RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

Revised Version of LTCS-Report 99-12: Rewriting Concepts Using Terminologies - Revisited

Franz Baader, Ralf Küsters, and Ralf Molitor

LTCS-Report 00-04

This is a revised version of LTCS-Report 99-12 containing revised proofs of the technical results.

An abridged version of the original report appeared in the *Procedings of the International Conference on Knowledge Representation and Reasoning (KR'2000).*

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

Rewriting Concepts Using Terminologies — Revisited

Franz Baader, Ralf Küsters, and Ralf Molitor
LuFg Theoretical Computer Science, RWTH Aachen
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 languages $\mathcal{ALE}$ and $\mathcal{ALN}$. Finally, we sketch other interesting instances of the framework.

Our interest for the minimal rewriting problem stems from the fact that algorithms for non-standard inferences, such as computing least common subsumers and matchers, usually produce concept descriptions not containing defined names. Consequently, these descriptions are rather large and hard to read and comprehend. First experiments in a chemical process engineering application show that rewriting can reduce the size of concept descriptions obtained as least common subsumers by almost two orders of magnitude.

1 Motivation

In description logics (DL), 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 [11, 5, 3] and computing the least common subsumer [13, 14, 2, 7]. 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 [9, 22] 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 [2, 7]. 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 rewritten into a "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 8). 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 $\mathrm{P} = \mathrm{NP}$) minimal rewritings cannot be computed in polynomial time, even for DLs with a polynomial subsumption problem. In Section 5 and Sec-

Construct name	Syntax	Semantics
Top	$\top$	Δ
Bottom	$\bot$	$\emptyset$
primitive negation ($P \in N_C$)	$\neg P$	$\Delta \setminus P^I$
negation	$\neg C$	$\Delta \setminus C^I$
conjunction	$C \sqcap D$	$C^I \cap D^I$
disjunction	$C \sqcup D$	$C^I \cup D^I$
existential restriction	$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$
value restriction	$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$
number restriction	$(\geq n\ r)$	$\{x \in \Delta \mid \#\{y \in \Delta \mid (x,y) \in r^I\} \geq n\}$
number restriction	$(\leq n\ r)$	$\{x \in \Delta \mid \#\{y \in \Delta \mid (x,y) \in r^I\} \leq n\}$

Table 1: Syntax and semantics of concept descriptions.

tion 6, we then introduce an algorithm for computing (minimal) rewritings for the DL $\mathcal{ALE}$ and $\mathcal{ALN}$, respectively. Finally, we describe a heuristic approach for computing (not necessarily minimal) rewritings in $\mathcal{ALE}$ in Section 7.

2 Preliminaries

We first formally introduce syntax and semantics of the description logics considered in this work as well as the inference problems subsumption and equivalence modulo TBox.

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 Table 2. When talking about an arbitrary DL, we will usually employ the letter $\mathcal{L}$ (possibly with subscript).

The size $|C|$ of a concept description C is defined to be the number of occurrences of concept and role names in C, where $\top$ and $\bot$ are *not* counted. Formally, $|C|$ is inductively defined as follows.

Definition 1 (Size of concept descriptions) *Let C be a concept description built using a set N_C of concept names, a set N_R of role names, and the constructors introduced in Table 1. The* size $|C|$ *is inductively defined by*

Construct name	$\mathcal{FL}_0$	$\mathcal{ALE}$	$\mathcal{ALC}$	$\mathcal{ALN}$
Top		x	x	x
Bottom		x	x	x
primitive negation		x	x	x
negation			x	
conjunction	x	x	x	x
disjunction			x	
existential restrictions		x	x	
value restrictions	x	x	x	x
number restrictions				x

Table 2: The description logics $\mathcal{FL}_0$, $\mathcal{ALE}$, $\mathcal{ALC}$, and $\mathcal{ALN}$.

$$\begin{array}{rcl@{\qquad}rcl}
|\top| & := & 0, & |C \sqcap D| & := & |C| + |D|, \\
|\bot| & := & 0, & |C \sqcup D| & := & |C| + |D|, \\
|P| & := & 1, & |\neg C| & := & |C|, \\
|(\leq n\ r)| & := & n+1, & |\exists r.C| & := & 1 + |C|, \\
|(\geq n\ r)| & := & n+1, & |\forall r.C| & := & 1 + |C|,
\end{array}$$

where $P \in N_C$ denotes a concept name, $r \in N_R$ denotes a role name, and $n \in \mathbb{N}$ is a nonnegative integer.

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

One of the most important inference services provided by DL systems is deciding the subsumption relation between concept descriptions.

Definition 2 (Subsumption and equivalence) *Let C, D be concept descriptions.*
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., $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.*

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

Definition 3 (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. The concept name A is called* defined name *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.

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 [20] for exact definitions). The TBox $\mathcal{T}$ is called *unfolded* iff all defining concepts in $\mathcal{T}$ do not contain defined names [20]. Because of our assumptions on TBoxes, a given TBox $\mathcal{T}$ can always be transformed into an equivalent unfolded TBox $\overline{\mathcal{T}}$ by exhaustively substituting defined names A occurring on the right hand side of a concept definition by their defining concept. However, this unfolding process can lead to an exponential blow-up of the TBox [21].

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 $A \doteq C \in \mathcal{T}$. For a 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 $\mathcal{L}$. The set of primitive names occurring in $\mathcal{T}$ is denoted by N_P and the set of defined names by N_D.

Definition 4 (Subsumption and equivalence modulo TBox) *Let C, D be two concept descriptions and $\mathcal{T}$ a TBox. D* subsumes C modulo $\mathcal{T}$ *(for short $C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. D is* equivalent to C modulo $\mathcal{T}$ *iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$, i.e., $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$.*

3 A general framework for rewriting

In this section, we introduce the general framework for *rewriting using terminologies* as well as the *minimal rewriting problem* that will be discussed in detail in the next two sections. A comparison of the framework to the problem of *rewriting queries using views* [10] and a description of another interesting instance of the framework can be found in the last section.

Definition 5 (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.*

*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

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

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 TBox; and (iii) $\mathcal{L}$-concept descriptions are ordered by size, i.e., $E \preceq E'$ iff $|E| \leq |E'|$.

Note that in contrast to inference tasks like subsumption or consistency, decidability results for the rewriting problem for DLs $\mathcal{L}_s, \mathcal{L}_d, \mathcal{L}_t$ cannot be transferred to sublanguages $\mathcal{L}'_s, \mathcal{L}'_d, \mathcal{L}'_t$. For example, $C = P \sqcup \neg P$ is a rewriting in $\mathcal{ALC}$ using the empty TBox of itself, but there does not exist a rewriting of C in $\mathcal{FL}_0$ using the empty TBox.

Other instances of the framework

Rewriting queries using views The problem of *rewriting queries using views* in DLs, as considered in [10], can be seen as another instance of our general framework. As source and TBox-DL, that paper considers the language $\mathcal{ALN}$ and its extension $\mathcal{ALCNR}$, i.e., $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALN}$ and $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALCNR}$, 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, [10] 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 [10] contains the following two results:

- For $\mathcal{L}_s = \mathcal{L}_t = \mathcal{ALCNR}$[1] and $\mathcal{L}_d = \{\sqcap, \sqcup\}$, 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 $\mathcal{ALN}$, 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.

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

It should be noted that in [10], the authors claim in the conclusion that for $\mathcal{ALN}$ a maximally contained rewriting can be computed in polynomial time, but the complexity bound given in [10], Theorem 3.2 actually yields an exponential time bound. This result coincides with our complexity results given in Section 4.

Translation of concept descriptions Another 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.

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:

Given: An $\mathcal{L}$-concept description C, an $\mathcal{L}$-TBox $\mathcal{T}$, and a nonnegative integer κ.

Question: Does there exists an $\mathcal{L}$-rewriting E in $\mathcal{L}$ 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 $\mathcal{ALC}$.

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

We give a reduction of the NP-complete problem SETCOVER [16] to the minimal rewriting decision problem in $\mathcal{FL}_0$. From this, we obtain NP-hardness for the DLs $\mathcal{ALN}$ and $\mathcal{ALE}$ by simply arguing that the reduction still holds in the presence of the additional constructors.

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 subsets of $\mathcal{U}$, and a nonnegative integer κ.

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

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:

$$\begin{array}{rcl} C_{\mathcal{U}} & := & u_1 \sqcap \ldots \sqcap u_n \\ \mathcal{T}_{\mathcal{F}} & := & \{A_j \doteq \bigsqcap_{u \in F_j} u \mid 1 \leq j \leq m\}. \end{array}$$

Obviously, $C_{\mathcal{U}}$ and $\mathcal{T}_{\mathcal{F}}$ are polynomial in the size of $(\mathcal{U}, \mathcal{F}, \kappa)$. NP-hardness for the minimal rewriting decision problem in $\mathcal{FL}_0$ is an immediate consequence of the following lemma.

Lemma 6 *There exists a minimal rewriting D of $C_{\mathcal{U}}$ using $\mathcal{T}_{\mathcal{F}}$ with $|D| \leq \kappa$ iff there exists a cover of $\mathcal{U}$ with $k \leq \kappa$ sets $F_{i_1}, \ldots, F_{i_k}$ 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 $k \leq \kappa$. □

The reduction of the SETCOVER problem is still valid if we consider the concept $C_{\mathcal{U}}$ as $\mathcal{ALN}$- or $\mathcal{ALE}$-concept description and the TBox $\mathcal{T}_{\mathcal{F}}$ as $\mathcal{ALN}$- or $\mathcal{ALE}$-TBox:

- In $\mathcal{ALE}$, a rewriting E of $C_{\mathcal{U}}$ is of the form $E = A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k \sqcap E'$, where $E' \equiv \top$, e.g., $E' = \forall r.\top$; otherwise, $C_{\mathcal{U}} \not\equiv_{\mathcal{T}} E$. In this case, $E'' := A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k$ is also a rewriting of $C_{\mathcal{U}}$ of the desired form with size $|E''| \leq |E| \leq \kappa$.

- In $\mathcal{ALN}$, a rewriting E of $C_{\mathcal{U}}$ is of the form $E = A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k \sqcap E'$, where $E' \equiv \top$, e.g., $E' = (\geq 0\ r)$; otherwise, $C_{\mathcal{U}} \not\equiv_{\mathcal{T}} E$. In this case, $E'' := A_{i_1} \sqcap \ldots \sqcap A_{i_l} \sqcap v_{l+1} \sqcap \ldots \sqcap v_k$ is also a rewriting of $C_{\mathcal{U}}$ of the desired form with size $|E''| \leq |E| \leq \kappa$.

To sum up, we get the following

Proposition 7 *The minimal rewriting decision problem is NP-hard for $\mathcal{FL}_0$, $\mathcal{ALN}$, and $\mathcal{ALE}$.*

4.2 PSPACE-Hardness for $\mathcal{ALC}$

The following Lemma 9 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 [23], this yields

Proposition 8 *The minimal rewriting decision problem is PSPACE-hard for $\mathcal{ALC}$.*

Lemma 9 *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 E 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 $C \sqsubseteq D$. This implies $C \equiv C \sqcap D$ and $P_1 \sqcap P_2 \sqcap C \equiv P_1 \sqcap P_2 \sqcap C \sqcap D$. 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 are primitive concept names not occurring in C or D, we get $C \equiv C \sqcap D$, and hence $C \sqsubseteq D$.

2. $E = \bot$: Then $P_1 \sqcap P_2 \sqcap C \equiv \bot$. Since P_1, P_2 are primitive concept names, we get $C \equiv \bot$, and hence $C \sqsubseteq D$.

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 not equal A:

   (a) $Q \in \{P_1, P_2\}$: W.l.o.g. let $Q = P_1$. 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 concept names.

   (b) $Q \notin \{A, P_1, P_2\}$: Then $Q \equiv P_1 \sqcap P_2 \sqcap C$. This 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 concept names.

5. $E = \forall r.E'$ with $|E'| = 0$: Then $E' \equiv_{\mathcal{T}} \bot$ or $E' \equiv_{\mathcal{T}} \top$. Assume $E' \equiv_{\mathcal{T}} \top$. This yields $E \equiv_{\mathcal{T}} \top$ in contradiction to minimality of E. Assume $E' \equiv_{\mathcal{T}} \bot$. This implies $\forall r.\bot \equiv_{\mathcal{T}} P_1 \sqcap P_2 \sqcap C$, and hence $\forall r.\bot \sqsubseteq P_1$ in contradiction to the fact that P_1 is a primitive concept name.

6. $E = \exists r.E'$ with $|E'| = 0$: Then $E' \equiv_{\mathcal{T}} \bot$ or $E' \equiv_{\mathcal{T}} \top$. Assume $E' \equiv_{\mathcal{T}} \bot$. This yields $E \equiv_{\mathcal{T}} \bot$ in contradiction to minimality of E. Assume $E' \equiv_{\mathcal{T}} \top$. This implies $\exists r.\top \equiv_{\mathcal{T}} P_1 \sqcap P_2 \sqcap C$, and hence $\exists r.\top \sqsubseteq P_1$ in contradiction to the fact that P_1 is a primitive name.

7. $E = E' \sqcap E''$ or $E = E' \sqcup E''$ or $E = \neg E'$. We prove the claim by induction ont the number of occurences of the constructors $\sqcap, \sqcup$, and $\neg$.

(a) $E = E' \sqcap E''$. Since $|E| \leq 1$, it follows $|E'| = 0$ or $|E''| = 0$. Let, w.l.o.g., $|E'| = 0$. Then $E' \equiv_{\mathcal{T}} \bot$ or $E' \equiv_{\mathcal{T}} \top$. Assume $E' \equiv_{\mathcal{T}} \bot$. This implies $E \equiv_{\mathcal{T}} \bot$ and we get $C \sqsubseteq D$ (see 2). Assume $E' \equiv_{\mathcal{T}} \top$. Then E'' is a minimal rewriting of $P_1 \sqcap P_2 \sqcap C$ using $\mathcal{T}$ of size ≤ 1. By induction, we get $C \sqsubseteq D$.

(b) $E = E' \sqcup E''$. Since $|E| \leq 1$, it follows $|E'| = 0$ or $|E''| = 0$. Let, w.l.o.g., $|E'| = 0$. Then $E' \equiv_{\mathcal{T}} \bot$ or $E' \equiv_{\mathcal{T}} \top$. Assume $E' \equiv_{\mathcal{T}} \top$. This implies $E \equiv_{\mathcal{T}} \top$ and we get a contradiction to E is a rewriting of C using $\mathcal{T}$ (see 3). Assume $E' \equiv_{\mathcal{T}} \bot$. Then E'' is a minimal rewriting of $P_1 \sqcap P_2 \sqcap C$ using $\mathcal{T}$ of size ≤ 1. By induction, we get $C \sqsubseteq D$.

(c) $E = \neg E'$ with $|E'| \leq 1$. Since $E \equiv_{\mathcal{T}} \neg(\neg E)$, let, w.l.o.g., E' be not of the form $\neg E''$, i.e., E' is $\top$ or $\bot$, a concept name, a value- or existential restriction, or a conjunction or disjunction.

i. For $E' = \top$, we get $C \sqsubseteq D$ as in 2.

ii. For $E' = \bot$ we get a contradiction as in 3.

iii. For $E' = Q$ and $Q \notin \{A, P_1, P_2\}$ we get a contradiction as in case 4(b).

iv. For $E' \in \{P_1, P_2\}$, let w.l.o.g. $E' = P_1$. Then we get $P_1 \sqcap P_2 \sqcap \equiv \neg P_1$. This yields a contradiction due to $\neg P_1 \sqsubseteq P_1$.

v. For $E' = A$, we get $\neg(P_1 \sqcap P_2 \sqcap C \sqcap D) \equiv P_1 \sqcap P_2 \sqcap C$. Since $\neg(P_1 \sqcap P_2 \sqcap C \sqcap D) \equiv \neg P_1 \sqcup \neg P_2 \sqcup \neg C \sqcup \neg D$, we get $\neg P_1 \sqcap P_1 \sqcap P_2 \sqcap C$. This yields a contradiction due to $\neg P_1 \sqsubseteq P_1$.

vi. For $E' = \exists r.E''$ it is $E \equiv \forall r.\neg E''$, where $|\neg E''| = 0$. This case has already been treated in 5.

vii. For $E' = \forall r.E''$ it is $E \equiv \exists r.\neg E''$, where $|\neg E''| = 0$. This case has already been treated in 6.

viii. For $E' = E_1 \sqcap E_2$, $|E| \leq 1$ implies $|E_1| = 0$ or $|E_2| = 0$. Let w.l.o.g. $|E_1| = 0$, i.e. $E_1 \equiv \top$ or $E_1 \equiv \bot$. If $E_1 \equiv \top$, we get $E \equiv \neg E_2$, where $\neg E_2$ contains less Boolean constructors than E. Thus, we get by induction $C \sqsubseteq D$. Otherwise, $E_1 \equiv \bot$ yields $E \equiv \top$ and thus, a contradiction as in 3.

ix. For $E' = E_1 \sqcup E_2$, $|E| \leq 1$ implies $|E_1| = 0$ or $|E_2| = 0$. Let w.l.o.g. $|E_1| = 0$, i.e. $E_1 \equiv \top$ or $E_1 \equiv \bot$. If $E_1 \equiv \top$, we get $E \equiv \bot$ and thus, $C \sqsubseteq D$ (see 2). Otherwise, $E_1 \equiv \bot$ implies $E \equiv \neg E_2$. Since $\neg E_2$ contains less Boolean constructors than E, we get by induction $C \sqsubseteq D$. $\square$

The above 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).

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 3: Complexity results for the minimal rewriting decision problem.

4.3 A general upper bound

The following simple algorithm decides whether there exists a rewriting of C using $\mathcal{T}$ of size $\leq \kappa$ in non-deterministic polynomial time, using an oracle for deciding equivalence modulo TBox: First, non-deterministically 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 get the following complexity results for this *restricted equivalence problem*:

$\mathcal{FL}_0$: For $\mathcal{FL}_0$, the restricted equivalence problem can be decided in polynomial time (see Appendix A, Theorem 45).

$\mathcal{ALN}$: If the $\mathcal{ALN}$-TBox is unfolded, then the equivalence problem modulo TBox is decidable in polynomial time (see [12]).

If the TBox is not unfolded, the equivalence problem modulo TBox is in Π_2^p (= coNP$^{\text{NP}}$), (see Appendix B, Theorem 49) and the restricted equivalence problem is in Δ_2^p (= P$^{\text{NP}}$) (see Appendix B, Theorem 50).

$\mathcal{ALE}$: If the $\mathcal{ALE}$-TBox is unfolded, then the equivalence problem modulo TBox is NP-complete [15].

If the TBox is not unfolded, the complexity of the equivalence problem modulo TBox is, to the best of our knowledge, an open problem. On the one hand, the (restricted) equivalence problem is NP-hard (because subsumption in $\mathcal{ALE}$ is NP-hard [15]). On the other hand, since $\mathcal{ALE}$ is a sublanguage of $\mathcal{ALC}$, the (restricted) equivalence problem modulo TBox in $\mathcal{ALE}$ is in PSPACE.

$\mathcal{ALC}$: Equivalence in $\mathcal{ALC}$ is PSPACE-complete [23], even modulo TBoxes [19].

The complexity results for the minimal rewriting decision problem for the DLs under consideration are summarized in Table 3. The upper bounds are obtained from the simple algorithm described above and the complexity results for the restricted equivalence problem.

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 for $\mathcal{ALE}$

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 for the DL $\mathcal{ALE}$. The results are adapted to $\mathcal{ALN}$ in the next section. 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 a hard problem. In addition, the following example shows that, for the DLs under consideration, the number of minimal rewritings of a concept description C using a TBox $\mathcal{T}$ can be exponential in the size of C and $\mathcal{T}$.

Example 10 For a nonnegative integer n, let

$$\begin{array}{rcl} C_n & := & P_1 \sqcap \ldots \sqcap P_n \\ \mathcal{T}_n & := & \{A_i \doteq P_i \mid 1 \leq i \leq n\} \end{array}$$

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

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

It is easy to see that for all $\mathbf{i} \in \{0,1\}^n$, it is $|E_{\mathbf{i}}| = n = |C_n|$, and 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

```
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 $\mathcal{ALE}$.

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 11 (Extension w.r.t. $\mathcal{T}$) *Let C be an $\mathcal{ALE}$-concept description and $\mathcal{T}$ an $\mathcal{ALE}$-TBox. The $\mathcal{ALE}$-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 position 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 $\mathcal{T}$, (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. This problem will be solved in Section 5.1. Before, we

- give an example illustrating the algorithm, and
- explain what the algorithm actually computes.

Example 12 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

$$C^* = P \sqcap Q \sqcap A_2 \sqcap \forall r.P \sqcap \exists r.(A_1 \sqcap P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q))$$

is an extension of C w.r.t. $\mathcal{T}$. 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 substituting $Q \sqcap \neg Q$ by $\bot$, since $\bot$ is the minimal inconsistent concept description. The resulting concept description $\widehat{C} = Q \sqcap A_2 \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$ 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 $\mathcal{T}$. $\square$

On the correctness of the improved algorithm

The following examples show that, on the one hand, there may exist exponentially many essentially different (i.e., not equivalent w.r.t. the empty TBox) extensions of C. On the other hand, for one extension there may exist exponentially many different reductions.

Example 13 *Consider Example 10 again. For each vector* $\mathbf{i} = (i_1, \ldots, i_n) \in \{0,1\}^n$*, the concept description*

$$C^*_{\mathbf{i}} := C_n \sqcap \bigsqcap_{1 \le j \le n, i_j = 1} A_j$$

yields an essentially different extension of C_n*.*

Example 14 *For a nonnegative integer* n*, let*

$$\begin{aligned} C_n &:= \bigsqcap_{1 \le i \le n} \exists r.(A_i \sqcap \exists r.P_i) \sqcap \exists r.(P_i \sqcap \exists r.A_i), \\ \mathcal{T}_n &:= \{A_i \doteq P_i \mid 1 \le i \le n\}. \end{aligned}$$

Further, for $1 \le j \le n$*, let* $C_j^1 := \exists r.(A_j \sqcap \exists r.P_j)$ *and* $C_j^2 := \exists r.(P_j \sqcap \exists r.A_j)$*. It is easy to see that, for all vectors* $\mathbf{i} = (i_1, \ldots, i_n) \in \{1,2\}^n$*, the concept description*

$$C_{\mathbf{i}} := \bigsqcap_{1 \le j \le n} C_j^{i_j}$$

yields a reduction of C_n *w.r.t.* $\mathcal{T}_n$*.* $\square$

As a consequence, the algorithm in Figure 1 should be viewed as a nondeterministic algorithm (with an oracle for the equivalence problem) that first guesses an extension of C w.r.t. $\mathcal{T}$ and then one reduction. We will show that it is correct in the following sense:

Theorem 15
1. *Every possible output of the algorithm is a rewriting of the input concept description* C *using the input TBox* $\mathcal{T}$*.*
2. *The set of all rewritings computed by the algorithm 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 an algorithm (with an oracle for equivalence modulo TBox) for computing one rewriting; however, the computed rewriting need not be minimal. Nevertheless, this opens the way for a heuristic approach to compute "small" (rather than minimal) rewritings in deterministic polynomial time using an oracle for equivalence modulo TBox (cf Section 7). We will also show the following: 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.

Remark 16 *One might think that considering concept descriptions computed by the improved rewriting algorithm and minimal rewritings modulo commutativity and associativity of conjunction should be sufficient, since a* minimal *rewriting should not contain conjuncts of the form $C \sqcap C$. This is not true, however, since we do not count any constructor occurring in C. Hence, $\bot \sqcap \bot$ is as well a minimal rewriting of an inconsistent concept C as $\bot$, whereby a "reasonable" rewriting algorithm would only compute $\bot$ as a (minimal) rewriting. Thus, we have to consider concept descriptions also modulo idempotence of conjunction (since $|F \sqcap \ldots \sqcap F| = |F| = 0$ for $F \in \{\top, \bot\}$), and the equivalence $C \equiv C \sqcap \top$ (since $|C \sqcap \top| = |C|$ for all concept descriptions C).*

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

A reduction of a concept description C w.r.t. a TBox $\mathcal{T}$ has been informally introduced above as a concept description that (i) is equivalent to C modulo $\mathcal{T}$, and (ii) can be obtained from C by removing all redundancies in C. The removal of parts of F will be formalized using the notion of a *subdescription.* Intuitively, D is a subdescription of C, if D can be obtained from C by

- replacing inconsistencies by $\bot$,
- removing some *primitive* concept names, value and existential restrictions on top-level of C, and
- recursively substituting concept descriptions C' occurring in the remaining value and existential restrictions $\forall r.C'/\exists r.C'$ by subdescriptions of C'.

In the first item, we only allow for substitutions of *inconsistent* concepts by $\bot$ in order to make sure that subdescriptions of C always subsume C. This property will be crucial in the proof of completeness of the reduction algorithm introduced below. Unfortunately, if in the formal definition just testing $C \equiv_{\mathcal{T}} \bot$ is not sufficient to describe all subdescriptions that conform to this intuitive definition. For instance, consider the TBox $\mathcal{T} = \{A_1 \doteq \forall r.\neg P\}$ and the concept description $C = A_1 \sqcap \forall r.P$. Obviously, for $D = A_1 \sqcap \forall r.\bot$ it is $C \equiv_{\mathcal{T}} D$, i.e. the concept description occurring in the value restriction on top-level of C is inconsistent in the context of the value restriction occurring on the top-level of the defining concept of A_1. Thus, D should intuitively be a subdescription of C w.r.t. $\mathcal{T}$. However, D would not be a subdescription of C according to the definition described above, because $\bot$ would not be a subdescription of P and hence, the definition does not allow for substituting P by $\bot$ in order to obtain D from C. In the formal definition given below, this problem is solved by taking into account the context F in which C occurs when testing C on inconsistency. In the above example, the context, in which we have to recursively compute a subdescription of P, is given by $\neg P$. Since $P \sqcap \neg P \equiv_{\mathcal{T}} \bot$, we get that $\bot$ is a valid subdescription of P w.r.t. $\mathcal{T}$ and the context $\neg P$.

For the formal definition of a subdescription as well as the formal specification of the reduction algorithm, we need 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 $\mathcal{T}$.[2] The set of all defined names occurring on the top-level of C is denoted by $\mathit{def}(C)$, and the set of all (negated) primitive names occurring on the top-level of C is denoted by $\mathit{prim}(C)$. For an $\mathcal{ALE}$-concept description C and a role name r,

- $\mathit{val}_r(C)$ denotes the concept description occurring in the unique value restriction on the top-level of the $\forall$-normal form of C, where $\mathit{val}_r(C) := \top$ if there is no such value restriction; and
- $\mathit{exr}_r(C)$ 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.

Definition 17 ($\mathcal{ALE}$-Subdescription w.r.t. $\mathcal{T}$ and F) *Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox, F an $\mathcal{ALE}$-concept description, and C an $\mathcal{ALE}$-concept description in $\forall$-normal form that may contain defined names from $\mathcal{T}$. The $\mathcal{ALE}$-concept description $\widehat{C}$ is a* subdescription of C w.r.t. $\mathcal{T}$ and F *iff*

1. $\widehat{C} = C$, *or*
2. $\widehat{C} = \bot$ *and* $C \sqcap F \equiv_{\mathcal{T}} \bot$; *or*
3. $\widehat{C}$ *is obtained from* C *by*
   (a) *removing some (negated) primitive names, value restrictions, and existential restrictions on the top-level of* C,
   (b) *substituting all concept descriptions* D *occurring in the remaining value restrictions* $\forall r.D$ *by subdescriptions* D' *of* D *w.r.t.* $\mathcal{T}$ *and* $\mathit{val}_r(F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_m \sqcap A_1 \ldots \sqcap A_n))$, *where* $\mathit{def}(F) = \{F_1, \ldots, F_m\}$ *and* $\mathit{def}(C) = \{A_1, \ldots, A_n\}$, *and*
   (c) *substituting all concept descriptions* D *occurring in the remaining existential restrictions* $\exists r.D$ *by subdescriptions of* D' *of* D *w.r.t.* $\mathcal{T}$ *and* $\mathit{val}_r(C \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_m \sqcap A_1 \ldots \sqcap A_n))$, *where* $\mathit{def}(F) = \{F_1, \ldots, F_m\}$ *and* $\mathit{def}(C) = \{A_1, \ldots, A_n\}$.

[2]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.

Now, in order to formalize that *all* redundancies have to be eliminated to obtain a reduction of C w.r.t. $\mathcal{T}$, we require subdescriptions to be of minimal size.

Definition 18 ($\mathcal{ALE}$-Reduction w.r.t. $\mathcal{T}$) *Let C be an $\mathcal{ALE}$-concept description in $\forall$-normal form and $\mathcal{T}$ an $\mathcal{ALE}$-TBox. An $\mathcal{ALE}$-concept description $\widehat{C}$ is called* reduction of C w.r.t. $\mathcal{T}$ *iff $\widehat{C}$ is a minimal (w.r.t. $|\cdot|$) subdescription of C w.r.t. $\mathcal{T}$ and $\top$ such that $C \equiv_{\mathcal{T}} \widehat{C}$.*

Consider Example 12 again. The concept description $\widehat{C}$ is a subdescription of the concept description 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).

Disallowing the removal of defined names in the definition of the notion "subdescription" makes sense since in the rewriting algorithm 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^* w.r.t. $\mathcal{T}$ in nondeterministic polynomial time (using an oracle for deciding equivalence modulo $\mathcal{T}$). Intuitively, a reduction $\widehat{C}$ of C 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 on the top-level of C, and $A_1 \sqcap \ldots \sqcap A_m$ the conjunction of the defined names on the top-level of C. Basically, $\widehat{C}$ is obtained form C as follows:

1. Remove the (negated) primitive concepts Q occurring on the top-level of C, if $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} Q$.
2. Remove $\exists r.C_1$ occurring on the top-level of C, if (a) $A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.C' \sqsubseteq_{\mathcal{T}} \exists r.C_1$, or (b) there is another existential restriction $\exists r.C_2$ on top-level such that $A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.C' \sqcap \exists r.C_2 \sqsubseteq_{\mathcal{T}} \exists r.C_1$.
3. Remove $\forall r.C'$ if $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} \forall r.C'$.
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_2 yields a smaller description than the reduction of C_1, we still remove $\exists r.C_1$. If the reductions are of equal size, then we must make a nondeterministic choice between removing the one or the other (see Example 14). This choice is a *don't care* nondeterministic choice, if one is interested in computing just one minimal rewriting, i.e., just one reduction for each extension. If one is interested in computing all minimal rewritings, it is a *don't know* nondeterministic choice, since all possible choices have to be considered.

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 Example 12, 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.

Definition 19 ($\mathcal{ALE}$-Reduction w.r.t. $\mathcal{T}$ and F) *Let C be an $\mathcal{ALE}$-concept description in $\forall$-normal form, $\mathcal{T}$ an $\mathcal{ALE}$-TBox, and F an $\mathcal{ALE}$-concept description. An $\mathcal{ALE}$-concept description $\widehat{C}$ is called* reduction of C w.r.t. $\mathcal{T}$ and F *iff $\widehat{C}$ is a minimal (w.r.t. $|\cdot|$) subdescription of C w.r.t. $\mathcal{T}$ and F such that $C \sqcap F \equiv_{\mathcal{T}} \widehat{C} \sqcap F$.*

The formal specification of the reduction algorithm is given in Figure 2.

In order to prove soundness and completeness of the reduction algorithm, we need the following characterization of subsumption (modulo TBox) in $\mathcal{ALE}$.

Proposition 20

1. *Let C, D be two $\mathcal{ALE}$-concept descriptions in $\forall$-normal form without defined names. It holds that $C \sqsubseteq D$ iff $C \equiv \bot$ or $D \equiv \top$ or*
   - *$prim(D) \subseteq prim(C)$,*
   - *$val_r(C) \sqsubseteq val_r(D)$ for all role names $r \in N_R$, and*
   - *for all $\exists r.D_i \in exr_r(D)$ there exists $\exists r.C_j \in exr_r(C)$ such that $C_j \sqcap val_r(C) \sqsubseteq D_i$.*
2. *Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox and C, D be two $\mathcal{ALE}$-concept descriptions that may contain defined names from $\mathcal{T}$. Let $\{A_1, \ldots, A_n\} = def(C)$ and $\{B_1, \ldots, B_m\} = def(D)$. It holds that $C \sqsubseteq_{\mathcal{T}} D$ iff $C \equiv_{\mathcal{T}} \bot$ or $D \equiv_{\mathcal{T}} \top$ or*
   - *$prim(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m)) \subseteq prim(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$,*
   - *$val_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} val_r(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m))$ for all role names $r \in N_R$, and*
   - *for all $\exists r.D_i \in exr_r(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m))$ there exists $\exists r.C_j \in exr_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ such that $C_j \sqcap val_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} D_i$.*

Proof sketch: Proposition 20(1) is an easy consequence of the characterization of subsumption for $\mathcal{ALE}$ given in [7]. The second part can be reduced to (1) applying the equivalence $C \sqsubseteq_{\mathcal{T}} D$ iff $\mathcal{T}(C) \sqsubseteq \mathcal{T}(D)$. □

```
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 $\{Q_1, \ldots, Q_\ell\} = \mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$;
    For each role name $r \in N_R$
      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 $\mathcal{D}_r$ be a subset of the set

        $\mathcal{C}_r := \{\mathsf{reduce}(C_j, \mathcal{T}, \mathit{val}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))) \mid C_j \in \mathit{exr}_r(C)\}$

      such that
          1. there does not exist $D_1, D_2 \in \mathcal{D}_r$, $D_1 \neq D_2$, with
             $D_1 \sqcap \mathit{val}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} D_2$,
          2. there does not exist $D \in \mathcal{D}_r$ 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 $C_i \in \mathit{exr}_r(C)$, $F \sqcap A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r.\mathit{val}_r(C) \sqsubseteq_{\mathcal{T}} \exists r.C_i$; or
             there exists $D \in \mathcal{D}_r$ with
             $\exists r.D \sqcap \forall r.\mathit{val}(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} \exists r.C_i$, and
          4. the size $\sum_{D \in \mathcal{D}_r}(|D| + 1)$ of the set is minimal among the sizes of all
             subsets of $\mathcal{C}_r$ satisfying (1)–(3).
    Define $\widehat{C}$ := $Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap$
                  $A_1 \sqcap \ldots \sqcap A_m \sqcap$
                  $\bigsqcap_{r \in N_R} \forall r.D^r \sqcap \bigsqcap_{D \in \mathcal{D}_r} \exists r.D$,
    where each value restriction $\forall r.D^r$ is omitted if $D^r = \top$;
  Return $\widehat{C}$.
```

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

In addition, we will need the following lemma. It can easily be proved by induction on the depth of the subdescription.

Lemma 21 *Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox, C an $\mathcal{ALE}$-concept description in $\forall$-normal form, and F an $\mathcal{ALE}$-concept description. Then, for each subdescription $\widehat{C}$ of C w.r.t. $\mathcal{T}$ and F, it holds that $C \sqcap F \sqsubseteq_{\mathcal{T}} \widehat{C}$.*

We now prove soundness and completeness of the reduction algorithm, i.e. we show that each result is a subdescription of C w.r.t. $\mathcal{T}$ and F, and conversely, that each reduction of C w.r.t. $\mathcal{T}$ and F is computed by the algorithm (if $F \not\equiv_{\mathcal{T}} \bot$). Note that formally, for $F \equiv_{\mathcal{T}} \bot$, also $\top$ would be a reduction of C w.r.t. $\mathcal{T}$ and F, which is actually not returned by the algorithm. For this, one would have to introduce a special case in the algorithm, which is omitted here for the sake of simplicity.

Lemma 22 *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.*

Proof of completeness: We show by induction on $\mathsf{depth}(E)$ that, for each reduction E of C w.r.t. $\mathcal{T}$ and F, there exists an output $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$ such that $\widehat{C} = E$ (modulo the given equivalences).

Assume $E = \bot$. Then $C \sqcap F \equiv_{\mathcal{T}} \bot$, and hence $\bot$ is the unique result of $\mathsf{reduce}(C, \mathcal{T}, F)$.

Assume $E \neq \bot$. Then $C \sqcap F \not\equiv_{\mathcal{T}} \bot$; otherwise, since $|\bot| = 0$, E would not be minimal. Thus, $\widehat{C}$ is computed according to the otherwise-part of the algorithm (see Figure 2).

We show that

1. For all results $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$ it is $\mathit{def}(\widehat{C}) = \mathit{def}(E)$ and $\mathit{prim}(\widehat{C}) = \mathit{prim}(E)$.

2. If there exists a value restriction $\forall r.E^r$ on the top-level of E, then there exists a value restriction $\forall r.C^r$ on the top-level of C and a result $\widehat{C^r}$ of the recursive call of the reduction algorithm such that $\widehat{C^r} = E^r$ (modulo the given equivalences). If there exists no value restriction of the form $\forall r.E^r$ on the top-level of E, then there exists a result $\widehat{C}$ with no value restriction of the form $\forall r.C^r$ on the top-level.

3. If $\mathit{exr}_r(E) = \{E_1, \ldots, E_\mu\}$, $\mu > 0$, then $\mathit{exr}_r(C) = \{C_1, \ldots, C_\nu\}$, $\nu \geq \mu$, and there exist results $\widehat{C_{i_j}}$, $1 \leq j \leq \mu$, of the recursive calls of the reduction algorithm such that $\widehat{C_{i_j}} = E_j$ for all $1 \leq j \leq \mu$ and the set $\{\widehat{C_{i_j}} \mid 1 \leq j \leq \mu\}$ satisfies the conditions (1)–(4) in the algorithm. If $\mathit{exr}_r(E) = \emptyset$, then there exists a result $\widehat{C}$ with $\mathit{exr}_r(\widehat{C}) = \emptyset$.

The items (1)–(3) imply that there exists a result $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$ such that $\widehat{C} = E$ (modulo the given equivalences).

Ad (1): By definition of subdescription, $\mathit{def}(C) = \mathit{def}(E)$, and hence $\mathit{def}(\widehat{C}) = \mathit{def}(E)$ for each result $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$.

Let $\mathit{prim}(E) = \{Q'_1, \ldots, Q'_\ell\}$ and $\mathit{def}(C) = \{A_1, \ldots, A_m\}$. It is $\mathit{prim}(E) \subseteq \mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$; otherwise, E would not be minimal. Conversely, by Proposition 20 $E \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$ implies $\mathit{prim}(C) \setminus \mathit{prim}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \subseteq \mathit{prim}(E)$. Thus, we get $\mathit{prim}(E) = \mathit{prim}(\widehat{C})$ for each result $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$.

Ad (2): Let $r \in N_R$. Proposition 20 implies

$$\mathit{val}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \equiv_{\mathcal{T}} \mathit{val}_r(E \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)).$$

Since $val_r(D_1 \sqcap D_2) \equiv val_r(D_1) \sqcap val_r(D_2)$, we get

$$val_r(C) \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \equiv_{\mathcal{T}} val_r(E) \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)).$$

If $val_r(E) = \top$, it is $val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} val_r(C)$ and for each result $\widehat{C}$ there does not exist a value restriction of the form $\forall r.D^r$ on the top-level.

If $val_r(E) \neq \top$, i.e. there exists a unique value restriction of the form $\forall r.E^r$ on the top-level of E, then by definition of subdescription, there exists a unique value restriction of the form $\forall r.C^r$ on the top-level of C. Further, E^r is a subdescription of C^r w.r.t. $\mathcal{T}$ and $val_r(F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m))$, where $def(F) = \{F_1, \ldots, F_n\}$. Let $F^r := val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$. Minimality of E implies that E^r is a minimal subdescription of C^r such that $E^r \sqcap F^r \equiv_{\mathcal{T}} C^r \sqcap F^r$. Thus, E^r is a reduction of C^r w.r.t. $\mathcal{T}$ and F^r.

In order to be able to apply the induction hypothesis, it remains to show that $F^r \not\equiv_{\mathcal{T}} \bot$. Assume $F^r \equiv_{\mathcal{T}} \bot$. Then E would not be minimal due to $|\forall r.E^r| \geq 1$, because removing $\forall r.E^r$ on top-level of E would yield a subdescription E' of C with $E' \sqcap F \equiv_{\mathcal{T}} C \sqcap F$ and $|E'| < |E|$. Thus, $F^r \not\equiv_{\mathcal{T}} \bot$ and by induction, there exists a result $\widehat{C^r}$ of $\mathsf{reduce}(C^r, \mathcal{T}, F^r)$ with $\widehat{C^r} = E^r$.

Ad (3): Let $r \in N_R$, $def(F) = \{F_1, \ldots, F_n\}$ and $F^r := val_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$.

If $exr_r(E) = \emptyset$, then by Proposition 20 $E \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$ implies that for all $C_i \in exr_r(E)$, there exists a $D' \in exr_r(F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m))$ such that $D' \sqcap val_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} C_i$. It is easy to see that the empty set satisfies the conditions (1)–(4) on the set $\mathcal{D}_r$ in the algorithm, i.e. there exists a result $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$ with $exr_r(\widehat{C}) = \emptyset$.

If $exr_r(E) = \{E_1, \ldots, E_\mu\}$, $\mu \geq 1$, then by definition of a subdescription $exr_r(C) = \{C_1, \ldots, C_\nu\}$, $\nu \geq \mu$. In addition, it is $F^r \not\equiv_{\mathcal{T}} \bot$, because otherwise, $C \sqcap F \equiv_{\mathcal{T}} \bot$ in contradiction to our assumption $C \sqcap F \not\equiv_{\mathcal{T}} \bot$.

We have to show that, for all $1 \leq i \leq \mu$, there exists a $C_{j_i} \in exr_r(C)$ such that E_i is a reduction of C_{j_i} w.r.t. $\mathcal{T}$ and F^r, i.e. E_i is a subdescription of C_{j_i} w.r.t. $\mathcal{T}$ and F^r with $E_i \sqcap F^r \equiv_{\mathcal{T}} C_{j_i} \sqcap F^r$ and E_i is minimal among the subdescriptions satisfying this condition.

Let $\mathcal{C}_i := \{C' \in exr_r(C) \mid E_i \text{ is a subdescription of } C'\}$. Since E is a subdescription of C, it is $\mathcal{C}_i \neq \emptyset$. Lemma 21 implies $C' \sqcap F^r \sqsubseteq_{\mathcal{T}} E_i \sqcap F^r$ for all $C' \in \mathcal{C}_i$. The converse subsumption relationship $E_i \sqcap F^r \sqsubseteq_{\mathcal{T}} C' \sqcap F^r$ also holds for all $C' \in \mathcal{C}_i$. To see this, assume $E_i \sqcap F^r \not\sqsubseteq_{\mathcal{T}} C' \sqcap F^r$ for some $C' \in \mathcal{C}_i$. Then by Proposition 20 $E \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$, $E \sqcap F \not\equiv_{\mathcal{T}} \bot$, and $C \sqcap F \not\equiv_{\mathcal{T}} \top$ imply that there exists a $D' \in exr_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m))$ such that

$$D' \sqcap val_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqcap_{\mathcal{T}} C'.$$

By (2) we already know that

$$val_r(E \sqcap F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_n \sqcap A_1 \sqcap \ldots \sqcap A_m)) \equiv_{\mathcal{T}} F^r.$$

This implies

(*) $D' \sqcap F^r \sqsubseteq_{\mathcal{T}} C' \sqcap F^r \sqsubseteq_{\mathcal{T}} E_i \sqcap F^r.$

Due to our assumption $E_i \sqcap F^r \not\sqsubseteq_{\mathcal{T}} C' \sqcap F^r$, it is $D' \neq E_i$. This yields a contradiction to minimality of E: removing $\exists r.E_i$ from the top-level of E would yield an equivalent but smaller subdescription of C, i.e. E would not be a reduction of C w.r.t. $\mathcal{T}$ and F.

Thus, we get that there exists a $C_{j_i} \in exr_r(C)$ with $C_{j_i} \sqcap F^r \equiv_{\mathcal{T}} E_i \sqcap F^r$ and E_i is a subdescription of C_{j_i} w.r.t. $\mathcal{T}$ and F^r. In addition, E_i is minimal: Assume, E_i would not be a minimal subdescription satisfying the equivalence condition. Then there exists a subdescription D' of C_{j_i} w.r.t. $\mathcal{T}$ and F^r with $C_{j_i} \sqcap F^r \equiv_{\mathcal{T}} D' \sqcap F^r$ and $|D'| < |E_i|$. Now, substituting E_i by D' would yield a subdescription E' of C w.r.t. $\mathcal{T}$ and F with $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$ and $|E'| < |E|$ in contradiction to the assumption that E is a reduction of C w.r.t. $\mathcal{T}$ and F.

By induction, we get that there exists a result $\widehat{C_{j_i}}$ of $\mathsf{reduce}(C_{j_i}, \mathcal{T}, F^r)$ with $\widehat{C_{j_i}} = E_i$.

It remains to show that the set $\{C_{j_1}, \ldots, C_{j_\mu}\}$ obtained this way satisfies the conditions (1)–(4) in the algorithm.

- Conditions (1) and (2) are satisfied, because otherwise, E would not be a *minimal* subdescription of C with $E \sqcap F \equiv_{\mathcal{T}} C \sqcap F$: removing a redundant existential restriction would yield a smaller subdescription, because $|\exists r.D| \geq 1$ for all concept descriptions D.
- The third condition is also satisfied; otherwise, by Proposition 20, $E \sqcap F \not\equiv_{\mathcal{T}} C \sqcap F$.
- Condition (4) is satisfied, because otherwise, again E would not be a *minimal* subdescription of C with $E \sqcap F \equiv_{\mathcal{T}} C \sqcap F$.

This completes the proof of completeness.

Proof of soundness: Let $\widehat{C}$ be an output of $\mathsf{reduce}(C, \mathcal{T}, F)$. By construction, $\widehat{C}$ is a subdescription of C w.r.t. $\mathcal{T}$ and F. It remains to show that

1. $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$, and
2. $\widehat{C}$ is a minimal subdescription satisfying (1).

Ad (1): Assume $\widehat{C} = \bot$. Then, by construction, $C \sqcap F \equiv_{\mathcal{T}} \bot$, and hence, $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$.

Let $\widehat{C} \neq \bot$. We show

a) $\widehat{C} \sqsubseteq_{\mathcal{T}} C \sqcap F$ and

b) $\widehat{C} \sqcap F \sqsupseteq_{\mathcal{T}} C \sqcap F$.

The subsumption relationship *b*) is an immediate consequence of Lemma 21. Because of $def(E) = def(\widehat{C})$, the subsumption relationship *a*) is an immediate consequence of the following three items:

(i) for all (negated) primitive concept names $Q \in prim(C)$ it is $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} Q$;

(ii) for all role names $r \in N_R$ it is $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \forall r.val_r(C)$;

(iii) for all role names $r \in N_R$ and all $C_i \in exr_r(C)$ it is $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \exists r.C_i$.

We show (i)–(iii) by induction on $\mathsf{depth}(C)$.

Ad (i): Let $Q \in prim(C)$. If $Q \in prim(\widehat{C})$, then $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} Q$. Otherwise, Q has been removed from the top-level of C. By the definition of $\widehat{C}$ we get that Q occurs on the top-level of $\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)$. Since $def(C) = def(\widehat{C})$, this implies $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} Q$.

Ad (ii): Let $r \in N_R$ and $val_r(C) \neq \top$. If there exists no value restriction of the form $\forall r.C'$ on the top-level of $\widehat{C}$, then by the definition of $\widehat{C}$, we get that $val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} val_r(C)$. If there exists a value restriction of the form $\forall r.C'$ on the top-level of $\widehat{C}$, then C' is a result of $\mathsf{reduce}(val_r(C), \mathcal{T}, val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)))$. By induction, we get

$$C' \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} val_r(C) \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)).$$

This implies

$$\forall r.C' \sqcap val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} val_r(C)$$

and thus, $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \forall r.val_r(C)$.

Ad (iii): Let $r \in N_R$ and $C_i \in exr_r(C)$. By condition (3) on the set $\mathcal{D}_r$ we get that either $F \sqcap A_1 \sqcap \ldots \sqcap A_m \forall r.val_r(C) \sqsubseteq_{\mathcal{T}} \exists r.C_i$ or there exists $D \in exr_r(\widehat{C})$ such that $\exists r.D \sqcap \forall r.val_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \sqsubseteq_{\mathcal{T}} \exists r.C_i$. In both cases $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \exists r.C_i$.

This completes the proof of (1), i.e. we have shown that $C \sqcap F \equiv_{\mathcal{T}} \widehat{C} \sqcap F$.

Ad (2): If $F \equiv_{\mathcal{T}} \bot$, then $C \sqcap F \equiv_{\mathcal{T}} \bot$ and $\bot$ is the unique result of $\mathsf{reduce}(C, \mathcal{T}, F)$. Obviously, $\bot$ is a minimal subdescription of C w.r.t. $\mathcal{T}$ and F satisfying $C \sqcap F \equiv_{\mathcal{T}} \bot$.

If $F \not\equiv_{\mathcal{T}} \bot$, we prove the claim using the completeness of the reduction algorithm. By induction on $\mathsf{depth}(C)$, we get that each result of $\mathsf{reduce}(val_r(C), \mathcal{T}, val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)))$ is a reduction of $val_r(C)$ w.r.t. $\mathcal{T}$ and $val_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$. In particular, all results have the same minimal size. Condition (4) on the set $\mathcal{D}_r$ ensures that all sets satisfying conditions (1)–(3) have the same minimal size. Thus, all results of $\mathsf{reduce}(C, \mathcal{T}, F)$ have the same size. Given that $F \not\equiv_{\mathcal{T}} \bot$, we already know that for each reduction E of C w.r.t. $\mathcal{T}$ and F there exists a result $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$ with $\widehat{C} = E$. Thus, all results have the same minimal size $|E$.

This completes the proof of Lemma 22. □

Remark 23 *It should be noted that the definition of the size of concept descriptions is crucial for completeness of the improved rewriting algorithm: by* $|\bot| := 0$*, we enforce that whenever possible, concept descriptions are substituted by* $\bot$*. Otherwise, the algorithm depicted in Figure 1 would not compute* all *minimal rewritings. To be more precise, if we would count* $\bot$ *as concept name, i.e.,* $|\bot| = 1$*, then the set of all rewritings computed by the algorithm would not contain all minimal rewritings. For instance, consider the concept description*

$C = Q \sqcap \forall r.(P \sqcap \neg P)$ and the TBox $\mathcal{T} = \{A \doteq Q \sqcap \forall r.\neg P\}$. The reduction algorithm as described in Figure 2 would always substitute $P \sqcap \neg P$ by $\bot$, i.e., the only rewritings computed by the minimal rewriting algorithm are $Q \sqcap \forall r.\bot$ and $A \sqcap \forall r.\bot$. However, the minimal rewriting $A \sqcap \forall r.P$ of C using $\mathcal{T}$ would not be computed by the algorithm.

Thus, neither the reduction algorithm nor the rewriting algorithm would be complete. As a consequence, for other definitions of the size of concept descriptions, we would have to

- *weaken the claim of Theorem 15(2): The set of all rewritings computed by the algorithm contains at least one minimal rewriting; or*
- *modify the reduction algorithm appropriately: In case that $C \sqcap F \equiv_{\mathcal{T}} \bot$, not only $\bot$ must be returned, but also each concept description D such that $|D| = |\bot|$, $D \sqcap F \equiv_{\mathcal{T}} \bot$, and D is a subdescription of C w.r.t. $\mathcal{T}$ and F.*

Not only the definition of the size of concept descriptions is crucial for completeness of the algorithm, but also taking into account the context in the definition of a subdescription. The definition of a subdescription of $\mathcal{ALE}$-concept descriptions not containing defined names given in [4] does not restrict the substitution of C by $\bot$ by some test on inconsistency. If we would replace (2) in Definition 17 by

2.' $\widehat{C} = \bot$;

i.e., we would allow for replacing any concept description by $\bot$ in order to obtain a subdescription, then the reduction algorithm would not be complete as shown in the following example.

Example 24 *Consider*

$$\begin{aligned} \mathcal{T} &= \{A_1 \doteq \forall r.P, A_2 \doteq \forall r.\neg P\}, \\ C &= \exists r.(A_2 \sqcap \forall r.P) \sqcap \exists r.(A_1 \sqcap \forall r.Q), \\ E &= \exists r.(A_1 \sqcap \forall r.\bot), \text{ and} \\ E' &= \exists r.(A_2 \sqcap \forall r.\bot). \end{aligned}$$

It is easy to see that $C \equiv_{\mathcal{T}} E$ and $C \equiv_{\mathcal{T}} E'$. Furthermore, E and E' are subdescriptions of C w.r.t. $\mathcal{T}$ and $\top$ with respect to the modified definition of subdescription. The reduction algorithm, however, would only return E'. Due to $\exists r.(A_2 \sqcap \forall r.P) \sqsubseteq_{\mathcal{T}} \exists r.(A_1 \sqcap \forall r.Q)$, the set $\mathcal{D}_r = \{A_2 \sqcap \forall r.\bot\}$ is the unique subset of $\mathcal{C}_r = \{A_2 \sqcap \forall r.\bot, A_1 \sqcap \forall r.Q\}$ satisfying the conditions (1)–(4). This is due to the fact that, in the recursion step, the concept description Q is computed w.r.t. the context P and thus is not *substituted by $\bot$.*

Hence, the reduction algorithm depicted in Figure 2 would not be complete w.r.t. the modified definition of a subdescription. However, this example is not *a counterexample on completeness of the rewriting algorithm, because since $C \not\equiv_{\mathcal{T}} \exists r.P \sqcap \exists r.Q$, C cannot be obtained as an extension of an $\mathcal{ALE}$-concept description w.r.t. $\mathcal{T}$.*

Proof of Theorem 15 for $\mathcal{ALE}$

The first item of Theorem 15 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 $C_i \sqcap \mathit{val}_r(C) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(C)$ (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 $C^* \sqcap F \equiv_{\mathcal{T}} C \sqcap F$ and C^* can be obtained from C by conjoining defined names from $\mathcal{T}$ at some positions in C. In addition, 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 (i) for the recursive definitions, the premise is satisfied, and (ii) there always exists a permutation of $\mathit{exr}_r(C)$ of the desired form.

Lemma 25 *Under the premise of the recursive definition of extensions w.r.t. $\mathcal{T}$ and F induced by E, let, for $r \in N_R$, $\mathit{exr}_r(C) = \{C_1, \dots, C_m\}$ and $\mathit{exr}_r(E) = \{E_1, \dots, E_\ell\}$. Then*

1. *D^r can be recursively defined, i.e., it holds that $\mathit{val}_r(C)$ is in $\forall$-normal form, $\mathit{val}_r(E)$ is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, and $\mathit{val}_r(C) \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} \mathit{val}_r(E) \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$; and*

2. *there exists a permutation $\{j_1, \dots, j_m\}$ of $\{1, \dots, m\}$ such that, for $1 \leq i \leq \ell$,*

$$C_{j_i} \sqcap \mathit{val}_r(C \sqcap F) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

Proof: Ad(1): We have to show

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,
- F 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 $\{Q_1, \dots, Q_k\} := \mathit{prim}(C)$;
  Let $\{A_1, \dots, A_n\} := \mathit{def}(E)$;
  For each role name $r \in N_R$
    Let $D^r$ be the recursively defined extension of
      $\mathit{val}_r(C)$ w.r.t. $\mathcal{T}$ and $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \dots \sqcap A_n))$ induced by $\mathit{val}_r(E)$;
    Let $\mathit{exr}_r(C) = \{C_1, \dots, C_m\}$ and $\mathit{exr}_r(E) = \{E_1, \dots, E_\ell\}$;
    Let $\{j_1, \dots, j_m\}$ be a permutation of $\{1, \dots, m\}$
      such that, for all $1 \leq i \leq \ell$,

        $C_{j_i} \sqcap \mathit{val}_r(C \sqcap F) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \dots \sqcap A_n))$;

    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 $\mathit{val}_r(C \sqcap F)$ induced by $E_i$;
  Then $C^*$ is defined by
```

$$\begin{array}{rcl} C^* & := & Q_1 \sqcap \dots \sqcap Q_k \sqcap A_1 \sqcap \dots \sqcap A_n \sqcap \\ & & \forall r.D^r \sqcap \bigsqcap_{1 \leq i \leq \ell} \exists r.C^*_{j_i} \sqcap \bigsqcap_{\ell+1 \leq i \leq m} \exists r.C_{j_i}, \end{array}$$

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

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

(i) $\mathit{val}_r(C)$ is in $\forall$-normal form and does not contain defined names;

(ii) $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \dots \sqcap A_n))$ does not contain defined names;

(iii) $\mathit{val}_r(E)$ is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \dots \sqcap A_n))$; and

(iv) $\mathit{val}_r(C) \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \dots \sqcap A_n)) \equiv_{\mathcal{T}} \mathit{val}_r(E) \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \dots \sqcap A_n))$.

Ad (i): If there exists no value restriction on the top-level of C, then $\mathit{val}_r(C) = \top$ and $\mathit{val}_r(C)$ is in $\forall$-normal form. Otherwise, since C is in $\forall$-normal form, there exists a unique value restriction of the form $\forall r.C^r$ on the top-level of C and C^r is in $\forall$-normal form. Since C does not contain defined names, C^r does not contain defined names either.

Ad (ii): Since F does not contain defined names and since an unfolded concept description does not contain defined names, $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ does not contain defined names.

Ad (iii): $\mathit{val}_r(E)$ is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, since otherwise, E would not be reduced w.r.t.$\mathcal{T}$ and F.

Ad (iv): By Proposition 20, $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$ implies

$$(*) \;\; \mathit{val}_r(C \sqcap F) \equiv_{\mathcal{T}} \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

Since $\mathit{val}_r(C_1 \sqcap C_2) \equiv_{\mathcal{T}} \mathit{val}_r(C_1) \sqcap \mathit{val}_r(C_2)$ and $C_1 \equiv_{\mathcal{T}} C_2 \sqcap C_3 \Longrightarrow C_1 \sqcap C_3 \equiv_{\mathcal{T}} C_2 \sqcap C_3$, this implies

$$\mathit{val}_r(C) \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} \mathit{val}_r(E) \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

Thus, the preconditions of the recursive definition are satisfied by $\mathit{val}_r(C)$, $\mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, and $\mathit{val}_r(E)$, and by induction, D^r is well-defined.

Ad (2): The permutation can be obtained as follows. Let $i \in \{1, \ldots, \ell\}$. By Proposition 20, there exists $C_k \in \mathit{exr}_r(C \sqcap F)$ such that $C_k \sqcap \mathit{val}_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} E_i$. It is $C_k \in \mathit{exr}_r(C)$; otherwise, we would get $F \sqcap \forall r.\mathit{val}_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} \exists r.E_i$. By (iv) from the first part of this proof, this would imply $F \sqcap \forall r.\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.E_i$. But then removing $\exists r.E_i$ from the top-level of E would yield a concept description E' with $E' \sqcap F \equiv_{\mathcal{T}} E \sqcap F$ and $|E'| < |E|$ in contradiction to E is reduced w.r.t. $\mathcal{T}$ and F.

Thus, there exists $C_k \in \mathit{exr}_r(C)$ with

$$(**) \;\; C_j \sqcap \mathit{val}_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} E_i.$$

We define $j_i := j$. Since $i \in \{1, \ldots, \ell\}$ has been chosen arbitrarily, this should yield the first ℓ elements of the permutation. To see this, it remains to show that for $i, k \in \{1, \ldots, \ell\}$ with $i \neq k$ it is $j_i \neq j_k$. We show

$$(+) \;\; C_{j_i} \sqcap \mathit{val}_r(C \sqcap F) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

By (*) and (**), we get

$$(++) \;\; C_{j_i} \sqcap \mathit{val}_r(C \sqcap F) \sqsubseteq_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)).$$

Assume, $C_{j_i} \sqcap \mathit{val}_r(C \sqcap F) \not\sqsupseteq_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$. By Proposition 20, $C \sqcap F \sqsupseteq_{\mathcal{T}} E \sqcap F$ implies that there exists $D \in \mathit{exr}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ with $D \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} C_{j_i}$. It even holds $D \in \mathit{exr}_r(E)$. Otherwise, we would get

$$A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap \forall r.\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.C_{j_i}.$$

Then (*) would imply

$$A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap \forall r.\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.C_{j_i} \sqcap \forall r.\mathit{val}_r(C \sqcap F),$$

from which (++) implies

$$A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap \forall r.\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \exists r.E_i.$$

As above, this yields a contradiction to E is reduced w.r.t. $\mathcal{T}$ and F.

Thus, there exists $D \in \mathit{exr}_r(E)$ with $D \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqcap_{\mathcal{T}} C_{j_i}$. It is $D = E_i$, because otherwise, we again would get a contradiction to E is reduced w.r.t. $\mathcal{T}$ and F. This implies (+). Using (+), assuming $j_i = j_k$ for some $i \neq k$ again would yield a contradiction. Hence, we have shown $j_i \neq j_k$ for all $i \neq k$ from $\{1, \ldots, \ell\}$.

Now, let $\{j_{\ell+1}, \ldots, j_m\}$ be an arbitrary permutation of $\{1, \ldots, m\} \setminus \{j_1, \ldots, j_\ell\}$. Then $j_1, \ldots, j_m\}$ is a permutation of $\{1, \ldots, m\}$. In order to show that $C_{j_i}^{r,*}$ is well-defined, it remains to show that

(i) C_{j_i} is in $\forall$-normal form and does not contain defined names;

(ii) $\mathit{val}_r(C \sqcap F)$ does not contain defined names;

(iii) E_i is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(C \sqcap F)$; and

(iv) $C_{j_i} \sqcap \mathit{val}_r(F \sqcap C) \equiv_{\mathcal{T}} E_i \sqcap \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$.

Ad (i): Since C is in $\forall$-normal form and does not contain defined names, C_{j_i} is in $\forall$-normal form and does not contain defined names.

Ad (ii): Since C and F do not contain defined names, $\mathit{val}_r(C \sqcap F)$ does not contain defined names.

Ad (iii): E_i is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, because otherwise, E would not be reduced w.r.t. $\mathcal{T}$ and F. By (*) we get that E_i is also reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(C \sqcap F)$.

Ad (iv): This is an immediate consequence of (+).

This completes the proof of Lemma 25. □

In order to complete the proof of Theorem 15, we have to show that each induced extension is an extension of the input concept description, and that the minimal rewriting is a subdescription of the induced extension.

Lemma 26 *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, F 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*

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

Proof: Ad (1): By construction, C^* is obtained from C by conjoining defined names. In particular, for each defined name A conjoined to C, it is $C \sqcap F \sqsubseteq_{\mathcal{T}} A$, and hence $C \sqcap A \sqcap F \equiv_{\mathcal{T}} C \sqcap F$. Consequently, $C^* \sqcap F \equiv_{\mathcal{T}} C \sqcap F$.

Ad (2): By induction on the role depth of E.

If $E = \bot$, then E is a subdescription of C^*.

Otherwise, we get by definition of C^* that $\mathit{def}(E) = \mathit{def}(C^*)$.

We show $\mathit{prim}(E) \subseteq \mathit{prim}(C)$: Let $Q \in \mathit{prim}(E)$. Since $C \sqcap F \equiv_{\mathcal{T}} E \sqcap F$, and C and F do not contain defined names, Proposition 20 implies $Q \in \mathit{prim}(C \sqcap F)$.

It is $Q \notin \mathit{prim}(F)$, since otherwise, E would not be reduced w.r.t. $\mathcal{T}$ and F. Hence, $Q \in \mathit{prim}(C)$.

Assume $\mathit{val}_r(E) \not\equiv \top$, i.e., there exists a value restriction of the form $\forall r.E^r$ on the top-level of E. Since E is reduced, $\forall r.E^r$ is unique. There exists a unique (!) value restriction $\forall r.C^r$ on the top-level of C; otherwise, since C and F do not contain defined names, Proposition 20 implies $\mathit{val}_r(F) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(E)$ in contradiction to the minimality of E. It is $C^r \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} E^r \sqcap \mathit{val}_r(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ (see the proof of Lemma 25). Let D^r be the recursively defined extension of C^r w.r.t. $\mathcal{T}$ and $\mathit{val}_r(F \sqcap (\mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$. By induction, E^r is a subdescription of D^r.

Let $\mathit{exr}_r(E) = \{E_1, \ldots, E_\ell\}$, $\mathit{exr}_r(C) = \{C_1, \ldots, C_m\}$, and $\{j_1, \ldots, j_m\}$ the permutation chosen in the definition of C^*. Since $\mathit{val}_r(C \sqcap F) \equiv_{\mathcal{T}} \mathit{val}_r(E \sqcap F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$, we get by induction that, for $1 \leq i \leq \ell$, E_i is a subdescription of $C^*_{j_i}$.

Thus, E can be obtained from C^* by removing some (negated) primitive names, value restrictions and existential restrictions, and substituting concept descriptions D occurring in remaining value/existential restrictions $\forall r.D/\exists r.D$ by subdescriptions of D, i.e., E is a subdescription of C^*. $\square$

Proof of Theorem 15(2): 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 a minimal rewriting of C w.r.t. $\mathcal{T}$. Let C^* be the extension of C w.r.t. $\mathcal{T}$ and $\top$ induced by E. By Lemma 26, we know that E is a subdescription of C^*. In fact, E is a reduction of C^* w.r.t. $\mathcal{T}$ and $\top$. Assume, E would not be a reduction of C^* w.r.t. $\mathcal{T}$ and $\top$. Since we already know $C^* \equiv_{\mathcal{T}} C$ and E is a subdescription of C^*, E would not be minimal among the subdescriptions D of C^* satisfying $D \equiv_{\mathcal{T}} C^*$. Then, there exists a subdescription E' of C^* with $E' \equiv_{\mathcal{T}} C^* \equiv_{\mathcal{T}} C$ and $|E'| < |E|$. This yields a contradiction to E is a minimal rewriting of C w.r.t. $\mathcal{T}$. Thus, E is a reduction of C^* w.r.t. $\mathcal{T}$ and $\top$. By Lemma 22 we get that there exists a result $\widehat{C}$ of $\mathsf{reduce}(C^*, \mathcal{T}, \top)$ with $\widehat{C} = E$. Thus, the rewriting algorithm computes a rewriting $\widehat{C}$ with $\widehat{C} = E$.

Complexity of the minimal rewriting computation problem for $\mathcal{ALE}$

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

Proposition 27 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 [19], 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. $\square$

6 The minimal rewriting computation problem for $\mathcal{ALN}$

In this section, we adapt the results of the previous section to $\mathcal{ALN}$: we will show that the minimal rewriting algorithm depicted in Figure 1 applied to an $\mathcal{ALN}$-concept description C and an $\mathcal{ALN}$-TBox $\mathcal{T}$ is correct in the sense of Theorem 15.

In $\mathcal{ALN}$, instead of existential restrictions we have to consider number restrictions, which in fact can be treated like primitive names. For the sake of simplicity, we assume that each conjunction occurring in an $\mathcal{ALN}$-concept description C contains, for each role name $r \in N_R$, at most one number restriction of the form $(\geq n\ r)$ resp. $(\leq m\ r)$. Note that this is without loss of generality due to the equivalences

$$\begin{array}{rcl}(\geq n\ r) \sqcap (\geq m\ r) & \equiv & (\geq n\ r) \text{ if } n \geq m, \text{ and} \\ (\leq n\ r) \sqcap (\leq m\ r) & \equiv & (\leq n\ r) \text{ if } n \leq m.\end{array}$$

For technical reasons, we do not allow for number restrictions of the form $(\leq 0\ r)$. On the one hand, this is w.l.o.g. due to the equivalence $(\leq 0\ r) \equiv \forall r.\bot$. On the other hand, since $|(\leq 0\ r)| = 1 = |\forall r.\bot|$, we would have to consider $(\leq 0\ r)$ as a minimal rewriting of $\mathcal{ALN}$-concept descriptions $\forall r.C$ with $C \equiv \bot$. As a consequence, each rewriting obtained from the above normal form yields a set of rewritings by replacing $\forall r.\bot$ by $(\leq 0\ r)$. This set is exponential in the number of occurences of concept descriptions of the form $\forall r.\bot$. But since we already know that the number of minimal rewritings of an $\mathcal{ALN}$-concept description w.r.t. an $\mathcal{ALN}$-TBox $\mathcal{T}$ may be exponential in the size of C and $\mathcal{T}$ independently of this syntactical restrictions (see Example 10), this syntactical restriction does not have any impact on the complexity of the minimal rewriting computation problem in $\mathcal{ALN}$.

In addition to the notations $\mathit{prim}(C)$, $\mathit{def}(C)$, $\mathit{val}_r(C)$, and $\mathcal{T}(C)$, we will need the following notation for $\mathcal{ALN}$-concept descriptions: $\mathit{min}_r(C)$ denotes the minimum of numbers occurring in number restrictions of the form $(\leq n\ r)$ on the top-level of C, where $\mathit{min}_r(C) = \infty$ if there does not exist a number restriction of the form $(\leq n\ r)$ on the top-level of C; and $\mathit{max}_r(C)$ denotes the maximum of numbers occurring in number restrictions of the form $(\geq n\ r)$ on the top-level of C, where $\mathit{max}_r(C) := 0$ if there does not exist a number restriction $(\geq n\ r)$ on the top-level of C.

The definition of extension is obtained from Definition 11 by just substituting $\mathcal{ALE}$ by $\mathcal{ALN}$.

Definition 28 (Extension w.r.t. $\mathcal{T}$) *Let C be an $\mathcal{ALN}$-concept description and $\mathcal{T}$ an $\mathcal{ALN}$-TBox. The $\mathcal{ALN}$-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 position in C.*

The notion "subdescription" has to be slightly modified.

Definition 29 (Subdescription w.r.t. $\mathcal{T}$ and F) *Let $\mathcal{T}$ be an $\mathcal{ALN}$-TBox, F an $\mathcal{ALN}$-concept description, and C an $\mathcal{ALN}$-concept description in $\forall$-normal form that may contain defined names from $\mathcal{T}$. The $\mathcal{ALN}$-concept description $\widehat{C}$ is a* subdescription of C w.r.t. $\mathcal{T}$ and F *iff*

1. *$\widehat{C} = C$, or*
2. *$\widehat{C} = \bot$ and $C \sqcap F \equiv_{\mathcal{T}} \bot$; or*
3. *$\widehat{C}$ is obtained from C by*
   (a) *removing some (negated) primitive names, value restrictions, and number restrictions on the top-level of C, and*
   (b) *substituting all concept descriptions D occurring in the remaining value restrictions $\forall r.D$ by subdescriptions D' of D w.r.t. $\mathcal{T}$ and $\mathrm{val}_r(F \sqcap \mathcal{T}(F_1 \sqcap \ldots \sqcap F_m \sqcap A_1 \ldots \sqcap A_n))$, where $\mathit{def}(F) = \{F_1, \ldots, F_m\}$ and $\mathit{def}(C) = \{A_1, \ldots, A_n\}$.*

Now, the definition of a *reduction $\widehat{C}$ of an $\mathcal{ALN}$-concept description C w.r.t. an $\mathcal{ALN}$-TBox $\mathcal{T}$* is obtained from Definition 18 by just substituting $\mathcal{ALE}$ by $\mathcal{ALN}$.

Definition 30 ($\mathcal{ALN}$-Reduction w.r.t. $\mathcal{T}$ (and F)) *Let C be an $\mathcal{ALN}$-concept description in $\forall$-normal form and $\mathcal{T}$ an $\mathcal{ALN}$-TBox. An $\mathcal{ALN}$-concept description $\widehat{C}$ is called* reduction of C w.r.t. $\mathcal{T}$ *iff $\widehat{C}$ is a minimal subdescription of C w.r.t. $\mathcal{T}$ and $\top$ such that $\widehat{C} \equiv_{\mathcal{T}} C$.*

For an $\mathcal{ALN}$-concept description F, $\widehat{C}$ is called reduction of C w.r.t. $\mathcal{T}$ and F *iff $\widehat{C}$ is a minimal subdescription of C w.r.t. $\mathcal{T}$ and F such that $C \sqcap F \equiv_{\mathcal{T}} \widehat{C} \sqcap F$.*

The algorithm for computing a reduction of an $\mathcal{ALN}$-concept description C w.r.t. an $\mathcal{ALN}$-TBox $\mathcal{T}$ is simpler due to the fact that we now have to consider number restrictions instead of existential restrictions. More precisely, there is no non-deterministic choice between different possible sets of subdescriptions. An intuitive description of the reduction algorithm for $\mathcal{ALN}$ is obtained from the intuitive description for $\mathcal{ALE}$ by substituting the second item by

2.' Remove the number restriction $(\geq \nu\ r)$ resp. $(\leq \mu\ r)$ occurring on the top-level of C, if $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} (\geq \nu\ r)$ resp. $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} (\leq \mu\ r)$.

```
Input: An $\mathcal{ALN}$-concept description $C$ in $\forall$-normal form, an $\mathcal{ALN}$-TBox $\mathcal{T}$ and an
$\mathcal{ALN}$-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 $\{Q_1, \ldots, Q_\ell\} := \mathit{prim}(C) \setminus \mathit{prim}(F \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_m))$;
    For each role name $r \in N_R$
      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 $D^r := \mathsf{reduce}(\mathit{val}_r(C), \mathcal{T}, \mathit{val}(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)))$;
    Define $\widehat{C}$  :=  $Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap$
                              $A_1 \sqcap \ldots \sqcap A_m \sqcap$
                              $\bigsqcap_{r \in N_R} \forall r.D^r \sqcap (\geq \mathit{max}_r(C)\ r) \sqcap (\leq \mathit{min}_r(C)\ r)$,
    where each value restriction $\forall r.D^r$ is omitted if $D^r = \top$,
    each number restriction $(\geq \mathit{max}_r(C)\ r)$ is omitted if
      $\mathit{max}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \geq \mathit{max}_r(C)$, and
    each number restriction $(\leq \mathit{min}_r(C)\ r)$ is omitted if
      $\mathit{min}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \leq \mathit{min}_r(C)$ or $D^r = \bot$.
  Return $\widehat{C}$.
```

Figure 4: The reduction algorithm for $\mathcal{ALN}$.

Note that $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} (\geq \nu\ r)$ iff $\mathit{max}_r(\mathcal{T}(A_1 \sqcap \ldots \sqcap A_m)) \geq \mathit{max}_r(C)$, and $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{T}} (\leq \mu\ r)$ iff $\mathit{min}_r(\mathcal{T}(A_1 \sqcap \ldots \sqcap A_m)) \leq \mathit{min}_r(C)$.

Just as for $\mathcal{ALE}$, we have to consider each concept description w.r.t. the context F in which it occurs. The resulting reduction algorithm for $\mathcal{ALN}$ is depicted in Figure 4. Obviously, the reduction algorithm for $\mathcal{ALN}$ is deterministic, i.e., for an $\mathcal{ALN}$-TBox $\mathcal{T}$ and $\mathcal{ALN}$-concept descriptions C and F, the result $\widehat{C}$ of $\mathsf{reduce}(C, \mathcal{T}, F)$ is unique. For the proof of soundness and completeness we again need an appropriate characterization of subsumption (modulo TBox) in $\mathcal{ALN}$ and a lemma on the subsumption relationships between concept descriptions and their subdescriptions.

Proposition 31 *1. Let C, D be two $\mathcal{ALN}$-concept descriptions without defined names. It holds that $C \sqsubseteq D$ iff $C \equiv \bot$ or $D \equiv \top$ or*

- *$\mathit{prim}(D) \subseteq \mathit{prim}(C)$,*
- *$\mathit{max}_r(D) \leq \mathit{max}_r(C)$ for all role names $r \in N_R$,*
- *$\mathit{min}_r(D) \geq \mathit{min}_r(C)$ or $\mathit{val}_r(C) \equiv \bot$ for all role names $r \in N_R$, and*
- *$\mathit{val}_r(C) \sqsubseteq \mathit{val}_r(D)$ for all role names $r \in N_R$.*

2. Let $\mathcal{T}$ be an $\mathcal{ALN}$-TBox, C, D two $\mathcal{ALN}$-concept descriptions that may contain defined names from $\mathcal{T}$. Let $\{A_1, \ldots, A_n\} = \mathit{def}(C)$ and $\{B_1, \ldots, B_m\} = \mathit{def}(D)$. It holds that $C \sqsubseteq_{\mathcal{T}} D$ iff $C \equiv_{\mathcal{T}} \bot$ or $D \equiv_{\mathcal{T}} \top$ or

- *$prim(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m)) \subseteq prim(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$,*
- *$max_r(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m)) \leq max_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ for all role names $r \in N_R$,*
- *$min_r(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m)) \geq min_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n))$ or $val_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} \bot$ for all role names $r \in N_R$, and*
- *$val_r(C \sqcap \mathcal{T}(A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} val_r(D \sqcap \mathcal{T}(B_1 \sqcap \ldots \sqcap B_m))$ for all role names $r \in N_R$.*

Proof sketch: Proposition 31(1) is an easy consequence of the characterization of subsumption for CLASSIC given in [12] ($\mathcal{ALN}$ is sublanguage of CLASSIC; a more convenient characterization of subsumption for $\mathcal{ALN}$ can be found in [6]). Note that the characterization is complete since we do not allow for number restrictions of the form $(\leq 0\ r)$.

The second part can be reduced to (1) applying the equivalence $C \sqsubseteq_{\mathcal{T}} D$ iff $\mathcal{T}(C) \sqsubseteq \mathcal{T}(D)$. □

The following lemma can easily be proved by induction on the depth of the subdescription.

Lemma 32 *Let $\mathcal{T}$ be an $\mathcal{ALN}$-TBox, C an $\mathcal{ALN}$-concept description in $\forall$-normal form, and F an $\mathcal{ALN}$-concept description. Then, for each subdescription $\widehat{C}$ of C w.r.t. $\mathcal{T}$ and F, it holds that $C \sqcap F \sqsubseteq_{\mathcal{T}} \widehat{C}$.*

We are now equipped to proof soundness and completeness of the reduction algorithm for $\mathcal{ALN}$.

Lemma 33 *Let C be an $\mathcal{ALN}$-concept description in $\forall$-normal form, $\mathcal{T}$ an $\mathcal{ALN}$-TBox, and F an $\mathcal{ALN}$-concept description. Let $\widehat{C}$ be the concept description computed by $\mathsf{reduce}(C, \mathcal{T}, F)$. If $F \not\equiv_{\mathcal{T}} \bot$, then $\widehat{C}$ is the unique reduction of C w.r.t. $\mathcal{T}$ and F.*

Proof: We show that

1. $\widehat{C}$ is a subdescription of C,
2. $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$, and
3. $\widehat{C}$ is minimal among the concept descriptions satisfying (2).

In order to prove (3), we show that

4. for a reduction E of C w.r.t. $\mathcal{T}$ and F, it is $E = \widehat{C}$.

Since the algorithm is deterministic, i.e., the algorithm always computes a unique result $\widehat{C}$, (4) implies that, for $F \not\equiv_{\mathcal{T}} \bot$, reductions of $\mathcal{ALN}$-concept descriptions w.r.t. $\mathcal{T}$ and F are unique.

Ad (1): By construction, $\widehat{C}$ is a subdescription of C.

Ad(2): We show $\widehat{C} \sqcap F \equiv_{\mathcal{T}} C \sqcap F$ by induction on the role depth of $\widehat{C}$.

Assume $\widehat{C} = \bot$. By construction, $C \sqcap F \equiv_{\mathcal{T}} \bot$ and hence, $C \sqcap F \equiv_{\mathcal{T}} \widehat{C} \sqcap F$.

Assume $\widehat{C} \neq \bot$. We show $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$ and $\widehat{C} \sqcap F \sqsupseteq_{\mathcal{T}} C \sqcap F$.

The latter subsumption relationship is an immediate consequence of Lemma 32. It remains to show that $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$. We show

(i) for all (negated) primitive names $Q \in \mathit{prim}(C)$: $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} Q$;

(ii) for all role names $r \in N_R$: $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} \forall r.\mathit{val}_r(C)$; and

(iii) for all role names $r \in N_R$: $\mathit{max}_r(\widehat{C} \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \geq \mathit{max}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$, and $\mathit{min}_r(\widehat{C} \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \leq \mathit{min}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$.

where $\{A_1, \ldots, A_m\} = \mathit{def}(C)$. Since $\mathit{def}(C) = \mathit{def}(\widehat{C})$, this implies $\widehat{C} \sqcap F \sqsubseteq_{\mathcal{T}} C \sqcap F$ (Proposition 31).

Items (i) and (ii) can be shown in exactly the same way as in the proof of Lemma 22. It remains to show the third item.

Let $r \in N_R$ and $\mathit{max}_r(\widehat{C} \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) > 0$, i.e., there exists a number restriction $(\geq \mathit{max}_r(\widehat{C})\ r)$ on the top-level of $\widehat{C}$. Since $\widehat{C}$ is a subdescription of C, $(\geq \mathit{max}_r(\widehat{C})\ r)$ also occurs on the top-level of C and hence, $\mathit{max}_r(\widehat{C} \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \geq \mathit{max}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$. If there exists a number restriction of the form $(\geq \mathit{max}_r(C)\ r)$ on the top-level of C that has been removed in $\widehat{C}$, then by construction, $\mathit{max}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \geq \mathit{max}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$, and hence $\mathit{max}_r(\widehat{C} \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \geq \mathit{max}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$.

The last claim $\mathit{min}_r(\widehat{C} \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \leq \mathit{min}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$ can be shown analogously.

Ad (4): We have to show that for a reduction E of C w.r.t. $\mathcal{T}$ and F, it is $\widehat{C} = E$.

Just as in the proof of Lemma 22 we get that

- $\mathit{def}(E) = \mathit{def}(\widehat{C})$ and $\mathit{prim}(E) = \mathit{prim}(\widehat{C})$, and
- for each value restriction $\forall r.E^r$ on the top-level of E, there exists a value restriction $\forall r.C^r$ on the top-level of $\widehat{C}$ such that $E^r = C^r$.

It remains to consider number restrictions occurring on the top-level of E. Since E is a subdescription of C, these number restrictions also occur on the top-level of C. Since E is a reduction w.r.t. $\mathcal{T}$ and F, we get, for number restrictions $(\geq \nu\ r)/(\leq \mu\ r)$ on the top-level of E, $F \sqcap A_1 \sqcap \ldots \sqcap A_m \not\sqsubseteq_{\mathcal{T}} (\geq \nu\ r)$ and $F \sqcap A_1 \sqcap \ldots \sqcap A_m \not\sqsubseteq_{\mathcal{T}} (\leq \mu\ r)$. Thus, $\mathit{max}_r(\mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m)) \not\geq \mathit{max}_r(C \sqcap \mathcal{T}(F \sqcap A_1 \sqcap \ldots \sqcap A_m))$, and hence, $(\geq \nu\ r)$ occurs on the top-level of $\widehat{C}$. Analogously, we get that also $(\leq \mu\ r)$ occurs on the top-level of $\widehat{C}$.

This completes the proof of Lemma 33. $\square$

Proof of Theorem 15 for $\mathcal{ALN}$

The proof of Theorem 15 for $\mathcal{ALN}$ works in exactly the same way as for $\mathcal{ALE}$, i.e., we show that for a minimal rewriting E of an $\mathcal{ALN}$-concept description C

```
Given: An $\mathcal{ALN}$-TBox, and $\mathcal{ALN}$-concept descriptions $C, F, E$, where
        - $C$ is in $\forall$-normal form and does not contain defined names,
        - $F$ 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 $\{Q_1, \dots, Q_k\} := prim(C)$;
    Let $\{A_1, \dots, A_n\} := def(E)$;
    For $r \in= NR$
      let $D^r$ be the recursively defined extension of $val_r(C)$
        w.r.t. $\mathcal{T}$ and $val_r(F \sqcap \mathcal{T}(A_1 \sqcap \dots \sqcap A_n)$ induced by $val_r(E)$;
    Then $C^*$ is defined by
  $C^* := Q_1 \sqcap \dots \sqcap Q_k \sqcap A_1 \sqcap \dots \sqcap A_n \sqcap$
          $\bigsqcap_{r \in N_R} \big(\forall r.D^r \sqcap (\geq max_r(C)\ r) \sqcap (\leq min_r(C)\ r)\big)$,
    where the value restriction $\forall r.D^r$ (the number restriction $(\geq max_r(C)\ r)/(\leq min_r(C)\ r)$) is omitted if there does not exist such a value restriction (number restriction) on the top-level of $C$.
```

Figure 5: Extensions w.r.t. $\mathcal{T}$ and F induced by E in $\mathcal{ALN}$.

using an $\mathcal{ALN}$-TBox $\mathcal{T}$, we can define an extension C^* of C such that E is a reduction of C^* and hence can be computed using the reduction algorithm for $\mathcal{ALN}$.

The idea underlying the recursive definition of the induced is extension is exactly the same as for $\mathcal{ALE}$. The notions "extension w.r.t. $\mathcal{T}$ and F" and "reduced w.r.t. $\mathcal{T}$ and F" are adopted to $\mathcal{ALN}$.

Just as for $\mathcal{ALE}$, one can show that this definition makes sense, since for the recursive definition of D^r, the premise of the definition is satisfied. The following lemma states the properties of an extension induced by E needed in the proof of Theorem 15. Its proof can be easily adapted from the proof of Lemma 26.

Lemma 34 *Let $\mathcal{T}$ an $\mathcal{ALN}$-TBox, C an $\mathcal{ALN}$-concept description in $\forall$-normal form that does not contain defined names, F an $\mathcal{ALN}$-concept description that does not contain defined names, and E an $\mathcal{ALN}$-concept description that is reduced w.r.t. $\mathcal{T}$ and F with $E \sqcap F \equiv_{\mathcal{T}} C \sqcap F$. If C^* is the concept description defined in Figure 5, then*

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

Now, using the modified notions and results for $\mathcal{ALN}$, the proof of soundness and completeness of the minimal rewriting algorithm for $\mathcal{ALN}$ works in exactly the same way as for $\mathcal{ALE}$.

Complexity of the minimal rewriting computation problem for $\mathcal{ALN}$

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

Proposition 35 *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 C is polynomial (modulo idempotence) in the size of C and $\mathcal{T}$. Since the restricted equivalence problem modulo TBox in $\mathcal{ALN}$ can be decided in Δ_2^p (see Appendix B, Theorem 49), the set of all (essentially different) extensions of C w.r.t. $\mathcal{T}$ can be enumerated using polynomial space. For each extension C^*, the unique reduction w.r.t. $\mathcal{T}$ can again be computed using polynomial space. Hence, by always storing only the smallest rewriting encountered so far, we can compute on minimal rewriting of C using polynomial space. Since, just as for $\mathcal{ALE}$, the number of minimal rewritings may be exponential, the set of all minimal rewritings of C using $\mathcal{T}$ can only be computed in exponential time. $\square$

7 A heuristic algorithm for $\mathcal{ALE}$

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 existential 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.

```
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: 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 (w.r.t. $\sqsubseteq_{\mathcal{T}}$)
      defined names $A_i$ with $C \sqcap F \sqsubseteq_{\mathcal{T}} A_i$;
    Let $\{Q_1, \ldots, Q_\ell\} = prim(C) \setminus prim(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n))$;
    For $r \in N_R$ do
      $D^r :=$ rewrite$(val_r(C), \mathcal{T}, val_r(\mathcal{T}^*(F \sqcap A_1 \sqcap \ldots \sqcap A_n)))$;
      Let $\{D_1, \ldots, D_m\} := exr_r(C)$ 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) $A_1 \sqcap \ldots \sqcap A_n \sqcap F \sqcap val_r(C) \sqsubseteq \exists r.D_i$
        then $\mathcal{D}^r := \mathcal{D}^r \setminus \{D_i\}$;
    Define  $\widehat{C}$  :=  $Q_1 \sqcap \ldots \sqcap Q_\ell \sqcap A_1 \sqcap \ldots \sqcap A_n \sqcap$
                  $\bigsqcap_{r \in N_R} \forall r.D^r \sqcap \bigsqcap_{D \in \mathcal{D}^r} \exists r.$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 6: A rewriting algorithm for $\mathcal{ALE}$ using a greedy heuristics.

The formal specification 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 theorem states correctness and complexity of the rewriting algorithm for $\mathcal{ALE}$ depicted in Figure 6.

Theorem 36 *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* 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 $\mathcal{ALE}$.*

Proof: The first claim can easily be shown by induction on the role depth of C using Proposition 20(2). The second claim is obvious. □

The first item of Theorem 36 implies that each concept description computed by $\mathsf{rewrite}(C, \mathcal{T}, \top)$ is a rewriting of C w.r.t. $\mathcal{T}$. This rewriting, however, need not be minimal as shown in the following example.

Example 37 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 6 produces the rewriting $\widehat{C}_n := A_1 \sqcap \ldots \sqcap A_n$ of size n. The unique minimal rewriting of C_n using $\mathcal{T}_n$ 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 the above 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.

Another difference between a minimal and heuristically computed rewriting is due to the fact that we only consider minimal (w.r.t. subsumption) defined names when adding defined names in the heuristic algorithm.

Example 38 *Let $\mathcal{T} = \{A_1 \doteq P_1 \sqcap P_2, A_2 \doteq P_1 \sqcap P_3, A_3 \doteq P_3\}$ and $C = P_1 \sqcap P_2 \sqcap P_3$. The heuristic algorithm computes the rewriting $\widehat{C} = A_1 \sqcap A_2$. Obviously, this is a minimal rewriting. Compared to the minimal rewriting $E = A_1 \sqcap A_3$, $\widehat{C}$ does not contain more defined names, but instead of A_3, $\widehat{C}$ contains the more specific defined name A_2.*

In principle, the two differences illustrated in the above examples are the only differences between a minimal rewriting and a rewriting computed by the heuristic algorithm. These differences are formally captured by the notion *quasi-subdescription.* This notion differs from the notion 'subdescription' in three aspects: on the one hand, in a quasi-subdescription, we do not allow for substituting a concept description by $\perp$. On the other hand, we allow for conjoining defined names and substituting defined names by more specific defined names.

Definition 39 (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 $\widehat{C}$ is obtained from C by*

- *removing some (negated) primitive names, value and existential restrictions on the top-level of C,*
- *adding some defined names on top-level of C,*

- *substituting some defined names on top-level of C by more specific defined names from $\mathcal{T}$, and*
- *substituting all concept descriptions D occurring in the remaining value/existential restrictions $\forall r.D/\exists r.D$ on the top-level of C by quasi-subdescriptions $\widehat{D}$ of D w.r.t. $\mathcal{T}$.*

Consider Example 37 and Example 38 again. It is easy to see that $\widehat{C_n}$ ($\widehat{C}$) is a quasi-subdescription of E_n (E). Example 37 illustrated that, in order to obtain a rewriting $\widehat{C}$ computed by the algorithm as a quasi-subdescription of a minimal rewriting E, some value restrictions must be removed. The following example shows that one might also have to remove some existential restrictions in $\widehat{C}$ in order to obtain E.

Example 40 Let $C = \forall r.P \sqcap \exists r.P$ and $\mathcal{T} = \{A \doteq \exists r.P\}$. The rewriting computed by the heuristic algorithm is given by $\widehat{C} = \forall r.P \sqcap A$. Since $|\exists r.\top| = |A| = 1$, the concept description $E = \forall r.P \sqcap \exists r.\top$ is also a minimal rewriting of C using $\mathcal{T}$. Obviously, $\widehat{C}$ is a quasi-subdescription of E, since $\widehat{C}$ can be obtained from E by (i) adding A on the top-level of E, and (ii) removing $\exists r.\top$ from the top-level of E.

We can now formalize the relationship between a minimal rewriting and the rewriting obtained from the rewriting algorithm described in Figure 6.

Theorem 41 *Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox, C an $\mathcal{ALE}$-concept description not containing defined names, and E a minimal rewriting of C using $\mathcal{T}$. Then the result $\widehat{C}$ of* $\mathsf{rewrite}(C, \mathcal{T}, \top)$ *is a quasi-subdescription of E.*

Proof: The theorem is an easy consequence of the following

claim: Let $\mathcal{T}$ be an $\mathcal{ALE}$-TBox and $C, E, F_1, F_2, \widehat{C}$ $\mathcal{ALE}$-concept descriptions such that

- C is in $\forall$-normal form and does not contain defined names,
- $F_1 \sqsubseteq_{\mathcal{T}} F_2$,
- E does not contain redundant defined names, i.e., if E' is obtained from E by removing some defined names in E, then $E' \sqcap F_2 \not\equiv_{\mathcal{T}} E \sqcap F_2$,
- $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$, and
- $\widehat{C}$ is the result of $\mathsf{rewrite}(C, \mathcal{T}, F_1)$.

Then either $\widehat{C} = \top$, $\widehat{C} = \bot$, or

1. $prim(\widehat{C}) \subseteq prim(E)$,
2. for all $A \in def(E)$, there exists $A' \in def(\widehat{C})$ with $A' \sqsubseteq_{\mathcal{T}} A$,
3. for each value restriction $\forall r.\widehat{C}'$ on the top-level of $\widehat{C}$, there exists a value restriction $\forall r.E'$ on the top-level of E such that $\widehat{C}'$ is a quasi-subdescription of E', and

4. for each existential restriction $\exists r.\widehat{C}_i$ on the top-level of $\widehat{C}$, there exists an existential restriction $\exists r.E_j$ on the top-level of E such that $\widehat{C_i}$ is a quasi-subdescription of E_j.

Since E is a minimal rewriting of C w.r.t. $\mathcal{T}$, E is reduced w.r.t. $\mathcal{T}$ and $\top$ and does not contain redundant defined names. With $F_1 = F_2 = \top$ the claim implies either $\widehat{C} = \top$, $\widehat{C} = \bot$, or the items (1)–(4) hold for $\widehat{C}$. If $\widehat{C} = \top$, then $C \equiv_{\mathcal{T}} \widehat{C}$ implies $E \equiv_{\mathcal{T}} \top$, and since E is minimal, it is $E = \top$ (modulo idempotence of conjunction). Analogously, we get $E = \bot$ in case that $\widehat{C} = \bot$. Otherwise, the items (1)–(4) ensure that $\widehat{C}$ is a quasi-subdescription of E.

Thus, it remains to prove the claim. If $\widehat{C} = \top$ or $\widehat{C} = \bot$, nothing has to be shown. Let $\widehat{C} \notin \{\top, \bot\}$.

Ad (1): Let $Q \in \mathit{prim}(\widehat{C})$. Assume $Q \notin \mathit{prim}(E)$. Since $\widehat{C} \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$, we get by Proposition 20 that Q occurs on the top-level of $\mathcal{T}(F_2)$ or $\mathcal{T}(A)$ for some $A \in \mathit{def}(E)$. Because of $F_1 \sqcap_{\mathcal{T}} F_2$ and item (2), either case yields a contradiction to the definition of the set $\{Q_1, \ldots, Q_\ell\}$ of (negated) primitive names occurring on the top-level of $\widehat{C}$.

Ad (2): Let $A \in \mathit{def}(E)$. Then $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ implies $C \sqcap F \sqsubseteq_{\mathcal{T}} A$. If $A \in \mathit{def}(\widehat{C})$, nothing has to be shown. If $A \notin \mathit{def}(\widehat{C})$, then A is not a minimal defined name satisfying $C \sqcap F_1 \sqsubseteq_{\mathcal{T}} A$. But then, there exists $A' \in \mathit{def}(\widehat{C})$ with $A' \sqsubset_{\mathcal{T}} A$.

Ad (3): Let $\forall r.^{C^r}$ be a value restriction on the top-level of $\widehat{C}$. By construction, $\widehat{C^r} \neq \top$ and there exists a value restriction $\forall r.C^r$ on the top-level of C such that $\widehat{C^r} = \mathsf{rewrite}(C^r, \mathcal{T}, \mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)))$, where $\{A_1, \ldots, A_n\} = \mathit{def}(\widehat{C})$. We show that

(a) there exists a value restriction $\forall r.E^r$ on the top-level of E,

(b) E^r is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$, where $\{B_1, \ldots, B_m\} = \mathit{def}(E)$,

(c) $C^r \sqcap \mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \equiv_{\mathcal{T}} E^r \sqcap \mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$, and

(d) $\mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$

From these four items, we get by induction that either $\widehat{C^r} = \top$, $\widehat{C^r} = \bot$ or the conditions (1)–(4) are satisfied for $\widehat{C^r}$. Now $\widehat{C^r}$ yields a contradiction to the construction of $\widehat{C}$. If $\widehat{C^r} = \bot$, then (a) and (c) and the precondition that E is reduced w.r.t. $\mathcal{T}$ and F_2 imply that $E^r = \bot$. Hence, $\widehat{C^r}$ is a quasi-subdescription of E^r. Finally, given that the conditions (1)–(4) are satisfied for $\widehat{C^r}$ and E^r, we get that $\widehat{C^r}$ is a quasi-subdescription of E^r w.r.t.$\mathcal{T}$ and $\mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n))$. Thus, in order to prove (3), it remains to show (a)–(d).

Ad (a): Assume that there does not exist a value restriction of the form $\forall r.E^r$ on the top-level of E. Then $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ implies that $\mathit{val}_r(\mathcal{T}^*(F_2 \sqcap$

$B_1 \sqcap \ldots \sqcap B_m)) \equiv_{\mathcal{T}} C^r \sqcap \mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n))$. Because of $F_1 \sqsubseteq_{\mathcal{T}} F_2$ and item (2), this yields $\mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} C^r$ in contradiction to the construction of $\widehat{C}$ and the fact that there exists a value restriction $\forall r.\widehat{C^r}$ on the top-level of $\widehat{C}$.

Ad (b): E^r is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$, since otherwise, E would not be reduced w.r.t. $\mathcal{T}$ and F_2.

Ad (c): This equivalence is a consequence from $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ by Proposition 20.

Ad (d): This subsumption relationship is an immediate consequence of $F_1 \sqsubseteq_{\mathcal{T}} F_2$ and item (2).

In order to complete the proof of Theorem 41 it remains to show item (4).

Ad (4): Let $\exists r.\widehat{C_i}$ be an existential restriction on the top-level of $\widehat{C}$. By construction there exists an existential restriction $\exists r.C_i$ on the top-level of C such that $\widehat{C_i} = \mathsf{rewrite}(C_i, \mathcal{T}, \mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)))$. We show that there exists an existential restriction $\exists r.E_j$ on the top-level of E such that

(a) E_j is reduced w.r.t. $\mathcal{T}$ and $\mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$,

(b) $C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_{\mathcal{T}} E_j \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$, and

(c) $\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_{\mathcal{T}} \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$.

By induction, these items imply that either $\widehat{C_i} = \top$, $\widehat{C_i} = \bot$ or that the items (1)–(4) are satisfied for $\widehat{C_i}$ and E_j. Now, $\widehat{C_i} = \bot$ yields a contradiction to our assumption $\widehat{C} \not\equiv_{\mathcal{T}} \bot$. If $\widehat{C_i} = \top$, then $\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n)) \sqsubseteq_{\mathcal{T}} C_i$. Hence, (b) implies

$$\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} E_j \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2)).$$

By (c), we get $\mathit{val}_r(\mathcal{T}^*(E \sqcap F_2)) \sqcap_{\mathcal{T}} E_j$. Then E_j only contains defined names, because otherwise, E would not be reduced w.r.t. $\mathcal{T}$ and F_2. But this yields a contradiction to the precondition that E does not contain redundant defined names, because if we remove the defined names from E_j, we would obtain a concept description E' with $E' \sqcap F_2 \equiv E \sqcap F_2$ and $|E'| < |E|$. Thus, also $\widehat{C_i} = \top$ yields a contradiction and the conditions (1)–(4) must be satisfied for $\widehat{C_i}$ and E_j. These imply that $\widehat{C_i}$ is a quasi-subdescription of E_j w.r.t. $\mathcal{T}$ and $\mathit{val}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n))$. In order to prove (4), it remains to show items (a)–(c).

It is easy to see that $\mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} \mathit{val}_r(\mathcal{T}^*(A_1 \sqcap \ldots \sqcap A_n))$. Using this subsumption relationship, item (c) is an easy consequence of item (c) in the proof of item (3).

It remains to show (a) and (b). By Proposition 20, $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ implies that there exists $D \in \mathit{exr}_r(\mathcal{T}^*(E \sqcap F_2))$ with $D \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2)) \sqsubseteq_{\mathcal{T}} C_i$.

Assume, $D \notin \mathit{exr}_r(E)$, i.e. $D \in \mathit{exr}_r(\mathcal{T}^*(F_2 \sqcap B_1 \sqcap \ldots \sqcap B_m))$, where $\{B_1, \ldots, B_m\} = \mathit{def}(E)$. Then item (2) and $F_1 \sqsubseteq_{\mathcal{T}} F_2$ imply that there exists $D' \in \mathit{exr}_r(\mathcal{T}^*(F_1 \sqcap A_1 \sqcap \ldots \sqcap A_n))$ with $D' \sqsubseteq_{\mathcal{T}} D$. By construction, C_i would have been removed from $\mathcal{D}_r$ in contradiction to $\widehat{C_i} \in \mathit{exr}_r(\widehat{C})$.

Thus, we get $D = E_j$ for some $E_j \in \mathit{exr}_r(E)$. Now, (c) implies

(*) $E_j \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$.

By Proposition 20, $C \sqcap F_1 \equiv_{\mathcal{T}} E \sqcap F_2$ implies that there exists $D \in \mathit{exr}_r(C \sqcap \mathcal{T}^*(F_1))$ with

$$D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} E_j.$$

Now, (*) implies $D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubseteq_{\mathcal{T}} C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$. Assume, $D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \sqsubset_{\mathcal{T}} C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$. Then C_i would have been removed from $\mathcal{D}_r$ in contradiction to $\widehat{C_i} \in \mathit{exr}_r(\widehat{C})$. Thus, we get $D \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_{\mathcal{T}} C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1))$. Item (c) implies $C_i \sqcap \mathit{val}_r(C \sqcap \mathcal{T}^*(F_1)) \equiv_{\mathcal{T}} E_j \sqcap \mathit{val}_r(\mathcal{T}^*(E \sqcap F_2))$, i.e. item (b). Now, (a) follows in the same way as (b) in the proof of item (3).

This completes the proof of Theorem 41. □

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 6. 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 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 6, 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.

8 Conclusion

In this work, we first introduced a general framework for the rewriting problem. We then investigated the instance of the framework in which one is interested in computing *minimal* rewritings of concept descriptions using defined concepts from a TBox, where all parts are given in the same DL. We showed that the corresponding decision problem is NP-hard for $\mathcal{FL}_0$, $\mathcal{ALE}$, and $\mathcal{ALN}$ and even

PSPACE-hard for $\mathcal{ALC}$. Finally, we introduced a non-deterministic algorithm for computing (minimal) rewritings in $\mathcal{ALN}$ and $\mathcal{ALE}$.

In order to study the usefulness of our minimal rewriting approach, we have implemented a prototype of the rewriting algorithm for $\mathcal{ALE}$ introduced in Section 5 which computes one (possibly non-minimal) rewriting using a greedy heuristics for determining the extension.

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. Springer-Verlag, 1998.

[3] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics 1999 (DL'99)*, number 22 in CEUR-WS, Sweden, 1999. Linköping University. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/.

[4] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference (KR2000)*, pages 261–272. Morgan Kaufmann, 2000.

[5] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.

[6] F. Baader, R. Küsters, and R. Molitor. Structural subsumption considered from an automata theoretic point of view. In E. Franconi, G. De Giacomo, R.M. MacGregor, W. Nutt, and C.A. Welty, editors, *Proceedings of the 1998 International Workshop on Description Logics (DL'98)*, Trento, Italy, 1998. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-11/.

[7] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence 1999 (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.

[8] 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.

[9] F. Baader and U. Sattler. Knowledge representation in process engineering. In L. Padgham, E. Franconi, M. Gehrke, D. McGuinness, and P. Patel-Schneider, editors, *Proceedings of the 1996 International Workshop on Description Logic (DL'96)*, pages 74–78. AAAI Press, 1996.

[10] C. Beeri, A.Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In L. Yuan, editor, *Proceedings of the 16th ACM SIG-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, pages 99–108, New York, NY 10036, USA, 1997. ACM Press.

[11] A. Borgida and D.L. McGuinness. Asking queries about frames. In L.C. Aiello, J. Doyle, and S. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 5th International Conference (KR'96)*, pages 340–349. Morgan Kaufmann, 1996.

[12] 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.

[13] W.W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 754–760. MIT Press, 1992.

[14] W.W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference (KR'94)*, pages 121–132. Morgan Kaufmann, 1994.

[15] F.M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A.M. Spaccamela. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.

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

[17] R. Küsters. Characterizing the semantics of terminological cycles in $\mathcal{ALN}$ using finite automata. LTCS-Report 97-04, LuFG Theoretical Computer Science, RWTH Aachen, 1997. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[18] R. Küsters. Characterizing the semantics of terminological cycles in $\mathcal{ALN}$ using finite automata. In A.G. Cohn, L. Schubert, and S.C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings*

of 6th International Conference (KR'98), pages 499–510. Morgan Kaufmann, 1998.

[19] C. Lutz. 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*, pages 181–200. Springer-Verlag, 1999.

[20] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence.* Springer Verlag, 1990.

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

[22] U. Sattler. *Terminological knowledge representation systems in a process engineering application.* PhD thesis, RWTH Aachen, 1998. Siehe http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[23] M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

A Equivalence modulo TBox for $\mathcal{FL}_0$

Motivated by the minimal rewriting problem, we consider the special case of equivalence modulo TBox in $\mathcal{FL}_0$ in which one concept does not contain defined concepts. It is well-known [21] that in general subsumption (and hence equivalence) modulo TBox in $\mathcal{FL}_0$ is coNP-complete. In the special case, however, deciding equivalence turned out to be still tractable.

In order to show the new complexity result, we use a *concept-centered normal form* [8]: 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 $\forall r_1 \ldots r_n.P_i$ 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 42 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 using the formal languages L_i [8].

Theorem 43 *Let C, D be $\mathcal{FL}_0$-concept descriptions with normal forms $C \equiv \forall L_1.P_1 \sqcap \ldots \sqcap \forall L_k.P_k$ and $D \equiv \forall M_1.P_1 \sqcap \ldots \sqcap \forall M_k.P_k$. Then $C \equiv D$ iff $L_i = M_i$ for $i = 1, \ldots, k$.*

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 coNP-complete [21]: In this case C and D may contain primitive concepts as well as defined concepts. Thus, in order to test the condition from Theorem 43, 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 [21], unfolding an $\mathcal{FL}_0$-TBox $\mathcal{T}$ may yield a TBox $\mathcal{T}'$ of size exponential in the size of $\mathcal{T}$.

In the minimal rewriting problem for $\mathcal{FL}_0$, we are interested in deciding 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

   $$\forall L_1.P_1 \sqcap \ldots \sqcap \forall L_k.P_k,$$

   containing only primitive concepts $P_i \in N_P$, and

2. D is an $\mathcal{FL}_0$-concept description of the form

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

   where $\{A_1, \ldots, A_\ell\} = N_D$ and $\{P_1, \ldots, P_k\} = N_P$.

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 44 Consider the $\mathcal{FL}_0$-TBox

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

The resulting automaton $\mathcal{A}_\mathcal{T}$ is depicted in Figure 7. The states $q_1, \ldots, q_3$ are introduced as intermediate states on the paths from A_1 respectively A_3 to P_2. We have

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

Theorem 45 *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 $1 \le i \le k$.*

$$\mathcal{A}_\mathcal{T}: \quad \begin{tikzcd} A_1 \arrow[r, "r"] \arrow[dr, "\varepsilon"] & q_1 \arrow[r, "s"] & q_2 \arrow[dd, "s"] \\ A_2 \arrow[r, "r"] & P_1 & \\ A_3 \arrow[u, "r"] \arrow[r, "r"] & q_3 \arrow[r, "r"] & P_2 \end{tikzcd}$$

Figure 7: The finite automaton corresponding to $\mathcal{T}$.

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 [6] 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 $r_1 \dots r_n \in M_j$, $n \geq 1$, introduce $n-1$ new states $q_1, \dots, q_{n-1}$ 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 $r_1 \dots r_n$, $n \geq 1$, introduce $n-1$ new states $q_1, \dots, q_{n-1}$ 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 $w \in L_C(P_i)$.

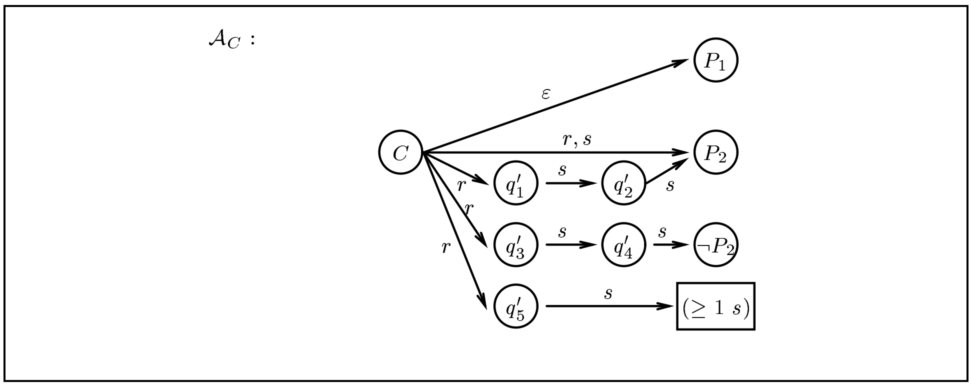

Figure 8: The finite automaton corresponding to C.

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. □

B Equivalence modulo TBox for $\mathcal{ALN}$

We will now consider problem of deciding equivalence modulo TBox in $\mathcal{ALN}$. In [18]it is shown that subsumption modulo *cyclic* TBoxes in $\mathcal{ALN}$is PSPACE-complete. With respect to *acyclic* TBoxes, however, the exact complexity of the subsumption problem is, to our knowledge, an open problem.

In the sequel, we present an improved upper bound for equivalence modulo an acyclic TBox as well as an even more improved upper bound for the special case in which one concept does not contain defined concepts. The result is, just as for $\mathcal{FL}_0$, based on the automata theoretic characterization of subsumption in $\mathcal{ALN}$ [18]. Number restrictions and negated primitive concepts are treated like new primitive concepts. They give rise to new states in the automaton of $\mathcal{T}$ and to additional inclusion conditions in the characterization of subsumption.

Example 46 The $\mathcal{ALN}$-concept description $C := \forall\{\varepsilon\}.P_1 \sqcap \forall\{r, s, rss\}.P_2 \sqcap \forall\{rss\}. \neq P_2 \sqcap \forall\{rs\}.(\geq 1\ s))$ yields the automaton $\mathcal{A}_C$ shown in Figure 8.

For the concept description $A_1 \sqcap A_2 \equiv_{\mathcal{T}} \forall\{r, rss\}.P_1 \sqcap \forall\{r, s\}.P_2$ (see Example) we have $C \sqsubseteq_{\mathcal{T}} A_1 \sqcap A_2$, even though (1) $rss \in L_{\mathcal{A}_{\mathcal{T}}}(A_1, P_1) \cap \overline{L_{\mathcal{P}(\mathcal{A}_C)}(C, P_1)}$, i.e., the automata-theoretic approach does not detect the subsumption relationship.

For an $\mathcal{FL}_0$-concept C, the language $L_{\mathcal{A}_C}(C, P)$ represents exactly those value restrictions on P that subsume the concept C, i.e., $C \sqsubseteq_{\mathcal{T}} \forall W.P$ iff $W \in$

$L_{\mathcal{A}_C}(C,P)$. Since the inconsistent concept $\bot$ is expressible in $\mathcal{ALN}$, the language $L_{\mathcal{A}_C}(C,P)$ is no longer sufficient to capture all these value restrictions. In addition, one must consider so-called *C-excluding words*, i.e., words W such that $C \sqsubseteq \forall W.\bot$. A formal definition of the set $E(C)$ of C-excluding words can be found in [18]. Here, we just illustrate it using our example. Obviously, $rss \in E(C)$ since this word leads to both states P_2 and $\neg P_2$, i.e., any rss-successor of an individual in C must belong both to P_2 and $\neg P_2$, which is impossible. In addition, $rssu \in E(C)$ for all words u, since the existence of an $rssu$-successor would imply the existence of an rss-successor. Finally, at-least restrictions can also force prefixes of rss to belong to $E(C)$: since rs leads to a state containing $(\geq 1\ s)$, every rs-successor of an individual in C also has an rss-successor; however, since $rss \in E(C)$ means that individuals in C cannot have rss-successors, this implies that they cannot have rs-successors.

Since $\forall W.\bot$ is subsumed by $\forall W.P$, C-excluding words yield additional value restrictions that are not explicitly represented by $L_{\mathcal{A}_C}(C,P)$. Thus, in order to represent all value restrictions that are satisfied by instances of C, we consider $L_{\mathcal{A}_C}(C,P) \cup E(C)$ instead of $L_{\mathcal{A}_C}(C,P)$. In our example, the inclusion condition "$L_{\mathcal{A}_\mathcal{T}}(A_1,P_1) \subseteq L_{\mathcal{P}(\mathcal{A}_C)}(C,P_1)$" for P_1 is replaced by "$L_{\mathcal{A}_\mathcal{T}}(A_1,P_1) \subseteq L_{\mathcal{P}(\mathcal{A}_C)}(C,P_1) \cup E(C)$." Consequently, rs no longer violates the inclusion condition for P_1.

The following characterization of subsumption modulo acyclic TBoxes is an easy consequence of the results in [18].

Theorem 47 *Let $\mathcal{T}$ be an acyclic $\mathcal{ALN}$-TBox and C, D two defined concepts in $\mathcal{T}$. Then $C \sqsubseteq_\mathcal{T} D$ iff*

1. *$E(D) \subseteq E(C)$,*
2. *$L_{\mathcal{A}_\mathcal{T}}(D,Q) \subseteq L_{\mathcal{A}_\mathcal{T}}(C,Q) \cup E(C)$ for all (negated) primitive concepts Q in $\mathcal{T}$,*
3. *$L_{\mathcal{A}_\mathcal{T}}(D,(\geq n\ r)) \subseteq \bigcup_{n' \geq n} L_{\mathcal{A}_\mathcal{T}}(C,(\geq n'\ r)) \cup E(C)$ for all $\geq$-number restrictions $(\geq n\ r)$ in $\mathcal{T}$, and*
4. *$L_{\mathcal{A}_\mathcal{T}}(D,(\leq m\ r)) \subseteq \bigcup_{m' \leq m} L_{\mathcal{A}_\mathcal{T}}(C,(\leq m'\ r)) \cup E(C) \cdot r^{-1}$ for all $\leq$-number restrictions $(\leq m\ r)$ in $\mathcal{T}$.*[3]

As an immediate consequence from previous complexity results, we get the following

Lemma 48 *Subsumption modulo TBox in $\mathcal{ALN}$ is coNP-hard and NP-hard.*

Proof: Subsumption modulo TBox in $\mathcal{FL}_0$ is coNP-complete [21]. Hence, subsumption modulo TBox in $\mathcal{ALN}$ is coNP-hard.

In [18] it is shown that for acyclic $\mathcal{ALN}$-TBoxes $\mathcal{T}$ the problem wether a given concept C is inconsistent w.r.t. $\mathcal{T}$ is NP-complete. Since C is inconsistent w.r.t. $\mathcal{T}$ iff $C \sqsubseteq_\mathcal{T} \bot$, we get that subsumption modulo TBox is also NP-hard. □

[3]With $E(C) \cdot r^{-1}$ we denote the set $\{w \mid wr \in E(C)\}$.

As a consequence of Lemma 48 we get that, unless NP=coNP, subsumption modulo TBoxes in $\mathcal{ALN}$ is neither in NP nor in coNP. We now show that subsumption modulo TBox in $\mathcal{ALN}$ is in coNP^{NP}. The corresponding hardness result, however, remains an open problem.

Theorem 49 *Subsumption modulo acyclic TBox in $\mathcal{ALN}$ is in $coNP^{NP}$.*

Proof: Let $\mathcal{T}$ be an $\mathcal{ALN}$-TBox and C, D two defined concepts in $\mathcal{T}$. We introduce an algorithm

- deciding $C \not\sqsubseteq_{\mathcal{T}} D$, and
- that is in NP^{coNP}.

Hence, deciding $C \sqsubseteq_{\mathcal{T}} D$ is in coNP^{NP}.

By Theorem 47 we get that $C \not\sqsubseteq_{\mathcal{T}} D$ iff one of the four conditions is not satisfied. For each of the four conditions, one can decide wether it is satisfied in non-deterministic polynomial time using a coNP-oracle for deciding $w \notin E(C)$ using the following non-deterministic algorithm:

1. Guess a word w of length $\leq |\mathcal{A}_{\mathcal{T}}|$.
2. Test $w \in E(D)$ (resp. $w \in L_{\mathcal{A}_{\mathcal{T}}}(D, Q)$, $L_{\mathcal{A}_{\mathcal{T}}}(D, (\geq n\ r))$, $L_{\mathcal{A}_{\mathcal{T}}}(D, (\leq m\ r))$).
3. Test $w \notin E(C)$ (resp. $w \notin L_{\mathcal{A}_{\mathcal{T}}}(C, Q) \cup E(C)$, $\bigcup_{n' \geq n} L_{\mathcal{A}_{\mathcal{T}}}(C, (\geq n'\ r)) \cup E(C)$, $\bigcup_{m' \leq m} L_{\mathcal{A}_{\mathcal{T}}}(C, (\leq m'\ r)) \cup E(C) \cdot r^{-1}$).

Since $\mathcal{A}_{\mathcal{T}}$ is acyclic, it is sufficient to consider words of length $\leq |\mathcal{A}_{\mathcal{T}}|$. Hence, the first step takes time polynomial in the size of $\mathcal{T}$.

The second step takes non-deterministic polynomial time in the first case (see [17], Theorem 29), and polynomial time in the other cases.

Finally, in order to test $w \notin E(C)$, we have to employ the NP-algorithm deciding $w \in E(C)$ as an oracle (in contrast to the first case in step 2, where the application of the NP-algorithm introduced in [17] still yields an NP-algorithm). Testing $w \notin L_{\mathcal{A}_{\mathcal{T}}}(C, Q) \cup E(C)$ resp. $w \notin \bigcup_{n' \geq n} L_{\mathcal{A}_{\mathcal{T}}}(C, (\geq n'\ r))$ resp. $w \notin \bigcup_{m' \leq m} L_{\mathcal{A}_{\mathcal{T}}}(C, (\leq m'\ r))$ takes polynomial time.

To sum up, deciding wether one of the four conditions of Theorem 47 is not satisfied takes non-deterministic polynomial time using a coNP-oracle for deciding $w \notin E(C)$. Hence, deciding $C \sqsubseteq_{\mathcal{T}} D$ is in coNP^{NP}. □

In the minimal rewriting problem, we are interested in the equivalence $C \equiv_{\mathcal{T}} D$, where the subsumee C does not contain defined concepts from $\mathcal{T}$. Since $C \equiv_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$, we can reduce our attention to the special cases of subsumption modulo TBox in which either the subsumee or the subsumer does not contain defined concepts.

Theorem 50 *Let C be an $\mathcal{ALN}$-concept description, $\mathcal{T}$ an $\mathcal{ALN}$-TBox and D a defined concept in $\mathcal{T}$. If C does not contain defined concepts from $\mathcal{T}$, then $C \equiv_{\mathcal{T}} D$ can be decided in polynomial time using an NP-oracle for deciding $w \in E(D)$.*

Proof: Since C does not contain defined concepts, the corresponding normal form can be computed in polynomial time. Further, $E(C)$ is of the form $L_0 \cdot \Sigma^*$, where L_0 is obtained from the subconcept $\forall L_0.\bot$ on top-level of the normal form of C. We show that

a $D \sqsubseteq_{\mathcal{T}} C$ can be decided in polynomial time using an NP-oracle for deciding $w \in E(D)$, and

b $C \not\sqsubseteq_{\mathcal{T}} D$ can be decided in non-deterministic polynomial time.

By 2. we get that $C \sqsubseteq_{\mathcal{T}} D$ is in coNP. Since coNP $\subseteq$ P^{NP}, this implies the theorem.

ad a): In order to prove a), we show that each of the four conditions of Theorem 47, can be decided in polynomial time using an NP-oracle.

1. We have to show $E(C) \subseteq E(D)$. It is $L_0 \cdot \Sigma^* \subseteq E(D)$ iff $w \in E(D)$ for all $w \in L_0$. Since $|L_0|$ is polynomial in the size of C, $E(C) \subseteq E(D)$ can be decided in time polynomial in the size of C and $\mathcal{T}$ using an NP-oracle for deciding $w \in E(D)$.
2. -4. Just as in the first case, $L_{\mathcal{A}_C}(C, Q)$ (resp. $L_{\mathcal{A}_C}(C, (\geq n\ r))$, $L_{\mathcal{A}_C}(C, (\leq m\ r)))$ has size polynomial in the size of C, and hence, the condition can be tested in polynomial time using an NP-oracle for deciding $w \in E(D)$.

ad b): Assume $C \not\sqsubseteq_{\mathcal{T}} D$. Then one of the four set inclusions in Theorem 47 is violated. In order to test wether one of these conditions is not satisfied, it is sufficient to guess a word w that is contained in the left hand side but not in the right hand side.

1. Guessing a word w of size $\leq |\mathcal{A}_{\mathcal{T}}|$ and testing $w \in E(D)$ takes non-deterministic polynomial time; $w \notin L_0 \cdot \Sigma^*$ can be tested in polynomial time. Hence, $E(D) \not\subseteq E(C)$ can be decided in non-deterministic polynomial time.
2. -4. Just as in the first case, guessing a word w in $L_{\mathcal{A}_{\mathcal{T}}}(D, Q)$ (resp. $L_{\mathcal{A}_{\mathcal{T}}}(D, (\geq n\ r))$, $L_{\mathcal{A}_{\mathcal{T}}}(D, (\leq m\ r)))$ of size $\leq |\mathcal{A}_{\mathcal{T}}|$ takes non-deterministic polynomial time; $w \notin L_{\mathcal{A}_C}(C, Q) \cup L_0 \cdot \Sigma^*$ (resp. $\bigcup_{n' \geq n} L_{\mathcal{A}_C}(C, (\geq n'\ r)) \cup L_0 \cdot \Sigma^a st$, $\bigcup_{m' \leq m} L_{\mathcal{A}_C}(C, (\leq m'\ r)) \cup (L_0 \cdot \Sigma^*) \cdot r^{-1}$) can be tested in polynomial time.