assume TBoxes to be (1) without multiple definitions, i.e., for each defined name A, there exists a unique concept definition of the form $A \doteq C$ in $\mathcal{T}$; and (2) acyclic, i.e., the defining concept of a defined name must not, directly or indirectly, refer to this name (see (Nebel, 1990a) for exact definitions). The TBox $\mathcal{T}$ is called *unfolded* iff all defining concepts in $\mathcal{T}$ do not contain defined names (Nebel, 1990a). Because of our assumptions on TBoxes, a given TBox can always be transformed into an equivalent unfolded TBox; however, this unfolding process can lead to an exponential blow-up of the TBox (Nebel, 1990b).

One of the most important inference services provided by DL systems is computing the subsumption hierarchy. The concept description D *subsumes* the concept description C ($C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$; D is *equivalent* to C ($C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$. In the presence of a TBox $\mathcal{T}$, we say that D *subsumes* C *modulo* $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$, and C is *equivalent to* D *modulo* $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$.

3 A general framework for rewriting in DLs

Definition 1 (Rewriting) *Let N_R be a set of role names and N_P a set of primitive names, and let $\mathcal{L}_s$, $\mathcal{L}_d$, and $\mathcal{L}_t$ be three DLs (the source-, destination, and TBox-DL, respectively). A* rewriting problem *is given by*

- *an $\mathcal{L}_t$-TBox $\mathcal{T}$ containing only role names from N_R and primitive names from N_P; the set of defined names occurring in $\mathcal{T}$ is denoted by N_D;*
- *an $\mathcal{L}_s$-concept description C using only the names from N_R and N_P;*
- *a binary relation $\rho \subseteq \mathcal{L}_s \times \mathcal{L}_d$ between $\mathcal{L}_s$- and $\mathcal{L}_d$-concept descriptions.*

Construct name	Syntax	Semantics	$\mathcal{FL}_0$	$\mathcal{ALE}$	$\mathcal{ALC}$	$\mathcal{ALN}$
Top	$\top$	$\Delta^{\mathcal{I}}$		x	x	x
Bottom	$\bot$	$\emptyset$		x	x	x
Primitive negation ($P \in N_C$)	$\neg P$	$\Delta^{\mathcal{I}} \setminus P^{\mathcal{I}}$		x	x	x
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$			x	
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	x	x	x	x
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$			x	
Existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$		x	x	
Value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	x	x	x	x
At least number restriction	$(\geq n\, r)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$				x
At most number restriction	$(\leq n\, r)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$				x

Table 1: Syntax and semantics of concept descriptions.

An $\mathcal{L}_d$-rewriting of C using $\mathcal{T}$ *is an* $\mathcal{L}_d$*-concept description* E *built using names from* N_R *and* $N_P \cup N_D$ *such that* $C \rho E$.

Given an appropriate ordering $\preceq$ *on* $\mathcal{L}_d$*-concepts, a rewriting* E *is called* $\preceq$-minimal *iff there does not exist a rewriting* E' *such that* $E' \prec E$.

As an example, consider the instance of the framework where all three DLs are the language $\mathcal{ALN}$, the relation ρ is instantiated by equivalence modulo $\mathcal{T}$, and the ordering $\preceq$ is induced by the size of the concept descriptions. Let

$$
\begin{array}{lcl}
C & = & \mathsf{Male} \sqcap \mathsf{Rich} \sqcap (\geq 1\ \mathsf{has\text{-}child}) \sqcap \\
 & & \forall \mathsf{has\text{-}child}.(\mathsf{Male} \sqcap \mathsf{Rich}), \quad \text{and} \\
\mathcal{T} & = & \{\mathsf{Father} \doteq \mathsf{Male} \sqcap (\geq 1\ \mathsf{has\text{-}child}), \\
 & & \mathsf{RichParent} \doteq \mathsf{Rich} \sqcap \forall \mathsf{has\text{-}child}.\mathsf{Rich} \sqcap \\
 & & \qquad (\geq 1\ \mathsf{has\text{-}child}), \\
 & & \mathsf{FatherOfSons} \doteq \mathsf{Father} \sqcap \forall \mathsf{has\text{-}child}.\mathsf{Male}\}.
\end{array}
$$

It is easy to see that the concept description $\mathsf{FatherOfSons} \sqcap \mathsf{RichParent}$ is an $\mathcal{ALN}$-rewriting of C using $\mathcal{T}$, and that its size is minimal.

This was an example of what we will call the *minimal rewriting problem*, i.e., the instance of the framework where (i) all three DLs are the same language $\mathcal{L}$; (ii) the binary relation ρ corresponds to equivalence modulo the TBox; and (iii) $\mathcal{L}$-concept descriptions are ordered by size, i.e., $E \preceq E'$ iff $|E| \leq |E'|$. The size $|E|$ of a concept description E is defined to be the number of occurrences of concept and role names in E (where $\top$ and $\bot$ are *not* counted).

In the present paper, we will restrict our attention to the minimal rewriting problem for the DLs $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$. Other interesting instances of the framework will be mentioned in Section 7.

4 The minimal rewriting decision problem

In order to determine the complexity of the minimal rewriting problem, we first consider the *decision problem* induced by this optimization problem for a given DL $\mathcal{L}$: Given an $\mathcal{L}$-concept description C, an $\mathcal{L}$-TBox $\mathcal{T}$, and a nonnegative integer κ, does there exist an $\mathcal{L}$-rewriting E of C using $\mathcal{T}$ such that $|E| \leq \kappa$?

Since this decision problem can obviously be reduced to the problem of computing a minimal rewriting of C using $\mathcal{T}$, hardness results for the decision problem carry over to the optimization problem. In the sequel, we give lower and upper bounds for the complexity of the minimal rewriting decision problem for the DLs $\mathcal{FL}_0$, $\mathcal{ALN}$, $\mathcal{ALE}$, and $\mathcal{ALC}$.

NP-Hardness for $\mathcal{FL}_0$, $\mathcal{ALN}$, and $\mathcal{ALE}$

We give a reduction of the NP-complete problem SET-COVER (Garey and Johnson, 1979) to the minimal rewriting decision problem in $\mathcal{FL}_0$. An instance of the SETCOVER problem is of the following form:

Instance: A finite set $\mathcal{U} = \{u_1, \ldots, u_n\}$, a family $\mathcal{F} = \{F_i \subseteq \mathcal{U} \mid 1 \leq i \leq m\}$ of non-empty subsets of $\mathcal{U}$, and a nonnegative integer κ.

Question: Does there exist a subset $\{F_{i_1}, \ldots, F_{i_k}\}$ of $\mathcal{F}$ of size $k \leq \kappa$ 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 at least $\mathcal{F}$ itself covers $\mathcal{U}$, i.e., $F_1 \cup \ldots \cup F_m = \mathcal{U}$.

For a given instance $(\mathcal{U}, \mathcal{F}, \kappa)$ of the SETCOVER problem, we view $\mathcal{U}$ as set of primitive names, and define the corresponding instance of the minimal rewriting decision problem in $\mathcal{FL}_0$ as follows:

$$C_{\mathcal{U}} \quad := \quad u_1 \sqcap \ldots \sqcap u_n$$

$$\mathcal{T}_{\mathcal{F}} \quad := \quad \{A_j \doteq \bigsqcap_{u \in F_j} u \mid 1 \leq j \leq m\}.$$

NP-hardness for the minimal rewriting decision problem in $\mathcal{FL}_0$ is an immediate consequence of the following lemma.

Lemma 2 *There exists a minimal rewriting E of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$ with $|E| \leq \kappa$ iff there exists a cover of $\mathcal{U}$ with $k \leq \kappa$ sets from $\mathcal{F}$.*

Proof: A rewriting of $C_{\mathcal{U}}$ of size $k \leq \kappa$ is of the form $D = A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k$ for some $1 \leq l \leq k$ and $v_j \in \mathcal{U}$ (for $l+1 \leq j \leq k$).

First, we show that we can (w.l.o.g.) assume that $l = k$, i.e., D does not contain primitive names. Since $\mathcal{F}$ covers $\mathcal{U}$, we know that for each v_j, $l+1 \leq j \leq k$, there exists $F_{i_j} \in \mathcal{F}$ with $v_j \in F_{i_j}$. Thus, replacing each v_j by A_{i_j} yields a rewriting D' of $C_{\mathcal{U}}$ such that D' does not contain primitive names, and $|D'| \leq |D|$.

Now, let $D = A_{i_1} \sqcap \ldots \sqcap A_{i_k}$ be a rewriting of $C_{\mathcal{U}}$ that does not contain primitive names. Then $C \equiv_{\mathcal{T}} D$ implies that, for each $u \in \mathcal{U}$ there exists a defined name A_{i_j} such that u occurs in the right-hand side of the definition of A_{i_j}. Hence, $F_{i_1} \cup \ldots \cup F_{i_k}$ is a cover of $\mathcal{U}$ of size $k \leq \kappa$.

Conversely, let $F_{i_1} \cup \ldots \cup F_{i_k}$ be a cover of $\mathcal{U}$ of size $k \leq \kappa$. Then $D := A_{i_1} \sqcap \ldots \sqcap A_{i_k}$ is a rewriting of $C_{\mathcal{U}}$ of size $k \leq \kappa$. □

It is easy to see that the above reduction is still valid if we view the concept $C_{\mathcal{U}}$ as $\mathcal{ALN}$- or $\mathcal{ALE}$-concept and the TBox $\mathcal{T}_{\mathcal{F}}$ as $\mathcal{ALN}$- or $\mathcal{ALE}$-TBox (Baader et al., 1999c). Thus, the minimal rewriting decision problem is also NP-hard for $\mathcal{ALN}$ and $\mathcal{ALE}$.

PSPACE-Hardness for $\mathcal{ALC}$

The following lemma yields a reduction of subsumption in $\mathcal{ALC}$ to the minimal rewriting decision problem for $\mathcal{ALC}$. Since subsumption in $\mathcal{ALC}$ is PSPACE-complete (Schmidt-Schauss and Smolka, 1991), this implies PSPACE-hardness for the minimal rewriting decision problem for $\mathcal{ALC}$.

Lemma 3 *Let C, D be two $\mathcal{ALC}$-concept descriptions, and A, P_1, P_2 three different concept names not occurring in C, D. Then $C \sqsubseteq D$ iff there exists a minimal rewriting of size ≤ 1 of the $\mathcal{ALC}$-concept description $P_1 \sqcap P_2 \sqcap C$ using the TBox $\mathcal{T} := \{A \doteq P_1 \sqcap P_2 \sqcap C \sqcap D\}$.*

Proof: First, assume that $C \sqsubseteq D$. This implies $C \equiv C \sqcap D$ and thus $P_1 \sqcap P_2 \sqcap C \equiv P_1 \sqcap P_2 \sqcap C \sqcap D$. Hence, A is a rewriting of size ≤ 1 of $P_1 \sqcap P_2 \sqcap C$ w.r.t. $\mathcal{T}$.

Conversely, let E be a rewriting of size ≤ 1 of $P_1 \sqcap P_2 \sqcap C$ w.r.t. $\mathcal{T}$. We distinguish several cases.

1. $E = A$: Then $P_1 \sqcap P_2 \sqcap C \equiv P_1 \sqcap P_2 \sqcap C \sqcap D$. Since P_1 and P_2 do not occur in C and D, it is easy to show (Baader et al., 1999c) that the above equivalence implies $C \equiv C \sqcap D$, and thus $C \sqcap D$.

2. $E = \bot$: Then $P_1 \sqcap P_2 \sqcap C \equiv \bot$. Since P_1, P_2 are primitive names not occurring in C, we obtain $C \equiv \bot$, and thus $C \sqsubseteq D$.

3. $E = \top$: Then $P_1 \sqcap P_2 \sqcap C \equiv \top$ in contradiction to $P_1 \sqcap P_2 \sqsubset \top$.

4. $E = Q$ for a concept name Q distinct from A: For $Q \in \{P_1, P_2\}$, let w.l.o.g. $Q = P_1$. Then $P_1 \equiv P_1 \sqcap P_2 \sqcap C$. This implies $P_1 \sqsubseteq P_1 \sqcap P_2 \sqcap C$ and hence $P_1 \sqsubseteq P_2$ in contradiction to the fact that P_1 and P_2 are different primitive names. Finally, assume $Q \notin \{A, P_1, P_2\}$. Then $Q \equiv P_1 \sqcap P_2 \sqcap C$ implies $Q \sqsubseteq P_1 \sqcap P_2 \sqcap C$, and hence $Q \sqsubseteq P_1$ in contradiction to the fact that Q and P_1 are different primitive names.

All other cases where $|E| \leq 1$ (e.g., $E = \neg E'$ with $|E'| \leq 1$; $E = \forall r.E'/\exists r.E'$ with $|E'| = 0$; ...) can be treated analogously (see (Baader et al., 1999c) for details). □

This reduction of subsumption to the minimal rewriting decision problem also works for sublanguages of $\mathcal{ALC}$ (if they allow for conjunction) as well as for extensions of $\mathcal{ALC}$ known from the literature. This shows that, for all such DLs, the minimal rewriting decision problem is at least as hard as the subsumption problem. Note that this yields an alternative proof of NP-hardness of the minimal rewriting decision problem for $\mathcal{ALE}$, but not for $\mathcal{FL}_0$ and $\mathcal{ALN}$ (since subsumption is polynomial for these languages).

A general upper bound

The following simple algorithm decides whether there exists a rewriting of C using $\mathcal{T}$ of size $\leq \kappa$ in nondeterministic polynomial time, using an oracle for deciding equivalence modulo TBox: First, nondeterministically compute a concept description E of size $\leq \kappa$; then test whether $E \equiv_{\mathcal{T}} C$.

Note that testing $E \equiv_{\mathcal{T}} C$ is a special case of the general equivalence problem modulo TBox: C does *not* contain defined names. In fact, we have shown that this *restricted equivalence problem* is less complex than the general problem for the DLs $\mathcal{FL}_0$ and $\mathcal{ALN}$ (see (Baader et al., 1999c) for details).

The complexity results for the minimal rewriting decision problem for the DLs under consideration are sum-

TBox	unfolded	not unfolded
$\mathcal{FL}_0$	NP-complete	NP-complete
$\mathcal{ALN}$	NP-complete	in Σ_2^p, NP-hard
$\mathcal{ALE}$	NP-complete	in PSPACE, NP-hard
$\mathcal{ALC}$	PSPACE-complete	PSPACE-complete

Table 2: Summary of the complexity results.

marized in Table 2. The upper bounds are obtained from

- the simple algorithm described above,
- two new complexity results for the restricted equivalence problem for $\mathcal{FL}_0$ (in P) and $\mathcal{ALN}$ (in Δ_2^p), and
- known complexity results for the equivalence problem modulo TBox for $\mathcal{ALE}$ and $\mathcal{ALC}$ (in PSPACE) (Donini et al., 1992; Lutz, 1999).

It should be noted that there are two independent sources of complexity for the minimal rewriting problem. On the one hand, we have to decide equivalence modulo TBox in order to test whether a computed concept description is a rewriting. On the other hand, in order to compute a minimal rewriting, we have to solve an optimization problem. Since the restricted equivalence problem for $\mathcal{FL}_0$ can be decided in polynomial time, the hardness result for $\mathcal{FL}_0$ implies that this optimization problem is hard, independently of the complexity of the equivalence problem.

5 The minimal rewriting computation problem

Whereas the previous section was concerned with deciding whether there exists a (minimal) rewriting within a given size bound, this section considers the problem of actually computing (minimal) rewritings. Due to lack of space, we restrict our attention to $\mathcal{ALE}$, but all notions and results can easily be adapted to $\mathcal{ALN}$ (Baader et al., 1999c).

For a given instance $(C, \mathcal{T})$ of the minimal rewriting *computation problem*, one is interested in either computing (1) *one* minimal rewriting of C using $\mathcal{T}$, or (2) *all* minimal rewritings of C using $\mathcal{T}$.

The hardness results of the previous section imply that computing one minimal rewriting is already a hard problem. In addition, the following simple example shows that the number of minimal rewritings of a concept description C using a TBox $\mathcal{T}$ can be exponential in the size of C and $\mathcal{T}$. This example works for all the four DLs considered in the previous section.

Input: An $\mathcal{ALE}$-concept description C in $\forall$-normal form and an $\mathcal{ALE}$-TBox $\mathcal{T}$.
Algorithm:
Compute an extension C^* of C.
Compute a reduction $\widehat{C}$ of C^* w.r.t. $\mathcal{T}$.
Return $\widehat{C}$.

Figure 1: The rewriting algorithm for $\mathcal{ALE}$.

For a nonnegative integer n, let

$$C_n := P_1 \sqcap \ldots \sqcap P_n \quad \text{and}$$
$$\mathcal{T}_n := \{A_i \doteq P_i \mid 1 \leq i \leq n\}.$$

For each vector $\mathbf{i} = (i_1, \ldots, i_n) \in \{0,1\}^n$, we define

$$E_{\mathbf{i}} := \bigsqcap_{1 \leq j \leq n, i_j = 0} P_j \sqcap \bigsqcap_{1 \leq j \leq n, i_j = 1} A_j.$$

Obviously, for all $\mathbf{i} \in \{0,1\}^n$, $E_{\mathbf{i}}$ is a rewriting of C_n of size $|E_{\mathbf{i}}| = n = |C_n|$. Furthermore, it is easy to see that there does not exist a smaller rewriting of C_n using $\mathcal{T}_n$. Hence, there exists an exponential number of different minimal rewritings of C_n using $\mathcal{T}_n$.

A *naïve algorithm* for computing one minimal rewriting would enumerate all concept descriptions E of size $k = 1$, then $k = 2$, etc., until a rewriting E_0 of C using $\mathcal{T}$ is encountered. By construction, this rewriting is minimal, and since C is a rewriting of itself, one need not consider sizes larger than $|C|$. If one is interested in computing all minimal rewritings, it remains to enumerate all concept descriptions of size $|E_0|$, and test for each of them whether they are equivalent to C modulo $\mathcal{T}$.

Obviously, this naïve algorithm is very inefficient. Its main drawback is that it is not source-oriented: the candidate rewritings are computed without using the input C. The main contribution of this paper is a nondeterministic rewriting algorithm that computes rewritings by directly modifying the input concept C. More precisely, the algorithm will work on the $\forall$-*normal form* of the input concept, i.e., the normal form obtained from C by exhaustively applying the rule $\forall r.E \sqcap \forall r.F \longrightarrow \forall r.(E \sqcap F)$. This normal form can be computed in polynomial time.

The idea underlying the improved algorithm depicted in Figure 1 is to split the computation of a rewriting E into two steps. First, an extension of C w.r.t. $\mathcal{T}$ is computed.

Definition 4 (Extension) *Let C be an $\mathcal{ALE}$-concept description and $\mathcal{T}$ an $\mathcal{ALE}$-TBox. The concept description C^* is an* extension of C w.r.t. $\mathcal{T}$ *iff $C^* \equiv_{\mathcal{T}} C$ and C^* can be obtained from C by conjoining defined names at some positions in C.*

In the second step, a so-called *reduction of C^* w.r.t. $\mathcal{T}$* is computed, i.e., a concept description $\widehat{C}$ that is (i) equivalent to C^* modulo $\mathcal{T}$, and (ii) obtained from C^* by eliminating all the redundancies in C^*.

The main technical problem to be solved is to give an appropriate formal definition of reduction, and to show how reductions can be computed. Before we go into such detail, we (1) give an example illustrating the rewriting algorithm of Figure 1; and (2) explain what this algorithm actually computes.

(1) As an example, consider the $\mathcal{ALE}$-concept description

$$C = P \sqcap Q \sqcap \forall r.P \sqcap \exists r.(P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q)),$$

and the $\mathcal{ALE}$-TBox $\mathcal{T} = \{\ A_1 \doteq \exists r.Q,\ A_2 \doteq P \sqcap \forall r.P,\ A_3 \doteq \forall r.P\ \}$. The concept description

$$\begin{aligned} C^* \quad = \quad & A_2 \sqcap P \sqcap Q \sqcap \forall r.P \sqcap \\ & \exists r.(A_1 \sqcap P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q)) \end{aligned}$$

is an extension of C. A reduction of C^* can be obtained by eliminating

- P and $\forall r.P$ on the top-level of C^*, because they are redundant w.r.t. A_2;
- P in both of the existential restrictions on the top-level of C^*, because it is redundant due to the value restriction $\forall r.P$;
- the existential restriction $\exists r.Q$, because it is redundant w.r.t. A_1;

and replacing $Q \sqcap \neg Q$ by $\bot$, since $\bot$ is the minimal inconsistent concept description. The resulting concept description $\widehat{C} = A_2 \sqcap Q \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$ is equivalent to C modulo $\mathcal{T}$, i.e., $\widehat{C}$ is a rewriting of C using $\mathcal{T}$. Furthermore, it is easy to see that $\widehat{C}$ is a minimal rewriting of C using $\mathcal{T}$.

(2) Obviously, there may exist exponentially many essentially different (i.e., not equivalent w.r.t. the empty TBox) extensions of C, and we can show (Baader et al., 1999c) that, for $\mathcal{ALE}$, each extension may have exponentially many essentially different reductions (for $\mathcal{ALN}$, reductions are unique). Thus, the algorithm of Figure 1 should be viewed as a nondeterministic algorithm (with an oracle for the equivalence problem). We will show that it is correct in the following sense:

Theorem 5 *1. Every possible output of the algorithm in Figure 1 is a rewriting of the input concept description C using the input TBox $\mathcal{T}$.*

2. The set of all computed rewritings contains all minimal rewritings of C using $\mathcal{T}$ (modulo associativity, commutativity and idempotence of conjunction, and the equivalence $C \sqcap \top \equiv C$).

If we compute just one extension and then one reduction of this extension, then we have a deterministic and polynomial-time algorithm (with an oracle for equivalence) for computing one rewriting; however, the computed rewriting need not be minimal. Nevertheless, this opens the way for a heuristic approach to computing "small" (rather than minimal) rewritings (see Section 6). We can also show the following (Baader et al., 1999c): if we compute all extensions and then just one reduction of each extension, then the set of all rewritings computed this way always contains at least one minimal rewriting.

Reduction of $\mathcal{ALE}$-concept descriptions

For the sake of simplicity, we assume the set of role names N_R to be the singleton $\{r\}$. However, the definitions and results can easily be generalized to arbitrary sets of role names (Baader et al., 1999c).

In order to formalize the notion of a reduction for $\mathcal{ALE}$, we need the following notation.

Definition 6 (Subdescription) *Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox and C an $\mathcal{ALE}$-concept description that may contain defined names from $\mathcal{T}$. The $\mathcal{ALE}$-concept description $\widehat{C}$ is a* subdescription of C w.r.t. $\mathcal{T}$ *iff* (i) *$\widehat{C} = C$; or* (ii) *$\widehat{C} = \bot$; or* (iii) *$\widehat{C}$ is obtained from C by removing some (negated)* primitive *names, value restrictions, or existential restrictions on the top-level of C, and for all remaining value/existential restrictions $\forall r.D/\exists r.D$ replacing D by a subdescription $\widehat{D}$ of D.*

In the above example, the concept description $\widehat{C}$ is a subdescription of C^*, whereas the concept description $Q \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$ is not since we do not allow for removing defined names in C (unless they occur within value or existential restrictions that are removed as a whole).

Definition 7 (Reduction w.r.t. $\mathcal{T}$) *Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox and C an $\mathcal{ALE}$-concept description in $\forall$-normal form that may contain defined names from $\mathcal{T}$. An $\mathcal{ALE}$-concept description $\widehat{C}$ is called* reduction of C w.r.t. $\mathcal{T}$ *iff $\widehat{C}$ is a minimal subdescription of C that is equivalent to C modulo $\mathcal{T}$.*

Disallowing the removal of defined names in the definition of the notion "subdescription" makes sense since

Input: An $\mathcal{ALE}$-concept description C in $\forall$-normal form, an $\mathcal{ALE}$-TBox $\mathcal{T}$, and an $\mathcal{ALE}$-concept description F.

Algorithm: $\mathsf{reduce}(C, \mathcal{T}, F)$

If $C \sqcap F \equiv_{\mathcal{T}} \bot$, then $\widehat{C} := \bot$;

Otherwise,

Let $\{A_1, \ldots, A_m\} := \mathit{def}(C)$;

Let $\{Q_1, \ldots, Q_\ell\} := \mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$;

If $\mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(C)$

then $D^r := \top$

else $D^r := \mathsf{reduce}(\mathit{val}_r(C), \mathcal{T}, \mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)))$;

Let $\mathcal{D}$ be a subset of the set $\mathcal{C} := \{\mathsf{reduce}(C_j, \mathcal{T}, \mathit{val}_r(C) \sqcap \mathit{val}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))) \mid C_j \in \mathit{exr}_r(C)\}$ such that

1. there do not exist $D_1, D_2 \in \mathcal{D}$, $D_1 \neq D_2$, with $\mathit{val}_r(C) \sqcap \mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqcap D_1 \sqsubseteq_{\mathcal{T}} \mathit{val}_r(C) \sqcap \mathit{val}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqcap D_2$,
2. there does not exist $D \in \mathcal{D}$ with $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.\mathit{val}_r(C) \sqsubseteq_{\mathcal{T}} \exists r.D$,
3. for each $C_i \in \mathit{exr}_r(C)$, there exists $D \in \mathcal{D}$ with $\exists r.D \sqcap \forall r.\mathit{val}_r(C) \sqcap F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} \exists r.C_i$; or $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.\mathit{val}_r(C) \sqsubseteq_{\mathcal{T}} \exists r.C_i$, and
4. the size $\sum_{D \in \mathcal{D}}(|D| + 1)$ of the set is minimal among the sizes of all subsets of $\mathcal{C}$ satisfying (1)–(3);

Define $\widehat{C} := Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.D^r \sqcap \bigsqcap_{D \in \mathcal{D}} \exists r.D$,

where the value restriction $\forall r.D^r$ is omitted if $D^r = \top$;

Return $\widehat{C}$.

Figure 2: The reduction algorithm for $\mathcal{ALE}$.

the reduction step is always applied after the extension step. It is possible that removal of defined names could yield a smaller rewriting, but this rewriting is obtained when considering the extension where these names have not been added in the first place. Allowing the removal of defined names would thus only increase the amount of nondeterminism without creating additional rewritings.

In the sequel, we describe an algorithm that computes a reduction of C in deterministic polynomial time (using an oracle for deciding equivalence modulo $\mathcal{T}$). The set of all reductions of C can be computed in exponential time.

Intuitively, a reduction $\widehat{C}$ of an $\mathcal{ALE}$-concept C in $\forall$-normal form is computed in a top-down manner. If $C \equiv_{\mathcal{T}} \bot$, then $\widehat{C} := \bot$. Otherwise, let $\forall r.C'$ be the (unique!) value restriction and $A_1 \sqcap \ldots \sqcap A_n$ the conjunction of the defined names on the top-level of C. Basically, $\widehat{C}$ is obtained from C as follows:

1. Remove the (negated) primitive concept Q occurring on the top-level of C, if $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq_{\mathcal{T}} Q$.
2. Remove $\exists r.C_1$ occurring on the top-level of C, if (a) $A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.C' \sqsubseteq_{\mathcal{T}} \exists r.C_1$, or (b) there is another existential restriction $\exists r.C_2$ on the top-level of C such that $A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.C' \sqcap \exists r.C_2 \sqsubseteq_{\mathcal{T}} \exists r.C_1$.
3. Remove $\forall r.C'$ if $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq_{\mathcal{T}} \forall r.C'$.
4. Finally, all concept descriptions D occurring in the remaining value and existential restrictions are reduced recursively.

The formal specification of the reduction algorithm is more complex than the intuitive description given above mainly for two reasons. First, in (2b) it could be the case that the subsumption relation also holds if the rôles of $\exists r.C_1$ and $\exists r.C_2$ are exchanged. In this case, one has a choice of which concept to remove. If the (recursive) reduction of C_1 and C_2 yields descriptions of different size, then we remove the existential restriction for the concept with the larger reduction. If, however, the reductions are of equal size, then we must make a (don't know) nondeterministic choice between removing the one or the other.

Second, in (4) we cannot really reduce the descriptions D without considering the context in which they occur. The reduction of these concepts must take into account the concept C' as well as all concepts D' occurring in value restrictions of the form $\forall r.D'$ on the top-level of the defining concepts for $A_1, \ldots, A_n$. For instance, consider our example from above, where the removal of P within the existential restrictions on the top-level of C^* was justified by the presence of $\forall r.P$ on the top-level of C^*. Since we want to apply the reduction algorithm recursively, we need a third input parameter to take care of the context. To be more precise, the reduction algorithm described in Figure 2 computes a reduction of an $\mathcal{ALE}$-concept description C w.r.t. an $\mathcal{ALE}$-TBox $\mathcal{T}$ and an $\mathcal{ALE}$-concept description F. A *reduction of C w.r.t. $\mathcal{T}$ and F* is a minimal subdescription $\widehat{C}$ of C such that $C \sqcap F \equiv_{\mathcal{T}} \widehat{C} \sqcap F$.

The formal specification of the reduction algorithm in Figure 2 is based on the following notations. Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox and C an $\mathcal{ALE}$-concept description that may contain defined names from $\mathcal{T}$. The *unfolded concept description* $\mathcal{T}(C)$ is defined as the concept description obtained from C by exhaustively substituting defined names in C by their defining concepts in $\mathcal{T}$.[1] The set of all defined names occurring on the top-level of C is denoted by $def(C)$, and the set of all (negated) primitive names occurring on the top-level of C is denoted by $prim(C)$. For an $\mathcal{ALE}$-concept description C and a role name r,

- $val_r(C)$ denotes the concept description occurring in the unique value restriction on the top-level of the $\forall$-normal form of C, where $val_r(C) := \top$ if there is no such value restriction; and
- $exr_r(C)$ denotes the set $\{C_1, \ldots, C_n\}$ of concept descriptions occurring in existential restrictions of the form $\exists r.C_i$ on the top-level of C.

The following lemma states soundness and completeness of the reduction algorithm of Figure 2.

Lemma 8 *(Baader et al., 1999c) Each output $\widehat{C}$ obtained from* $\mathsf{reduce}(C, \mathcal{T}, F)$ *is a reduction of C w.r.t. $\mathcal{T}$ and F. Conversely, for each reduction E of C w.r.t. $\mathcal{T}$ and F, there exists an output $\widehat{C}$ of* $\mathsf{reduce}(C, \mathcal{T}, F)$ *that is equal to E.*

Consequently, $\mathsf{reduce}(C, \mathcal{T}, \top)$ produces all reductions of C w.r.t. $\mathcal{T}$.

[1] Note that $\mathcal{T}(C)$ is well-defined due to our assumptions on TBoxes. However, just as for unfolding TBoxes, this step may lead to an exponential blow-up.

Proof of Theorem 5

The first item in the statement of Theorem 5 is a direct consequence of the definition of extensions and reductions. In order to prove the second item, let E be a minimal rewriting of C using $\mathcal{T}$. The main point is now that we can define an extension C^* of C induced by E such that E is a subdescription of C^*.

Intuitively, C^* can be obtained from C as follows:

1. conjoin to C all defined names occurring on the top-level of E;
2. if there exists a value restriction $\forall r.E'$ on the top-level of E, then there also exists a value restriction $\forall r.C'$ on the top-level of C (otherwise, C would not be equivalent to E modulo $\mathcal{T}$): substitute C' by the recursively defined extension of C' induced by E';
3. for each existential restriction $\exists r.E_i$ on the top-level of E, there exists a corresponding existential restriction $\exists r.C_i$ on the top-level of C such that $C_i \sqcap val_r(C) \equiv_{\mathcal{T}} E_i \sqcap val_r(C)$ (otherwise, C would not be equivalent to E modulo $\mathcal{T}$): substitute C_i by the recursively defined extension of C_i induced by E_i.

In the formal definition of C^*, we must, just as for the reduction algorithm, take into account the context in which a concept description occurs. To this purpose, we extend the notion "extension w.r.t. $\mathcal{T}$" to "extension w.r.t. $\mathcal{T}$ and F": C^* is an *extension of C w.r.t. $\mathcal{T}$ and F* iff C^* is obtained from C by conjoining defined names from $\mathcal{T}$ at any position in C such that $C^* \sqcap F \equiv_{\mathcal{T}} C \sqcap F$. Furthermore, we need the notion "reduced w.r.t. $\mathcal{T}$ and F": a concept description E is called *reduced w.r.t. $\mathcal{T}$ and F* if E is a reduction w.r.t. $\mathcal{T}$ and F of itself.

The recursive definition of an extension C^* of C induced by E w.r.t. $\mathcal{T}$ and F is depicted in Figure 3. This definition makes sense since it can be shown (Baader et al., 1999c) that there always exists a permutation of $exr_r(C)$ of the desired form.

In order to complete the proof of the second part of Theorem 5, we need the following lemma.

Lemma 9 *(Baader et al., 1999c) Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox, C, F, E $\mathcal{ALE}$-concept descriptions such that C is in $\forall$-normal form and does not contain defined names, E is reduced w.r.t. $\mathcal{T}$ and F, and $E \sqcap F \equiv_{\mathcal{T}} C \sqcap F$. If C^* is the concept description defined in Figure 3, then*

1. *C^* is an extension of C w.r.t. $\mathcal{T}$ and F, and*

Given: An $\mathcal{ALE}$-TBox $\mathcal{T}$, and $\mathcal{ALE}$-concept description C, F, E, where
- C is in $\forall$-normal form and does not contain defined names,
- E is reduced w.r.t. $\mathcal{T}$ and F, and $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$.

Recursive definition of the extension C^* of C w.r.t. $\mathcal{T}$ and F induced by E:
If $E \sqcap F \equiv_{\mathcal{T}} \bot$, then $C^* := C$;
Otherwise,
Let $\{Q_1, \ldots, Q_k\} := prim(C)$;
Let $\{A_1, \ldots, A_n\} := def(E)$;
Let D^r be the recursively defined extension of $val_r(C)$ w.r.t. $\mathcal{T}$ and $val_r(F)$ induced by $val_r(E)$;
Let $\mathrm{exr}_r(C) = \{C_1, \ldots, C_m\}$ and $\mathrm{exr}_r(E) = \{E_1, \ldots, E_\ell\}$;
Let $\{j_1, \ldots, j_m\}$ be a permutation of $\{1, \ldots, m\}$ such that, for all $1 \leq i \leq \ell$,

$$C_{j_i} \sqcap val_r(C \sqcap F) \equiv_{\mathcal{T}} E_i \sqcap val_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n));$$

For $1 \leq i \leq \ell$, let $C^*_{j_i}$ be the recursively defined extension of C_{j_i} w.r.t. $\mathcal{T}$ and $val_r(C \sqcap F)$ induced by E_i;
Then C^* is defined by

$$C^* \;:=\; Q_1 \sqcap \ldots \sqcap Q_k \sqcap A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.D^r \sqcap \bigsqcap_{1 \leq i \leq \ell} \exists r.C^*_{j_i} \sqcap \bigsqcap_{\ell+1 \leq i \leq m} \exists r.C_{j_i},$$

where the value restriction $\forall r.D^r$ is omitted if there does not exist a value restriction on the top-level of C.

Figure 3: The recursive definition of extensions w.r.t. $\mathcal{T}$ and F induced by E.

2. *E is a subdescription of C^*.*

Now, let E be a minimal rewriting of C using $\mathcal{T}$. Then E is reduced w.r.t. $\mathcal{T}$ and $\top$ since otherwise E would not be minimal. Let C^* be the extension of C w.r.t. $\mathcal{T}$ and $\top$ induced by E. By Lemma 9, we know that E is a subdescription of C^*. Thus, minimality of E implies that E is a reduction of C^*, and hence E is contained in the set of all rewritings computed by the algorithm (by Lemma 8).

Complexity of the minimal rewriting computation problem

Using the improved rewriting algorithm for $\mathcal{ALE}$ described in Figure 1, we can show the following complexity results.

Proposition 10 *1.* One *minimal rewriting of C using $\mathcal{T}$ can be computed using polynomial space.*

2. *The set of* all *minimal rewritings of C using $\mathcal{T}$ can be computed in exponential time.*

Proof: Each extension of C is polynomial (modulo idempotence) in the size of C and $\mathcal{T}$. Furthermore, there are "only" exponentially many (essentially different) extensions of C. Since equivalence modulo TBox in $\mathcal{ALE}$ can be decided in PSPACE (Lutz, 1999), the set of all extensions can be enumerated using polynomial space. For each extension C^*, the reductions $\widehat{C}$ can again be enumerated in polynomial space. Thus, if we are interested in just one minimal rewriting, it is sufficient always to store the smallest rewriting encountered so far. Hence, we can compute one minimal rewriting of C using polynomial space. Since the number of minimal rewritings may be exponential, the set of all minimal rewritings can only be computed in exponential time. □

6 A heuristic rewriting algorithm

In this section, we present an algorithm that computes a small, but not necessarily minimal, rewriting of an $\mathcal{ALE}$-concept description C using an $\mathcal{ALE}$-TBox $\mathcal{T}$ in *deterministic polynomial time* using an oracle for deciding equivalence modulo $\mathcal{T}$. The idea underlying the algorithm can be described as follows. Instead of first computing an extension of C and then the reduction of this extension, we interleave these two steps in a single pass through the concept. Both, for the extension and the reduction, we employ a *greedy heuristics*. To be more precise, the concept description C is processed recursively. In each recursion step, we build a local extension by conjoining to the top level of C the set $\{A_1, \ldots, A_n\}$ of *all* minimal (w.r.t. $\sqsubseteq_{\mathcal{T}}$) defined names in $\mathcal{T}$ subsuming C. Then we remove *all* (negated) primitive names, value restrictions, and exis-

tential restrictions on the top-level of C that are redundant w.r.t. $A_1, \ldots, A_n$, and the context in which they occur, i.e., the value restrictions obtained from previous recursion steps. Finally, the concept descriptions in the remaining value and existential restrictions are rewritten recursively. Like the reduction algorithm, the heuristic rewriting algorithm thus takes as inputs the concept C to be rewritten, the underlying TBox $\mathcal{T}$, and a concept description F describing the context C has to be considered in.

The formal specification of the rewriting algorithm requires an additional notation. For an $\mathcal{ALE}$-concept description C that may contain defined names, $\mathcal{T}^*(C)$ denotes the concept description obtained from C by exhaustively substituting defined names *on the top-level of* C by their defining concepts from the underlying TBox $\mathcal{T}$. In contrast to $\mathcal{T}(C)$, the size of $\mathcal{T}^*(C)$ is always polynomial in the size of C and $\mathcal{T}$. The following lemma states the correctness and the complexity of the heuristic rewriting algorithm for $\mathcal{ALE}$ depicted in Figure 4.

Lemma 11 *(Baader et al., 1999c) Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox, C, F $\mathcal{ALE}$-concept descriptions without defined names, and let $\widehat{C}$ be the result of* $\mathsf{rewrite}(C, \mathcal{T}, F)$.

1. $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$.
2. *$\widehat{C}$ is computed in deterministic polynomial time using an oracle for deciding subsumption modulo TBox in $\mathcal{ALE}$.*

The following example shows that the rewriting computed by the heuristic algorithm need not be minimal. For a nonnegative integer $n > 2$, we consider the $\mathcal{ALE}$-concept description $C_n = \forall r.(P_1 \sqcap \ldots \sqcap P_n)$ and the $\mathcal{ALE}$-TBox

$$\begin{aligned} \mathcal{T}_n \quad := \quad & \{ A_i \doteq \forall r.P_i \mid 1 \leq i \leq n\} \cup \\ & \{A_{n+1} \doteq P_1 \sqcap \ldots \sqcap P_n\}. \end{aligned}$$

The heuristic rewriting algorithm of Figure 4 produces the rewriting $\widehat{C}_n := A_1 \sqcap \ldots \sqcap A_n$ of size n. The unique minimal rewriting of C_n using $\mathcal{T}$ is $E_n := \forall r.A_{n+1}$, which is of size 2. Hence, this example even shows that the difference between the size of the rewriting produced by the heuristic algorithm and the size of the minimal rewritings can become arbitrarily large.

The reason why the heuristic algorithm does not find the minimal rewriting in this example is that it introduces too many defined names on the top level. These names allow for the removal of all the value restrictions on the top level, which makes it impossible to recognize that at a lower level a more promising extension could have been found.

In principle, this is the only reason for the heuristic algorithm not to find a minimal rewriting. In order to characterize the difference between the rewriting computed by the heuristic algorithm and the minimal rewritings, we need the notion of a quasi-subdescription. There are two differences between a subdescription and a quasi-subdescription: on the one hand, in a quasi-subdescription, we do not allow for substituting a concept description by $\bot$. On the other hand, we allow for conjoining defined names at some positions in C.

Definition 12 (Quasi-subdescription) *Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox and C an $\mathcal{ALE}$-concept description that may contain defined names from $\mathcal{T}$. The $\mathcal{ALE}$-concept description $\widehat{C}$ is a* quasi-subdescription of C w.r.t. $\mathcal{T}$ *iff (i) $\widehat{C} = C$, or (ii) $\widehat{C}$ is obtained from C by removing some (negated) primitive names, conjoining some defined names, removing some value restrictions or existential restrictions, and for all remaining value/existential restrictions $\forall r.D/\exists r.D$ replacing D by a quasi-subdescription $\widehat{D}$ of D.*

Lemma 13 *(Baader et al., 1999c) Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox, C an $\mathcal{ALE}$-concept description not containing defined names, and E, F $\mathcal{ALE}$-concept descriptions such that E is reduced w.r.t. $\mathcal{T}$ and F and $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$. The result $\widehat{C}$ of* $\mathsf{rewrite}(C, \mathcal{T}, F)$ *is a quasi-subdescription of E.*

If E is a minimal rewriting of C using $\mathcal{T}$, then E is reduced w.r.t. $\mathcal{T}$ and $\top$, and $C \sqcap \top \equiv_{\mathcal{T}} E \sqcap \top$. Thus, Lemma 13 implies that the rewriting $\widehat{C} := \mathsf{rewrite}(C, \mathcal{T}, \top)$ produced by the heuristic algorithm is a quasi-subdescription of E.

Intuitively, if we view concept descriptions as trees where edges are due to existential and value restrictions, and nodes are labeled with (negated) concept names, then the above result can be interpreted as follows. The rewriting $\widehat{C}$ produced by the heuristic algorithm may have a tree structure that is smaller than the one of the minimal rewriting E. The labels of the nodes in the tree corresponding to $\widehat{C}$ may contain less (negated) primitive names, but more defined names.

First experimental results

In order to study the usefulness of our minimal rewriting approach, we have implemented a prototype of the rewriting algorithm depicted in Figure 4. First results obtained in our process engineering application are encouraging: for a TBox with about 65 defined and 55 primitive names, 128 source descriptions of size about 800 (obtained as results of the lcs computation) were

Input: An $\mathcal{ALE}$-concept description C in $\forall$-normal form, an $\mathcal{ALE}$-TBox $\mathcal{T}$, and an $\mathcal{ALE}$-concept description F.

Algorithm: $\mathsf{rewrite}(C, \mathcal{T}, F)$
If $C \sqcap F \equiv_{\mathcal{T}} \bot$, then $\widehat{C} := \bot$;
If $F \sqsubseteq_{\mathcal{T}} C$, then $\widehat{C} := \top$;
Otherwise,
Let $\{A_1, \ldots, A_n\}$ be the set of all minimal defined names A_i with $C \sqcap F \sqsubseteq_{\mathcal{T}} A_i$;
Let $\{Q_1, \ldots, Q_\ell\} := prim(C) \setminus prim(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n))$;
Let $D^r := \mathsf{rewrite}(val_r(C), \mathcal{T}, val_r(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n)))$;
Let $\{D_1, \ldots, D_m\} := \mathrm{exr}_r(C)$ and $\mathcal{D}^r := \{D_1, \ldots, D_m\}$;
For $i = 1, \ldots, m$ do
if (1) there exists $D \in \mathcal{D}^r \setminus \{D_i\}$ with $D \sqcap val_r(C \sqcap \mathcal{T}^*(F)) \sqsubseteq D_i$, or
(2) $A_1 \sqcap \ldots \sqcap A_n \sqcap val_r(C) \sqcap F \sqsubseteq \exists r.D_i$
then $\mathcal{D}^r := \mathcal{D}^r \setminus \{D_i\}$;
Define $\widehat{C} := Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.D^r \sqcap \bigsqcap_{D \in \mathcal{D}^r} \exists r.\mathsf{rewrite}(D, \mathcal{T}, val_r(C \sqcap \mathcal{T}^*(F)))$,
where $\forall r.D^r$ is omitted if $D^r = \top$;
Return $\widehat{C}$.

Figure 4: A rewriting algorithm for $\mathcal{ALE}$ using a greedy heuristics.

rewritten into descriptions of size about 10.

For each of these rewritings, the set of defined names computed in each recursion step, i.e., the set $\{A_1, \ldots, A_n\}$ in Figure 4, had size one, i.e., there existed just one (minimal) defined name subsuming $C \sqcap F$. Thus, the negative effect (illustrated by the above example) that too many defined names were conjoined did not occur in our experiments. For the future, we are planning a more thorough empirical evaluation, also comparing the heuristic algorithm with one that actually computes (all) minimal rewritings.

7 Related and future work

In this paper, we have restricted our attention to the minimal rewriting problem. There are, however, also other interesting instances of the general rewriting framework introduced in Section 3.

Rewriting queries using views

The problem of *rewriting queries using views* in DLs, as considered in (Beeri et al., 1997), is one such instance. As source and TBox-DL, that paper considers the language $\mathcal{ALN}$ and its extension $\mathcal{ALCNR}$,[2] i.e., $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALN}$ or $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALCNR}$, and as destination DL $\mathcal{L}_d = \{\sqcap, \sqcup\}$. The rewritings to be computed are maximally contained rewritings, i.e., the relation ρ is subsumption $\sqsubseteq$, and the ordering $\preceq$ is inverse subsumption $\sqsupseteq$. More precisely, (Beeri et al., 1997) is concerned with *total* rewritings, i.e., the rewriting E should no longer contain primitive names. In our framework, total rewritings can be taken into account by modifying the optimality ordering $\preceq$ as follows: $E \preceq E'$ iff (a) E does not contain primitive names and E' contains primitive names, or (b) E and E' do not contain defined names and $E \sqsupseteq E'$. If there exists at least one total rewriting E of C using $\mathcal{T}$, then each minimal (w.r.t. the modified ordering $\preceq$) rewriting of C is total.

Section 3 of (Beeri et al., 1997) contains the following two results:

- For $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALCNR}$ and $\mathcal{L}_d = \{\sqcap, \sqcup\}$, a maximally contained total rewriting is computable. Using the subsumption algorithm for $\mathcal{ALCNR}$, this can also be used to decide whether there exists a total rewriting equivalent to the input concept C.
- If $\mathcal{ALCNR}$ is replaced by $\mathcal{ALN}$, then one can compute a maximally contained total rewriting in exponential time, and existence of a total rewriting equivalent to C can also be decided in exponential time.

It should be noted that, in the conclusion of (Beeri et al., 1997), the authors state that, for $\mathcal{ALN}$, a maximally contained rewriting can be computed in polynomial time; however, the actual complexity bound given in (Beeri et al., 1997), Theorem 3.2, only yields an exponential time bound. This coincides with our

[2] In addition to the constructors in $\mathcal{ALC}$, $\mathcal{ALCNR}$ allows for number restrictions and *role conjunction* ($r_1 \sqcap r_2$).

complexity results given in Section 4.

Translation of concept descriptions

Another interesting instance of the framework, which we intend to investigate in the future, is the *translation of concept descriptions* from one DL into another, i.e., the instance where (i) $\mathcal{L}_s$ and $\mathcal{L}_d$ are *different* DLs; (ii) the TBox is assumed to be empty; and (iii) the binary relation ρ is given as $\equiv$, $\sqsubseteq$, or $\sqsupseteq$. By trying to rewrite an $\mathcal{L}_s$-concept C into an *equivalent* $\mathcal{L}_d$-concept E, one can find out whether C is expressible in $\mathcal{L}_d$. In many cases, such an exact rewriting may not exist. In this case, one can try to approximate C by an $\mathcal{L}_d$-concept from above (below), i.e., find a minimal (maximal) concept description E in $\mathcal{L}_d$ such that $C \sqsubseteq E$ ($E \sqsubseteq C$). An inference service that can compute such rewritings could, for example, support the transfer of knowledge bases between different systems.

References

Baader, F. and Küsters, R. (1998). Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proceedings of the 22nd Annual German Conference on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, Springer Verlag.

Baader, F. and Küsters, R. (2000). Matching in description logics with existential restrictions. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*. Morgan Kaufmann.

Baader, F., Küsters, R., Borgida, A., and McGuinness, D. (1999a). Matching in description logics. *Journal of Logic and Computation*, 9(3).

Baader, F., Küsters, R., and Molitor, R. (1999b). Computing least common subsumers in description logics with existential restrictions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence 1999 (IJCAI'99)*, Morgan Kaufmann.

Baader, F., Küsters, R., and Molitor, R. (1999c). Rewriting concepts using terminologies – revisited. LTCS-Report 99-12, LuFG Theoretical Computer Science, RWTH Aachen, Germany. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

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

Beeri, C., Levy, A. Y., and Rousset, M.-C. (1997). Rewriting queries using views in description logics. In *PODS '97. Proceedings of the Sixteenth ACM SIG-SIGMOD-SIGART Symposium on Principles of Database Systems*, ACM Press.

Borgida, A. and McGuinness, D. L. (1996). Asking queries about frames. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, Morgan Kaufmann.

Cohen, W., Borgida, A., and Hirsh, H. (1992). Computing least common subsumers in description logics. In *Proceedings of the 10th National Conference on Artificial Intelligence*, MIT Press.

Cohen, W. and Hirsh, H. (1994). Learning the CLASSIC description logic: Theoretical and experimental results. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, Morgan Kaufmann.

Donini, F., Lenzerini, M., Nardi, D., Hollunder, B., Nutt, W., and Spaccamela, A. (1992). The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Lutz, C. (1999). Complexity of terminological reasoning revisited. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, Springer Verlag.

Nebel, B. (1990a). *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer Verlag.

Nebel, B. (1990b). Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2).

Sattler, U. (1998). *Terminological knowledge representation systems in a process engineering application*. PhD thesis, RWTH Aachen.

Schmidt-Schauss, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1).