On the Closure of Description Logics under Substitutions *

Jon Haël Brenas[1] Rachid Echahed[1] Martin Strecker[2]

[1] CNRS and University of Grenoble Alpes
[2] Université de Toulouse / IRIT

Abstract. We investigate the extension of Description Logics (DL) with a notion of "substitution". Substitutions naturally arise when reasoning about programs which modify graph structures that are characterized by DLs. They constitute also a means to express concept and role modifications such as addition or deletion of individuals (respectively, pairs of individuals) to or from concepts (respectively, roles). After a formal definition of substitutions, we conduct a systematic study of a wide range of DLs with the purpose of proving or disproving conservativity of an extension of the respective DL with substitutions. The resulting classification is a gauge of the expressiveness of description logics and their adequacy for reasoning about change of graph structures.

1 Introduction

Motivation: Description Logics (DLs) [3] constitute the main class of logics serving as foundation of knowledge representation systems. They have been used for instance as the underlying formal framework for graph and knowledge bases and corresponding querying languages such as SPARQL [17] and OWL [14].

These logics differ by their expressive power and complexity[1]. Besides their static descriptions, Knowledge bases may also be amenable to evolve. Such evolutions can be the result of arbitrary programs. This paper has been motivated by the problem of evolving KBs and the means that may be used to guarantee their formal descriptions. Actually, by using a program logic following a Hoare style, see e.g. [4], one may prove that a KB, say K, which satisfies an assertion Pre, can be modified by a program P and the resulting KB, say K', satisfies an assertion $Post$.

One of the questions explored in this paper is therefore: which DLs that have proven to be adequate for knowledge representation can also be used for reasoning about change, particularly to specify assertions such as Pre and $Post$ and the underlying Hoare calculus? In other words, how do classical DLs behave when used in a Hoare calculus where *substitutions* arise in the computation of assertions? Recall that to every elementary action of a program P, is associated a substitution [10].

A result of this paper is that all the families of description logics considered here can be extended to a decidable program logic, with the aid of substitutions. In some cases, these substitutions are conservative (and can therefore be eliminated within the

* This research has been supported by the *Climt* project (ANR-11-BS02-016)

[1] see, e.g., `http://www.cs.man.ac.uk/~ezolin/dl/`

logic itself), and in some cases, this is impossible. Providing rigorous proofs of this latter fact is one of the main aims of this paper.

Overview of the approach: Although DLs come in numerous flavors allowing to cater to diverse needs in term of expressiveness and complexity, they are tailored to represent "static" knowledge, that is to represent and reason about a given state of knowledge. The need for ways to make knowledge evolve has lead to an increasingly rich field of knowledge update and action [20,11].

The work presented in this paper is inspired by the desire to take advantage of notions of program verification.In a program verification framework, one would compute the weakest precondition associated to a postcondition $Post$, say $wp(Prog, Post)$. $wp(Prog, Post)$ would then be the condition that the original knowledge base has to satisfy to be sure that we reach a knowledge base satisfying the postcondition. Computation of weakest preconditions is done stepwise, starting from the postcondition and by considering the effect of each instruction starting from the last one and going back to the first instruction. The effect of an elementary instruction, say $inst$, on a postcondition, say Q, is reflected by a so-called *substitution*, say θ. Substitutions are classic and central elements of Hoare Logic and the most intuitive way in such a framework to translate into logic the effect of an action. Such an approach is quite unusual in the graph transformations framework though and being able to effectively express substitutions in the case of such transformations is of utmost importance.

The task of proving the correctness of a program can thus, through the use of substitutions, be reduced to the task of determining whether or not a formula is valid. The main issue is that most logics for which there are results on the decidability or the complexity of the validity problem do not explicitly consider substitutions as a constructor. The other way round, most logics with substitutions have received very little attention and there are thus few results on their validity problems.

Note that our goal is to inspect logics and classify them depending on whether or not adding substitutions to them modifies their expressive power. We do not claim to produce efficient ways to prove that a program is correct.

The paper is structured as follows: We recall in Sect. 2 different DLs starting from $\mathcal{ALC}$ and introduce their extensions with substitutions. In Sect. 3 , we provide a family of DLs which are closed under substitutions, whereas Sect. 4 shows that some other DLs are not closed under substitutions. Sect. 5 discusses the case of the logic $\mathcal{ALCO}$ and some of its extensions. We conclude in Sect. 6.

2 Syntax

In this section, we define the syntax of the description logics we consider. We start by the elementary logic $\mathcal{ALC}$. $\mathcal{ALC}$, and all of its extensions, are defined using a signature (**C, R, I**) where **C** is a set of *concept names*, **R** is a set of *role names* and **I** a set of *individuals*.

Definition 1 ($\mathcal{ALC}$ **roles).** *The set of* $\mathcal{ALC}$*-roles is* ***R****.*

Definition 2 ($\mathcal{ALC}$ concepts). *The set of $\mathcal{ALC}$-concepts, C, is defined as:*
$C ::= \bot$ *(empty concept)* $\mid A$ *(concept name)* $\mid \neg\, C$ *(negation)*
$\mid\ C \sqcap C$ *(conjunction)* $\mid \exists R.C$ *(exists)*
where c is a concept name ($\in \mathbf{C}$) and R is a role. As usual, $\top$ is used for $\neg\, \bot$, $C \sqcup D$ for $\neg(\neg C \sqcap \neg D)$ and $\forall R.C$ for $\neg\exists R.\neg C$.

As an example, $\exists R.\ (c \sqcup \forall R.\ c)$ is an $\mathcal{ALC}$-concept satisfied by nodes having an R-neighbour which satisfies c or whose R-neighbours satisfy c.

Concepts are the basic building block of DL formulae. Usually, the problem we are interested in looks like: "Is concept C valid (or satisfiable) given some axioms?". These axioms are split into ABoxes and TBoxes.

Definition 3 (ABox and TBox). *An* individual assertion *IA is defined as:*
$IA ::= a : C$ *(concept assertion)* $\mid a = b$ *(equality)*
$\mid\ a \neq b$ *(inequality)* $\mid (a, b) : R$ *(positive role assertion)*
$\mid\ (a, b) : \neg R$ *(negative role assertion)*
where C and D are concepts, R is a role and a and b are individuals. An ABox *is a finite set of individual assertions. A* TBox *is a finite set of concept inclusions of the form $C \sqsubseteq D$, where C and D are concepts.*

Each extension of $\mathcal{ALC}$ we are going to focus on adds new concept or role constructors. Each constructor is associated to a letter added to the name of the logic[2]. The constructors that we consider in this paper are $\mathcal{Q}$, $\mathcal{O}$, $\mathcal{S}elf$, $\mathcal{I}$, @ and $\mathcal{U}$, defined as follows:

$\mathcal{Q}$	$(\geq n\ R\ C)$	(at least)	$\mathcal{O}$	$\{a\}$	(nominal)
$\mathcal{S}elf$	$\exists S.\mathcal{S}elf$	(local reflexivity)	$\mathcal{I}$	R^-	(inverse role)
@	$@_a C$	(concept assertion)	$\mathcal{U}$	U	(universal role)

where a is an individual, S is a role, different from U, R is a role and C is a concept. $(< n\ R\ C)$ is a shorter way to write $\neg(\geq n\ R\ C)$.

From now on, as there are numerous logics that are considered, we will introduce new notations. $\mathcal{ALC}\mathcal{L}_1 \ldots \mathcal{L}_n[\mathcal{L}_{n+1} \ldots \mathcal{L}_m]$ denotes the family of description logics with all constructors corresponding to $\mathcal{L}_1 \ldots \mathcal{L}_n$ and any combination of those corresponding to $\mathcal{L}_{n+1} \ldots \mathcal{L}_m$.

Definition 4 (Interpretation). *An interpretation $\mathfrak{I}$ is a pair $\{\Delta^\mathfrak{I}, \cdot^\mathfrak{I}\}$ where $\Delta^\mathfrak{I}$ is the universe (set of elements) and $\cdot^\mathfrak{I}$, the valuation, is a function that maps each concept name to a subset of $\Delta^\mathfrak{I}$, each role name to a subset of $\Delta^\mathfrak{I} \times \Delta^\mathfrak{I}$ and each individual to an element of $\Delta^\mathfrak{I}$.*

Below, we define the valuation of roles and concepts:

$\bot^\mathfrak{I}$	$= \emptyset$	$(\neg\, C)^\mathfrak{I}$	$= \Delta^\mathfrak{I} \backslash C^\mathfrak{I}$, *written* $\overline{C^\mathfrak{I}}$
$(C \sqcap D)^\mathfrak{I}$	$= C^\mathfrak{I} \cap D^\mathfrak{I}$	$(\mathrm{U})^\mathfrak{I}$	$= \Delta^\mathfrak{I} \times \Delta^\mathfrak{I}$
$(\exists R.C)^\mathfrak{I}$	$= \{\delta \in \Delta^\mathfrak{I} \mid \#(R^\mathfrak{I}_C(\delta)) \geq 1\}$	$\{a\}^\mathfrak{I}$	$= \{a^\mathfrak{I}\}$
$(\exists R.\mathcal{S}elf)^\mathfrak{I}$	$= \{\delta \in \Delta^\mathfrak{I} \mid (\delta, \delta) \in R^\mathfrak{I}\}$	$(@_a C)^\mathfrak{I}$	$= \Delta^\mathfrak{I}$ *if* $a^\mathfrak{I} \in C^\mathfrak{I}$
$(\geq n\ R\ C)^\mathfrak{I}$	$= \{\delta \in \Delta^\mathfrak{I} \mid \#(R^\mathfrak{I}_C(\delta)\}) \geq n\}$		$\emptyset$ *otherwise*
$(R^-)^\mathfrak{I}$	$= \{(\delta, \delta') \in \Delta^\mathfrak{I} \times \Delta^\mathfrak{I} \mid (\delta', \delta) \in R^\mathfrak{I}\}$		

[2] see, e.g., `http://www.cs.man.ac.uk/~ezolin/dl/`

where $R_C^{\mathfrak{I}}(\delta)$ *is* $\{\delta' \in \Delta | (\delta, \delta') \in R^{\mathfrak{I}} \wedge \delta' \in C^{\mathfrak{I}}\}$ *and* $\#(V)$ *stands for the cardinal of* V.

Definition 5 (Model). *We say that an interpretation* $\mathfrak{I}$ satisfies *an assertion* $a : C$ *(resp.* $a = b$*,* $a \neq b$*,* $(a,b) : R$*,* $(a,b) : \neg R$*) if* $a^{\mathfrak{I}} \in C^{\mathfrak{I}}$ *(resp.* $a^{\mathfrak{I}} = b^{\mathfrak{I}}$*,* $a^{\mathfrak{I}} \neq b^{\mathfrak{I}}$*,* $(a^{\mathfrak{I}}, b^{\mathfrak{I}}) \in R^{\mathfrak{I}}$*,* $(a^{\mathfrak{I}}, b^{\mathfrak{I}}) \notin R^{\mathfrak{I}}$*). We say that an interpretation* $\mathfrak{I}$ *is a* model *of an ABox* $\mathcal{A}$ *if* $\mathfrak{I}$ *satisfies all the assertions of* $\mathcal{A}$*. We say that an interpretation* $\mathfrak{I}$ satisfies *a concept inclusion* $C \sqsubseteq D$ *if* $C^{\mathfrak{I}} \subseteq D^{\mathfrak{I}}$*. We say that an interpretation* $\mathfrak{I}$ *is a* model *of a TBox* $\mathcal{T}$ *if* $\mathfrak{I}$ *satisfies all the concept inclusions of* $\mathcal{T}$*. We say that an interpretation* $\mathfrak{I}$ satisfies *a concept* C *if* $C^{\mathfrak{I}} \neq \emptyset$*.*

Now we introduce the notion of substitution that allows one to modify the valuation of concepts and roles by adding or removing elements.

Definition 6 (Substitution). *Given a role name* R*, a concept name* c *and individuals* a *and* b*, a substitution* θ *is defined as follows:*

$\theta ::= \epsilon$ *(empty substitution)*
$\mid R := R - (a,b)$ *(deletion of role instance)*
$\mid R := R + (a,b)$ *(insertion of role instance)*
$\mid A := A - a$ *(deletion of concept instance)*
$\mid A := A + a$ *(insertion of concept instance)*

Substitutions define an additional concept constructor. We use σ in logic names to signify that substitutions are allowed as constructors. It is important to note that this constructor can only be used to build the concept part of the considered formulae and not in assertions nor in concept inclusions.

Definition 7 ($\mathcal{L}\sigma$ concepts). *Given a logic* $\mathcal{L}$*, the set of* $\mathcal{L}\sigma$*-concepts,* C*, is defined as:*

$C ::= L$ *(concept of* $\mathcal{L}$*)*
$C\theta$ *(explicit substitution)*

where L *is a concept of* $\mathcal{L}$ *and* θ *is a substitution.*

For instance, $C[R := R + (a,b)]$ means that C is satisfied after adding the pair (a,b) to the valuation of R. The interpretation is thus modified accordingly.

Definition 8 (Valuation of explicit substitutions). *Let* A *be a concept name,* a *and* b *be individuals,* C *be a concept and* R *be a role name, the valuation of explicit substitutions is defined as follows:*

- $(C\epsilon)^{\mathfrak{I}} = C^{\mathfrak{I}}$
- $(C[A := A + a])^{\mathfrak{I}} = C^{\mathfrak{K}}$ *where* $\Delta^{\mathfrak{K}} = \Delta^{\mathfrak{I}}$*,* $\forall B \in \mathbf{C}, B \neq A, B^{\mathfrak{K}} = B^{\mathfrak{I}}$*,* $\forall R \in \mathbf{R}, R^{\mathfrak{K}} = R^{\mathfrak{I}}$*,* $\forall o \in \mathbf{I}, o^{\mathfrak{K}} = o^{\mathfrak{I}}$ *and* $A^{\mathfrak{K}} = A^{\mathfrak{I}} \cup \{a^{\mathfrak{I}}\}$*.*
- $(C[A := A - a])^{\mathfrak{I}} = C^{\mathfrak{K}}$ *where* $\Delta^{\mathfrak{K}} = \Delta^{\mathfrak{I}}$*,* $\forall B \in \mathbf{C}, B \neq A, B^{\mathfrak{K}} = B^{\mathfrak{I}}$*,* $\forall R \in \mathbf{R}, R^{\mathfrak{K}} = R^{\mathfrak{I}}$*,* $\forall o \in \mathbf{I}, o^{\mathfrak{K}} = o^{\mathfrak{I}}$ *and* $A^{\mathfrak{K}} = A^{\mathfrak{I}} \cap \overline{\{a^{\mathfrak{I}}\}}$
- $(C[R := R + (a,b)])^{\mathfrak{I}} = C^{\mathfrak{K}}$ *where* $\Delta^{\mathfrak{K}} = \Delta^{\mathfrak{I}}$*,* $\forall A \in \mathbf{C}, A^{\mathfrak{K}} = A^{\mathfrak{I}}$*,* $\forall P \in \mathbf{R}, P \neq R, P^{\mathfrak{K}} = P^{\mathfrak{I}}$*,* $\forall o \in \mathbf{I}, o^{\mathfrak{K}} = o^{\mathfrak{I}}$ *and* $R^{\mathfrak{K}} = R^{\mathfrak{I}} \cup \{(a^{\mathfrak{I}}, b^{\mathfrak{I}})\}$
- $(C[R := R - (a,b)])^{\mathfrak{I}} = C^{\mathfrak{K}}$ *where* $\Delta^{\mathfrak{K}} = \Delta^{\mathfrak{I}}$*,* $\forall A \in \mathbf{C}, A^{\mathfrak{K}} = A^{\mathfrak{I}}$*,* $\forall P \in \mathbf{R}, P \neq R, P^{\mathfrak{K}} = P^{\mathfrak{I}}$*,* $\forall o \in \mathbf{I}, o^{\mathfrak{K}} = o^{\mathfrak{I}}$ *and* $R^{\mathfrak{K}} = R^{\mathfrak{I}} \cap \overline{\{(a^{\mathfrak{I}}, b^{\mathfrak{I}})\}}$

The definition of the valuation for roles and concepts are conserved.

For instance, the valuation of $(\geq\ 3\ R\ C)[C := C + a][R := R - (b, d)]$ is $\{\delta \in \Delta | \#(\{\delta' \in \Delta | (\delta, \delta') \in R^{\mathfrak{L}} \wedge \delta' \in C^{\mathfrak{K}}\}) \geq 3\}$ where $C^{\mathfrak{K}} = C^{\mathfrak{I}} \cup \{a^{\mathfrak{I}}\}$ and $R^{\mathfrak{L}} = R^{\mathfrak{I}} \cap \overline{\{(b^{\mathfrak{I}}, d^{\mathfrak{I}})\}}$.

In this paper, we are interested in how the addition of substitutions to concepts affects the expressive power of a given Description Logic. For now, explicit substitutions can only be used in the definition of a concept (and thus are forbidden in ABoxes and TBoxes). To be more precise, ABoxes and TBoxes are usually allowed but their treatment is not what we are interested in. The reason why it is so is that, in the most expressive logics we consider, they can be simulated into the concept itself. For the least expressive logics, we look at whether or not allowing substitutions in ABoxes and TBoxes is sufficient to close the gap between the logics with and without substitutions.

To be able to evaluate the expressiveness of a logic, we define an equivalence relation on logics.

Definition 9 (Equivalence). *Two concepts C and D are said to be* equivalent *if, for every interpretation $\mathfrak{I}$, $C^{\mathfrak{I}} = D^{\mathfrak{I}}$. A logic $\mathcal{L}_1$ is* at most as expressive as *a logic $\mathcal{L}_2$ wrt. concepts written $\mathcal{L}_1 \leq_C \mathcal{L}_2$ if every concept in $\mathcal{L}_1$ has an equivalent concept in $\mathcal{L}_2$. Two logics $\mathcal{L}_1$ and $\mathcal{L}_2$ are* concept-equivalent *if $\mathcal{L}_1 \leq_C \mathcal{L}_2$ and $\mathcal{L}_2 \leq_C \mathcal{L}_1$.*

Definition 10 (Closure under substitutions). *A logic L is said to be* closed under substitutions *if L and Lσ are concept-equivalent.*

3 DLs Closed under Substitutions

In this section we prove that DLs in $\mathcal{ALCU}\mathcal{O}[\mathcal{QI}@\mathcal{S}elf]$ are closed under substitutions. Let us notice first that in $\mathcal{ALCQUIO}@\mathcal{S}elf$ some constructs are rendundant. In particular, when a DL includes $\mathcal{O}$ (nominals), assertions can be rewritten as concept inclusions. When a DL includes $\mathcal{U}$ (universal role), concept inclusions can be rewritten as mere concepts. Additionally, as the concept $@_a C$ is equivalent to the assertion $a\ :\ C$, all $@$-concepts can be removed in presence of both $\mathcal{O}$ and $\mathcal{U}$. Finally, in presence of both $\mathcal{O}$ and $@$, all ABox assertions can be rewritten as concepts.

As both $\mathcal{O}$ and $\mathcal{U}$ are part of $\mathcal{ALCQUIO}@\mathcal{S}elf$, we consider for this chapter that the TBox and the ABox are empty.

Theorem 1. *$\mathcal{ALCQUIO}@\mathcal{S}elf$ is closed under substitutions.*

The proof of Theorem 1 is done by providing a terminating rewriting system consisting of rules which translate concepts of $\mathcal{ALCQUIO}@\mathcal{S}elf\sigma$ into concepts of $\mathcal{ALCQUIO}@\mathcal{S}elf$ (without substitutions). The idea is to associate to each possible concept constructor Φ coupled with a substitution constructor σ an equivalent substitution-free concept Φ'. The rules thus have the form: $\Phi\sigma \rightsquigarrow \Psi$. Ψ is a concept of $\mathcal{ALCQUIO}@\mathcal{S}elf\sigma$ equivalent to $\Phi\sigma$ such that substitutions are applied to "simpler" concepts. The rules are made so that they are orthogonal, thus ensuring that the rewriting system is confluent. Furthermore, we have shown that the considered rules constitute a

terminating rewrite system. Therefore each formula $\Phi\sigma$ rewrites to a unique substitution free normal form Φ' ($\Phi\sigma \rightsquigarrow^* \phi'$).

Reporting all rules in here would be burdensome. Thus, to illustrate our point, we give the example of a rule $\Phi\sigma \rightsquigarrow \Psi$ for the $\mathcal{ALCQUIO@}\mathcal{S}elf\sigma$-concept where the left-hand side $\Phi\sigma$ is such that $\phi = (\geq \ n \ R \ C)$ and $\sigma = [R := R - (a, b)]$. The right-hand side Ψ of the rule is a concept equivalent to $\Phi\sigma$, that is to say a concept such that for every node μ, μ satisfies Ψ iff μ satisfies $(\geq \ n \ R \ C)[R := R - (a, b)]$. A possible concept for Ψ is given below :

$$\begin{aligned}&((\{a\} \sqcap \exists U.(\{b\} \ \sqcap C[R := R - (a, b)]) \sqcap \exists R.\{b\}) \Rightarrow \\ &\qquad\qquad (\geq \ (n + 1) \ R \ C[R := R - (a, b)])) \\ &\sqcap ((\neg\{a\} \sqcup \forall U.(\neg\{b\} \sqcup \neg C[R := R - (a, b)]) \sqcup \ \forall R.\neg\{b\}) \Rightarrow \\ &\qquad\qquad (\geq \ n \ R \ C[R := R - (a, b)])).\end{aligned}$$

- If the current node is a and b is an R-neighbour of a satisfying $C[R := R - (a, b)]$ (that is if $\{a\} \sqcap \exists U.(\{b\} \ \sqcap C[R := R - (a, b)]) \sqcap \exists R.\{b\}$ is satisfied), then a will actually lose exactly one R-neighbour satisfying C after the substitution and it will thus have at least n remaining R-neighbours satisfying C after the substitution if it had at least $n + 1$ before and thus $(\geq \ (n + 1) \ R \ C[R := R - (a, b)])$.
- If either the current node is not a, b does not satisfy $C[R := R - (a, b)]$ or b is not an R-neighbour (that is if $\neg\{a\} \sqcup \forall U.(\neg\{b\} \sqcup \neg C[R := R - (a, b)]) \sqcup \forall R.\neg\{b\}$ is satisfied) then the removal of the edge (a, b) from R doesn't affect the current node and thus $(\geq \ n \ R \ C[R := R - (a, b)])$ must be satisfied.

By following the same proof technique used for Theorem 1, we can show that the following DLs are also closed under substitutions.

Corollary 1. *The logics of $\mathcal{ALCUO}[\mathcal{QI@}\mathcal{S}elf]$ are closed under substitutions.*

One can easily see from the one rule presented that the rewriting system risks causing an exponential blow-up in the size of the concept whose satisfiability is tested. In a worst case scenario, this could change the complexity of the satisfiability problem to 2NEXPTIME.

Another noteworthy observation is that the constructors @ and U are both used to access remote nodes. A quick check of the rules shows that all occurrences of U in $\mathcal{T}$ are of the form $\exists U.(\{a\} \sqcap C)$ or $\forall U.(\neg\{a\} \sqcup C)$ which are both equivalent to $@_a C$. Hence:

Corollary 2. *The logics of $\mathcal{ALCO@}[\mathcal{QIS}elf]$ are closed under substitutions.*

4 DLs Not Closed under Substitutions

In this section, we give a family of Description Logics which are strictly less expressive than their extension with substitutions. In addition to the purely technical interest of sorting logics depending on whether or not they have the property of being closed under substitutions, studying less expressive logics is meaningful as the more expressive logics are known to have a greater complexity[18]. Moreover, the translation removing substitutions is itself exponential which makes the problem clearly intractable.

We first consider the Description Logic $\mathcal{ALC}$. $\mathcal{ALC}$ does not contain $\mathcal{O}$ nor $\mathcal{U}$ or @, thus ABoxes and TBoxes need to be used. Yet, we still require that no substitution appears in them.

Theorem 2. *$\mathcal{ALC}$ is not closed under substitutions.*

More formally, to prove that the addition of substitutions to $\mathcal{ALC}$ strictly increases its expressiveness, we use a result from [6]:

Theorem 3. *For every concept C of the logic $\mathcal{ALC}$, for every two interpretations $\mathfrak{I}_1$ and $\mathfrak{I}_2$, for every bisimulation relation Z between $\mathfrak{I}_1$ and $\mathfrak{I}_2$, if an element x_1 of $\mathfrak{I}_1$ satisfies C and there is an element x_2 of $\mathfrak{I}_2$ such that $x_1 Z x_2$, then x_2 satisfies C. The same can be said for the satisfiability of $ABox$ and $TBox$ assertions.*

Definition 11 (Bisimulation). *Given a signature (**C, R, I**) and two interpretations $\mathfrak{I}_1$ and $\mathfrak{I}_2$, a non-empty binary relation $\mathcal{Z} \subseteq (\Delta_1^{\mathfrak{I}} \times \Delta_2^{\mathfrak{I}})$ is a bisimulation if it satisfies:*

($\mathcal{ALC}_1$) $d_1 Z d_2 \implies$ *for all* $A \in \boldsymbol{C}, (d_1 \in A^{\mathfrak{I}_1} \Leftrightarrow d_2 \in A^{\mathfrak{I}_2})$
($\mathcal{ALC}_2$) *For all* $R \in \boldsymbol{R}, (d_1 Z d_2 \wedge d_1 R^{\mathfrak{I}_1} e_1 \implies \exists e_2 . d_2 R^{\mathfrak{I}_2} e_2 \wedge e_1 Z e_2)$
($\mathcal{ALC}_3$) *For all* $R \in \boldsymbol{R}, (d_1 Z d_2 \wedge d_2 R^{\mathfrak{I}_2} e_2 \implies \exists e_1 . d_1 R^{\mathfrak{I}_1} e_1 \wedge e_1 Z e_2)$
($\mathcal{ALC}_4$) *For all* $a \in \boldsymbol{I}, a^{\mathfrak{I}_1} Z a^{\mathfrak{I}_2}$

The idea of the proof of Theorem 2 consists in building an interpretation $\mathfrak{I}_1$ such that an $\mathcal{ALC}$-concept C containing substitutions holds and $\mathfrak{I}_2$ bisimilar to $\mathfrak{I}_1$ where C does not hold, thus proving that $\mathcal{ALC}$ is not as expressive as $\mathcal{ALC}\sigma$. Fig. 1 depicts such interpretations.

Let's check that $\mathfrak{I}_1$ and $\mathfrak{I}_2$ are bisimilar. As the only concept in **C** is A and **R** is empty, we verify the four axioms given in Def 11:

($\mathcal{ALC}_1$) $d_1 Z d_3 \rightsquigarrow (d_1 \in A^{\mathfrak{I}_1} \Leftrightarrow d_3 \in A^{\mathfrak{I}_2})$✓	$\mathcal{ALC}_2$ $\mathbf{R} = \emptyset$ ✓.
$d_2 Z d_3 \rightsquigarrow (d_2 \in A^{\mathfrak{I}_1} \Leftrightarrow d_3 \in A^{\mathfrak{I}_2})$✓	($\mathcal{ALC}_2$ $\mathbf{R} = \emptyset$ ✓.
($\mathcal{ALC}_3$) $\mathbf{R} = \emptyset$ ✓.	($\mathcal{ALC}_4$) $a^{\mathfrak{I}_1} Z a^{\mathfrak{I}_2}$✓

So $\mathfrak{I}_1$ and $\mathfrak{I}_2$ are bisimilar.

However, after the removal of the individual a from the concept A, the concept A still holds in d_2 and does not hold in d_3. This shows that there is no equivalent concept to $A[A := A - a]$ in $\mathcal{ALC}$, which is enough to show that $\mathcal{ALC}$ is not closed under substitution.

One could wonder whether it is possible to find a way, using a TBox and an ABox to find something that would be equivalent to $A[A := A - a]$ in $\mathcal{ALC}$. As the signature is $(\{A\}, \emptyset, \{a\})$, there are only few choices available as individual assertions or concept inclusions. A rapid analyses of these choices shows that there is no way to build an ABox nor a TBox that would provide an equivalent for $A[A := A - a]$.

In [6], one may find other definitions of bisimulations for various Description Logics used to show that DL-concepts are stable. Below, we recall briefly these definitions. The names of the axioms indicate which logic they apply to. For instance, if one wants to deal with $\mathcal{ALCUI}$, one has to use axioms ($\mathcal{ALC}_1$), ($\mathcal{ALC}_2$), ($\mathcal{ALC}_3$), ($\mathcal{ALC}_4$), ($\mathcal{U}_1$), ($\mathcal{U}_2$), ($\mathcal{I}_1$) and ($\mathcal{I}_2$) to define bisimulations.

($\mathcal{U}_1$) For all $d \in \Delta^{\mathfrak{I}_1}$, there exists $d' \in \Delta^{\mathfrak{I}_2} . d Z d'$

($\mathcal{U}_2$) For all $d' \in \Delta^{\mathfrak{I}_2}$, there exists$d \in \Delta^{\mathfrak{I}_1}.dZd'$
($\mathcal{I}_1$) For all $R \in \mathbf{R}, (d_1 Z d_2 \wedge d_1 R^{-\mathfrak{I}_1} e_1 \implies$ there exists $e_2.d_2 R^{-\mathfrak{I}_2} e_2 \wedge e_1 Z e_2)$
($\mathcal{I}_2$) For all $R \in \mathbf{R}, (d_1 Z d_2 \wedge d_2 R^{-\mathfrak{I}_2} e_2 \implies$ there exists $e_1.d_1 R^{-\mathfrak{I}_1} e_1 \wedge e_1 Z e_2)$
($\mathcal{O}$) For all $a \in \mathbf{I}, d_1 Z d_2 \implies (d_1 = a^{\mathfrak{I}_1} \Leftrightarrow d_2 = a^{\mathfrak{I}_2})$
($\mathcal{Q}$) For all $R \in \mathbf{R}, (d_1 Z d_2 \implies$ there is a bijection $h : \{y | (d_1, y) \in R^{\mathfrak{I}_1}\} \rightarrow \{y' | (d_2, y') \in R^{\mathfrak{I}_2}\}$ such that $h \subseteq Z$.
($\mathcal{IQ}$) For all $R \in \mathbf{R}, (d_1 Z d_2 \implies$ there is a bijection $h : \{y | (y, d_1) \in R^{\mathfrak{I}_1}\} \rightarrow \{y' | (y', d_2) \in R^{\mathfrak{I}_2}\}$ such that $h \subseteq Z$.
($\mathcal{S}elf$) For all $R \in \mathbf{R}, d_1 Z d_2 \implies (d_1 R^{\mathfrak{I}_1} d_1 \Leftrightarrow d_2 R^{\mathfrak{I}_2} d_2)$.

One could see that there is no rule for @. Actually, it is easy to prove that ($\mathcal{ALC}_4$) is enough for @.

The same example allows to prove that the DLs given in the following corollary are not closed under substitutions. The reader may verify that the axioms of bisimulations corresponding to these logics are satisfed by Z.

Corollary 3. *The logics of $\mathcal{ALC}[\mathcal{U}@\mathcal{IQ}\mathcal{S}elf]$ are not closed under substitutions.*

If a DL contains $\mathcal{O}$, the example of Fig. 1 no longer is a counter-example because Z is not a bisimulation anymore. Indeed $d_2 Z d_3$ but $d_3 = a^{\mathfrak{I}_2}$ whereas $d_2 \neq a^{\mathfrak{I}_2}$ contradicting ($\mathcal{O}$).

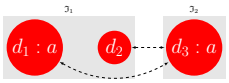

Fig. 1. On the left, $\mathfrak{I}_1$ is a model of $A[A := A - a]$. On the right, $\mathfrak{I}_2$ is not. Nodes satisfying A are drawn in red. Dashed arrows represent Z. The signature is $(\{A\}, \emptyset, \{a\})$.

Up to now, in this section, we have considered that the only part of the logic where substitutions where allowed was the concept whose satisfiability or validity we wanted to prove. Usually, the problem when DLs are considered is not the satisfiability or the validity of a concept, though, but whether or not the association of an ABox and a TBox can yield a model.

Another interesting question then is whether or not, for an ABox and a TBox both possibly containing substitutions, it is possible to find a equivalent couple ABox-TBox without substitutions. It is not. Indeed, the ABox $(a : A)[A := A + b]$ is satisfied in only two possible cases: either $a : A$ before the substitution occurs or $a = b$. None of these two cases can be expressed using TBox assertions and, using an ABox, you can only express them as a Boolean ABox[19], which is not usually allowed, or using a nominal.

5 The case of $\mathcal{ALCO}$

The technique we used in the previous section cannot be applied directly in the case of the logic $\mathcal{ALCO}$. Indeed, we can show the following result.

Theorem 4. *$\mathcal{ALCO}\sigma$ concepts are stable under $\mathcal{ALCO}$-bisimulations.*

To show this theorem, we assume given two interpretations $\mathfrak{I}_1 = \{\Delta^{\mathfrak{I}_1}, .^{\mathfrak{I}_1}\}$ and $\mathfrak{I}_2 = \{\Delta^{\mathfrak{I}_2}, .^{\mathfrak{I}_2}\}$, $x_1 \in \Delta^{\mathfrak{I}_1}$ and $x_2 \in \Delta^{\mathfrak{I}_2}$, Z an $\mathcal{ALCO}$-bisimulation between $\mathfrak{I}_1$ and $\mathfrak{I}_2$ such that $x_1\ Z\ x_2$ and prove for any $\mathcal{ALCO}\sigma$-concept C, $x_1 \in C^{\mathfrak{I}_1} \Leftrightarrow x_2 \in C^{\mathfrak{I}_2}$.

As the complete proof is quite long, only a summary will be reported here.

We will mostly reuse the rewriting system intropduced in Sect. 3. Given an $\mathcal{ALCO}\sigma$-concept, the rules for atomic concepts give an equivalent $\mathcal{ALCO}$-concept which is invariant for $\mathcal{ALCO}$-bisimulations.

For the other rules, we will reason by induction. We assume that the concept whose invariance we want to show is $(C \sqcup D)\theta$. The correct translation is $C\theta \sqcup D\theta$. Let's prove that $x_1 \in (C\theta \sqcup D\theta)^{\mathfrak{I}_1} \Rightarrow x_2 \in (C\theta \sqcup D\theta)^{\mathfrak{I}_2}$. We know that $x_1 \in (C\theta \sqcup D\theta)^{\mathfrak{I}_1}$. That is, $x_1 \in (C\theta)^{\mathfrak{I}_1}$ or $x_1 \in (D\theta)^{\mathfrak{I}_1}$. Then, by induction, $x_2 \in (C\theta)^{\mathfrak{I}_2}$ or $x_2 \in (D\theta)^{\mathfrak{I}_2}$. Thus $x_2 \in (C\theta \sqcup D\theta)^{\mathfrak{I}_2}$. The same way, we can show that $x_2 \in (C\theta \sqcup D\theta)^{\mathfrak{I}_2} \Rightarrow x_1 \in (C\theta \sqcup D\theta)^{\mathfrak{I}_1}$. Showing $(C \sqcap D)\theta$ and $(\neg D)\theta$ are invariant is done mutatis mutandis.

The concepts using the constructors $\exists$ and $\forall$ are handled by going back to the definition of the interpretations and bisimulations. Let's prove that $(\exists R.C)[R := R{+}(a,b)]$ is invariant for $\mathcal{ALCO}$-bisimulations. Once again, we start assuming $x_1 \in ((\exists R.C)[R := R{+}(a,b)])^{\mathfrak{I}_1}$. That means $x_1 \in \{y \mid \exists z \text{ such that } (y,z) \in (R[R := R{+}(a,b)])^{\mathfrak{I}_1} \wedge z \in (C[R := R + (a,b)])^{\mathfrak{I}_1}\}$. Then, by replacing $(R[R := R + (a,b)])^{\mathfrak{I}_1}$ by its expression, one gets $x_1 \in \{y \mid \exists z \text{ such that } (y,z) \in (R \cup \{(a,b)\})^{\mathfrak{I}_1} \wedge z \in (C[R := R + (a,b)])^{\mathfrak{I}_1}\}$. Thus, $x_1 \in \{y \mid \exists z \text{ such that } (y,z) \in R^{\mathfrak{I}_1} \wedge z \in (C[R := R + (a,b)])^{\mathfrak{I}_1}\} \cup \{a^{\mathfrak{I}_1}, b^{\mathfrak{I}_1} \in (C[R := R{+}(a,b)])^{\mathfrak{I}_1}\}$. As x_1 belongs to at least one of those sets: either there exists $z_1 \in \Delta_1$ such that $(x_1, z_1) \in R^{\mathfrak{I}_1} \wedge z_1 \in (C[R := R{+}(a,b)])^{\mathfrak{I}_1}$ which means that, by ($\mathcal{ALC}_3$), there exists $z_2 \in \Delta_2$ such that $(x_2, z_2) \in R^{\mathfrak{I}_2}$ and, by induction, $z_2 \in (C[R := R + (a,b)])^{\mathfrak{I}_2}$. Thus $x_2 \in ((\exists R.C)[R := R + (a,b)])^{\mathfrak{I}_2}$,
or $x_1 = a^{\mathfrak{I}_1}$ and $b^{\mathfrak{I}_1} \in (C[R := R{+}(a,b)])^{\mathfrak{I}_1}$ which means that, by ($\mathcal{O}$), $x_2 = a^{\mathfrak{I}_2}$ and, by induction, $b^{\mathfrak{I}_2} \in (C[R := R + (a,b)])^{\mathfrak{I}_2}$. Thus $x_2 \in ((\exists R.C)[R := R + (a,b)])^{\mathfrak{I}_2}$.

The exact same reasoning can be done in the other direction to prove that this is an equivalence.

Theorem 4 could lead us to think that the addition of substitutions to $\mathcal{ALCO}$ does not increase the expressiveness of the logic. This is not true. We propose below another kind of binary relations that are able to distinguish between $\mathcal{ALCO}$ and $\mathcal{ALCO}\sigma$. The only difference is the last axiom ($r\mathcal{ALC}_4$). To put it simply, it checks that the interpretation agree on the connected subgraph containing the root instead of agreeing on all connected subgraphs containing at least one nominal.

Definition 12. *Given two interpretations $\mathfrak{I}_1$ and $\mathfrak{I}_2$, a* rooted-bisimulation *between $\mathfrak{I}_1$ and $\mathfrak{I}_2$ rooted in (x,x′) is a binary relation $\mathcal{Z} \subseteq (\Delta^{\mathfrak{I}_1} \times \Delta^{\mathfrak{I}_2})$ such that:*

($\mathcal{ALC}_1$) $d_1 Z d_2 \implies$ *for all* $A \in \boldsymbol{C}, (d_1 \in A^{\mathfrak{I}_1} \Leftrightarrow d_2 \in A^{\mathfrak{I}_2})$
($\mathcal{ALC}_2$) *For all* $R \in \boldsymbol{R}, (d_1 Z d_2 \wedge d_1 R^{\mathfrak{I}_1} e_1 \implies \exists e_2 . d_2 R^{\mathfrak{I}_2} e_2 \wedge e_1 Z e_2)$
($\mathcal{ALC}_3$) *For all* $R \in \boldsymbol{R}, (d_1 Z d_2 \wedge d_2 R^{\mathfrak{I}_2} e_2 \implies \exists e_1 . d_1 R^{\mathfrak{I}_1} e_1 \wedge e_1 Z e_2)$
($r\mathcal{ALC}_4$) $x Z x'$

This definition is extended for logics containing $\mathcal{ALC}$ the same way as the definition of bisimulations.

Definition 13. *A concept C is* stable under rooted-bisimulations *if, for any interpretations $\mathfrak{I}_1$, $\mathfrak{I}_2$, any rooted-bisimulation Z between $\mathfrak{I}_1$, $\mathfrak{I}_2$ rooted in (x,x') and any y reachable from x, there exists y' such that yZy' and $y \in C^{\mathfrak{I}_1} \Leftrightarrow y' \in C^{\mathfrak{I}_2}$.*

Theorem 5. *$\mathcal{ALCO}$-concepts are stable under rooted-bisimulations*

Once again, the proof is done by induction on the length of the concepts. It is obvious for concept names due to rule ($\mathcal{ALC}$) and for nominals due to rule ($\mathcal{O}$).

Theorem 6. *$\mathcal{ALCO}\sigma$-concepts are not stable under rooted-bisimulations.*

Fig. 2 provides a counter-example illustrating Theorem 6. The results of Theorems 5 and 6 can be extended to the logics in $\mathcal{ALCO}[\mathcal{IQS}elf]$.

Let's check that $\mathfrak{I}_1$ and $\mathfrak{I}_2$ are (d_1,d_3)-bisimilar. As the only concept in $\mathbf{C}$ is A and there is no role in $\mathbf{R}$, we verify the four axioms given in Def 12:

($\mathcal{ALC}_1$) $d_1 Z d_3 \rightsquigarrow (d_1 \in A^{\mathfrak{I}_1} \Leftrightarrow d_3 \in A^{\mathfrak{I}_2})$✓
($\mathcal{ALC}_3$) $\mathbf{R} = \emptyset$ ✓.
($\mathcal{O}$) $d_1 Z d_3 \rightsquigarrow d_1 = a^{\mathfrak{I}_1} \Leftrightarrow d_3 = a^{\mathfrak{I}_2}$ ✓

($\mathcal{ALC}_2$)] $\mathbf{R} = \emptyset$ ✓.
($r\mathcal{ALC}_4$) $d_1^{\mathfrak{I}_1} Z d_3^{\mathfrak{I}_2}$ ✓

So $\mathfrak{I}_1$ and $\mathfrak{I}_2$ are (d_1,d_3)-bisimilar.

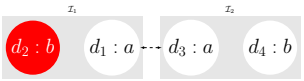

Fig. 2. On the left, $\mathcal{I}_1$ is a model of $(\forall R.A)[R := R + (a, b)]$. On the right, $\mathcal{I}_2$ is not. Nodes satisfying A are drawn in red. Dashed arrows represent Z. The signature is $\{\{A\}, \emptyset, \{a, b\}\}$. $\mathcal{I}_1$ and $\mathcal{I}_2$ are (d_1, d_3)-rooted-bisimilar.

Theorem 7. *$\mathcal{ALCO}$ is not closed under substitutions.*

Even in the presence of ABoxes and TBoxes, $\mathcal{ALCO}$ is still not closed under substitutions. Indeed, no ABox or TBox allows to express the concept $(\{a\} \wedge \exists R.A)[R := R + (a, b)]$ as it can only express one of the two possible "templates" of models at a time: the concept $\{a\} \wedge \exists R.A$ is satisfied or the concept $\{a\}$ together with the ABox $b : A$ is satisfied. It is possible to express any of the two alone, the intersection of the two but not their union, which is the one we are looking for.

Once again, one may want to look at the problem of whether or not there exist an equivalent couple ABox-TBox without substitutions for any ABox-Tbox with substitutions. Once again, it is not possible. The ABox assertion $a : (\exists R.A)[R := R + (a, b)]$ yields the same possible models given previously. Once more the solution would be to use a Boolean ABox [19] to be closed under substitutions.

6 Related Work and Conclusion

We tackled the problem of adding substitutions to concepts in various DLs. We have described different situations where substitutions increase the expressive power of DLs and where they do not. In the latter situations, proving concept satisfiability in presence of substitutions can be carried out using the same proof procedures tailored for DLs without substitutions. However, when the expressive power of the logics are altered by substitutions, new proof procedures are to be devised. We have shown that the most expressive logics we considered are closed under substitutions. Moreover, as they allow to define the Abox and the Tbox directly as a part of the concept, it is possible to represent this way knowledge representation bases. On the other hand, logics without nominals, are strictly extended with the introduction of substitutions. Adding Abox or Tbox assertions is not enough to cover the gap in expressiveness either. Finally, logics with nominals are a borderline case where there is no equivalent formula without substitutions allowing to express the formula with substitutions but it is possible to find a disjunction of such formulae. A solution, that we didn't consider here, for these logics would be to allow role constructors that would be the union and difference of roles.

This paper presents a systematic study of the closure under substitutions of various families of DLs. We are not aware of similar work in the literature.

This paper only concentrates on DLs. There are other approaches to deal with graph or pointer transforming programs, such as separation logic [16], which are to a large extent orthogonal to our approach (also see [5]), or Pale [13] based on monadic second-order logic, requiring a tree-shaped backbone to be identified in graphs.

DLs have been investigated in various areas with dynamic structures. For instance, Dynamic Description Logics [21] combine description logics with a PDL-style action language, with unspecified elementary actions, whereas in our case, we concentrate on a specific set of primitives .

Knowledge update is a field that has been widely studied be it as database update [20,7] or in the more philosophical epistemic logic[2,11].

In [12], the authors study the effects of elementary updates on sets of ABox formulae. As updates and substitutions look differently at the actions, their results and ours differ even though the topics are related.

Only few approaches use action languages with updates in the form of substitutions. [4] use a full programming language (including loops), where updates are restricted to elementary positive or negative concept and role expressions.

[1] uses a simpler update language (not containing loops), but updates may contain complex concept and role expressions. This provides a logic that could be used for verification of graph transformations since the logic is closed under substitutions.

Last but not least, Hoare style program verification for graph transformation has already been proposed in the literature (see, e.g., [15,8]). The investigated logics are often dialects of nested graph condition [9] which are more expressive than the DLs considered in the present paper. Our aim is not to achieve high expressivity, but investigate which reasoning principles can be implemented soundly with less complex logics.

References

1. Shqiponja Ahmetaj, Diego Calvanese, Magdalena Ortiz, and Mantas Simkus. Managing change in graph-structured data using description logics. In *Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI 2014)*, pages 966–973. AAAI Press, 2014.
2. Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *J. Symb. Log.*, 50(2):510–530, 1985.
3. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
4. Jon Haël Brenas, Rachid Echahed, and Martin Strecker. A Hoare-like calculus using the $SROIQ^{\sigma}$ logic on transformations of graphs. In Josep Diaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science*, volume 8705 of *Lecture Notes in Computer Science*, pages 164–178. Springer Berlin Heidelberg, 2014.
5. Stéphane Demri and Morgan Deters. Separation logics and modalities: A survey. *Journal of Applied Non-Classical Logics*, 25(1):50–99, 2015.
6. Ali Rezaei Divroodi and Linh Anh Nguyen. On bisimulations for description logics. *Information Sciences*, 295:465 – 493, 2015.
7. Ronald Fagin, Gabriel M. Kuper, Jeffrey D. Ullman, and Moshe Y. Vardi. Updating logical databases. *Advances in Computing Research*, 3:1–18, 1986.
8. Annegret Habel and Karl-Heinz Pennemann. Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science*, 19(02):245–296, 2009.
9. Annegret Habel, Karl-Heinz Pennemann, and Arend Rensink. Weakest preconditions for high-level programs. In *Procs. of 3rd International Conference on Graph Transformations, ICGT 2006*, volume LNCS 4178, pages 445–460. Springer, 2006.
10. C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.
11. Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings KR'91. Cambridge, MA, USA, April 22-25, 1991.*, pages 387–394, 1991.
12. Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. Foundations of instance level updates in expressive description logics. *Artificial Intelligence*, 175(18):2170–2197, 2011.
13. Anders Møller and Michael I. Schwartzbach. The pointer assertion logic engine. In *PLDI*, pages 221–231, 2001.
14. OWL Standard. OWL Standard. `http://www.w3.org/2001/sw/wiki/OWL`.
15. Christopher M. Poskitt and Detlef Plump. Hoare-style verification of graph programs. *Fundamenta Informaticae*, 118(1-2):135–175, 2012.
16. John C. Reynolds. Separation logic: A logic for shared mutable data structures. *Logic in Computer Science, Symposium on*, 0:55, 2002.
17. SPARQL Standard. SPARQL Standard. `http://www.w3.org/2001/sw/wiki/SPARQL`.
18. Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artif. Intell. Res. (JAIR)*, 12:199–217, 2000.
19. Jigneshkumar R. Viradia. Reasoning with boolean aboxes. Master's thesis, Technisch Universität Dresden, 2008.
20. Marianne Winslett. *Updating logical databases*, volume 9. Cambridge University Press, 2005.
21. Frank Wolter and Michael Zakharyaschev. Dynamic description logics. *Advances in Modal Logic*, 2:431–446, 1998.