TECHNISCHE
UNIVERSITÄT
DRESDEN

Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Integrating Description Logics and Action Formalisms for Reasoning about Web Services

Franz Baader, Carsten Lutz, Maja Miličić, Ulrike Sattler, Frank Wolter

LTCS-Report 05-02

Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
http://lat.inf.tu-dresden.de

Hans-Grundig-Str. 25
01062 Dresden
Germany

Integrating Description Logics and Action Formalisms for Reasoning about Web Services

Franz Baader[1], Carsten Lutz[1], Maja Miličić[1], Ulrike Sattler[2], and Frank Wolter[3]

[1] Inst. für Theoretische Informatik
TU Dresden
Germany
lastname@tcs.inf.tu-dresden.de

[2] Dept. of Computer Science
Univ. of Manchester
UK
sattler@man.cs.ac.uk

[3] Dept. of Computer Science
Univ. of Liverpool
UK
frank@csc.liv.ac.uk

Abstract

Motivated by the need for semantically well-founded and algorithmically managable formalisms for describing the functionality of Web services, we introduce an action formalism that is based on description logics (DLs), but is also firmly grounded on research in the reasoning about action community. Our main contribution is an analysis of how the choice of the DL influences the complexity of standard reasoning tasks such as projection and executability, which are important for Web service discovery and composition.

Contents

1 Introduction

Description logics [2] play an important rôle in the Semantic Web since they are the basis of the W3C-recommended Web ontology language OWL [3, 12], which can be used to create semantic annotations describing the content of Web pages [33]. In addition to static information, the Web also offers services, which allow their users to effect changes in the world, like buying a book or opening a bank account. As in the case of static information, annotations describing the semantics of the service should facilitate discovery of the right service for a given task. Since services create changes of the world, a faithful representation of its functionality should deal with this dynamic aspect in an appropriate way.

The OWL-S initiative [32] uses OWL to develop an ontology of services, covering different aspects of Web services, among them functionality. To describe their functionality, services are viewed as processes that (among other things) have pre-conditions and effects. A similar approach is taken by the Web service modelling ontology WSMO [15]. However, the faithful representation of the dynamic behaviour of processes (what changes of the world they cause) is beyond the scope of a static ontology language like OWL.

In AI, the notion of an action is used both in the planning and the reasoning about action communities to denote an entity whose execution (by some agent) causes changes of the world (see e.g. [28, 34]). Thus, it is not surprising that theories developed in these comunities have been applied in the context of Semantic Web services. For example, [20, 21] use the situation calculus [28] and GOLOG [17] to formalize the dynamic aspects of Web services and to describe their composition. In [31], OWL-S process models are translated into the planning language of the HTN planning system SHOP2 [22], which is then used for automatic Web service composition.

The approach used in this paper is in a similar vein. We are interested in the faithful description of the changes to the world induced by the invocation of a service. To this purpose, we describe services as actions that have pre-conditions and post-conditions (its effects). These conditions are expressed with the help of description logic assertions, and the current state of the world is (incompletely) described using a set of such assertions (a so-called ABox). In addition to atomic services, we also consider simple composite services, which are sequences of atomic services. The semantics of a service is defined using the possible models approach developed in the reasoning about action community [38, 39, 40, 6, 9, 5], and is fully compatible with the usual DL semantics.. However, we will also show that this semantics can be viewed as an instance of Reiter's approach [27, 25, 16, 28] for taming the situation calculus. In particular, our semantics solves the frame problem in precisely the same way.

Then, we concentrate on two basic reasoning problems for (possibly composite) services: executability and projection. Executability checks whether, given our current and possibly incomplete knowledge of the world, we can be sure that the service is executable, i.e., all pre-conditions are satisfied. Projection checks whether a certain condition always holds after the successful execution of the service, given our knowledge of the current state of the world. Both tasks are relevant for service discovery. It is obviously preferable to choose a service that is guaranteed to be executable in the

current (maybe incompletely known) situation. In addition, we execute the service to reach some goal, and we only want to use services that achieve this goal. Though these reasoning tasks may not solve the discovery problem completely, they appear to be indispensable subtasks.

The main contribution of this paper is an analysis of how the choice of the DL influences the complexity of these two reasoning tasks for services. For the DLs $\mathcal{L}$ considered here, which are all sublanguages of the DL $\mathcal{ALCQIO}$, the complexity of executability and projection for services expressed in this DL coincides with the complexity of standard DL reasoning in $\mathcal{L}$ extended with so-called nominals (i.e., singleton concepts). The reason is that we can reduce both tasks for services to the standard DL task of checking consistency of an ABox w.r.t. an acyclic TBox, provided that we can use nominals within concept descriptions. This reduction is optimal since our hardness results show that the complexity increase (sometimes) caused by the addition of nominal cannot be avoided. We also motivate the restrictions we impose: we discuss the semantic and the computational problems that arise when these restrictions are loosened. Most importantly, we prove that allowing for complex concepts in post-conditions not only yields semantic problems, but also the undecidability of the two service reasoning problems.

2 The Formalism

The framework for reasoning about Web services proposed in this paper is not restricted to a particular description logic, but can be instantiated with any description logic that seems appropriate for the application domain at hand. For our complexity results, we consider the DL $\mathcal{ALCQIO}$ and a number of its sublanguages. The reason for choosing $\mathcal{ALCQIO}$ is that it forms the core of OWL-DL, the description logic variant of OWL. The additional OWL-DL constructors could be easily added, with the exception of transitive roles which are discussed in Section 4. In this section, we first introduce $\mathcal{ALCQIO}$ and several of its fragments, then define the framework for representing Web services, propose relevant reasoning tasks for services, and finally discuss the relation between our formalism and the situation calculus.

2.1 The Description Logic $\mathcal{ALCQIO}$

In DL, concepts are inductively defined with the help of a set of constructors which determine the expressive power of the DL. We introduce the constructors available in $\mathcal{ALCQIO}$ and explain how its fragments discussed here can be obtained by omitting constructors. The syntax of $\mathcal{ALCQIO}$ as presented here can be easily translated into an XML syntax as used for OWL-DL [12].

Definition 1 ($\mathcal{ALCQIO}$ Syntax). Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be disjoint and countably infinite sets of *concept names*, *role names*, and *individual names*. A *role* is either a role name or the inverse r^- of a role name r. The set of $\mathcal{ALCQIO}$-concepts is the smallest set satisfying the following properties:

- each concept name $A \in \mathsf{N_C}$ is a concept;

Symbol	Constructor	$\mathcal{ALC}$	$\mathcal{ALCO}$	$\mathcal{ALCQ}$	$\mathcal{ALCI}$	$\mathcal{ALCQO}$	$\mathcal{ALCIO}$	$\mathcal{ALCQI}$
$\mathcal{Q}$	$(\leqslant n\ r\ C)$ $(\geqslant n\ r\ C)$			x		x		x
$\mathcal{I}$	r^-				x		x	x
$\mathcal{O}$	$\{a\}$		x			x	x	

Figure 1: Fragments of $\mathcal{ALCQIO}$.

- if C and D are concepts, r is a role, a an individual name, and n a natural number, then the following are also concepts:

$$\begin{aligned} \neg C \quad & (\textit{negation}) \\ C \sqcap D \quad & (\textit{conjunction}) \\ C \sqcup D \quad & (\textit{disjunction}) \\ \{a\} \quad & (\textit{nominal}) \\ (\geqslant n\ r\ C) \quad & (\textit{atmost number restriction}) \\ (\leqslant n\ r\ C) \quad & (\textit{atleast number restriction}) \end{aligned}$$

△

It is convenient to introduce some abbreviations. As usual, we use the Boolean standard abbreviations $\rightarrow$ and $\leftrightarrow$. Additionally, we use

$\exists R.C$	for	$(\geqslant 1\ R\ C)$	(*existential restriction*)
$\forall R.C$	for	$(\leqslant 0\ R\ \neg C)$	(*universal restriction*)
$\top$	for	a propositional tautology	(*top*)
$\bot$	for	$\neg\top$	(*bottom*)

The DL that allows only for negation, conjunction, disjunction, and universal and existential restrictions is called $\mathcal{ALC}$. The availability of additional constructors is indicated by concatenation of a corresponding letter: $\mathcal{Q}$ stands for number restrictions; $\mathcal{I}$ stands for inverse roles, and $\mathcal{O}$ for nominals. This explains the name $\mathcal{ALCQIO}$ for our DL, and also allows us to refer to sublanguages as indicated in Figure 1.

We now define the semantics of $\mathcal{ALCQIO}$ concepts.

Definition 2 ($\mathcal{ALCQIO}$ Semantics). An interpretation $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a mapping that assigns

- to each concept name A, a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
- to each individual name a, an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$;
- to each role name r, a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

The interpretation of inverse roles and complex concepts is then defined as follows, with

$\#S$ denoting the cardinality of the set S:

$$\begin{aligned}
(r^-)^{\mathcal{I}} &= \{(e,d) \mid (d,e) \in r^{\mathcal{I}}\} \\
(\{a\})^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\leqslant n\ r\ C)^{\mathcal{I}} &= \{d \mid \#\{e \in C^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \leq n\} \\
(\geqslant n\ r\ C)^{\mathcal{I}} &= \{d \mid \#\{e \in C^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \geq n\}
\end{aligned}$$

An interpretation $\mathcal{I}$ is called a *model* of a concept C if $C^{\mathcal{I}} \neq \emptyset$ △

A central idea of the Semantic Web is to represent relevant terminological knowledge in *ontologies*: ontologies formally define notions that are relevant for the description of Web page content, and, conversely, such descriptions can refer to ontologies to ensure a well-understood and unambiguous meaning of the used notions. The presence of ontologies in the Semantic Web architecture provides one of the main motivations for using a DL approach for reasoning about services: a large collection of ontologies are *already available* on the Semantic Web, and many of them are formulated in the description logic OWL-DL. By allowing the description of Web services to refer to (DL-based) ontologies, we thus enable the reuse of existing ontologies for describing services, and ensure that the description of static Web pages and of services are based on the same terminology. In the DL world, ontologies are usually called TBoxes.

Definition 3 (TBox, Acyclic TBox). A *concept definition* is of the form $A \equiv C$, where A is a concept name and C a concept. A *TBox* is a finite set of concept definitions with unique left-hand sides. We say that a concept name A *directly uses* a concept name B w.r.t. $\mathcal{T}$ if there is a concept definition $A \equiv C \in \mathcal{T}$ with B occurring in C. Let *uses* be the transitive closure of directly uses. Then a TBox $\mathcal{T}$ is *acyclic* if no concept name uses itself w.r.t. $\mathcal{T}$.

An interpretation $\mathcal{I}$ *satisfies* a concept definition $A \equiv C$ (written $\mathcal{I} \models A \equiv C$) if $A^{\mathcal{I}} = C^{\mathcal{I}}$. $\mathcal{I}$ is called a *model* of a TBox $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies all concept definitions in $\mathcal{T}$. △

We call a concept name A *defined w.r.t.* a TBox $\mathcal{T}$ if A occurs on the left-hand side of a concept definition in $\mathcal{T}$, and *primitive w.r.t.* $\mathcal{T}$ otherwise. Throughout this paper, we will usually restrict ourselves to acyclic TBoxes. The reason for this restriction is that cyclic TBoxes cause semantic problems as discussed in Section 4.

To predict the outcome of applying a service, an agent usually needs to take into account her knowledge about the current state of the world. In our framework, a complete description of the state of the world corresponds to an interpretation. However, a client usually has only incomplete knowledge. In DLs, such incomplete knowledge about the world is represented in an ABox.

Definition 4 (ABox). An *assertion* is of the form $C(a)$, $r(a,b)$ or $\neg r(a,b)$, where $a, b \in \mathsf{N_I}$, C is a concept, and r a role. An *ABox* is a finite set of assertions. An

interpretation $\mathcal{I}$ *satisfies* an assertion

$$\begin{array}{lll} C(a) & \text{iff} & a^{\mathcal{I}} \in C^{\mathcal{I}}; \\ r(a,b) & \text{iff} & (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}; \\ \neg r(a,b) & \text{iff} & (a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}. \end{array}$$

If φ is an assertion, then we write $\mathcal{I} \models \varphi$ iff $\mathcal{I}$ satisfies φ. An interpretation $\mathcal{I}$ is called a *model* of an ABox $\mathcal{A}$, written $\mathcal{I} \models \mathcal{A}$, if $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$ $\triangle$

To improve readability, we will sometimes write the assertion $C(a)$ in the form $a : C$. Negated role assertions are usually not considered in DL, but they are very useful as pre- and post-conditions. As described below, reasoning with such assertions can easily be reduced to reasoning without them if the DL under consideration allows for value restriction and atomic negation.

Various reasoning problems are considered for DLs. For the purpose of this paper, it is sufficient to introduce only three of them: concept satisfiability, ABox consistency, and ABox consequence.

Definition 5 (DL Reasoning Problems). Let C be a concept, $\mathcal{A}$ an ABox, and $\mathcal{T}$ a TBox. Then

- C is *satisfiable* w.r.t. the TBox $\mathcal{T}$ iff there exists an interpretation $\mathcal{I}$ that is a model of both C and $\mathcal{A}$;
- $\mathcal{A}$ is *consistent* w.r.t. the TBox $\mathcal{T}$ iff there exists an interpretation $\mathcal{I}$ that is a model of both $\mathcal{T}$ and $\mathcal{A}$;
- an ABox assertion φ is a *consequence* of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ (written $\mathcal{A}, \mathcal{T} \models \varphi$) if every model of $\mathcal{A}$ and $\mathcal{T}$ satisfies φ.

$\triangle$

ABox consequence will play the most important role in this paper. As it is a slightly unusual DL reasoning problem, we briefly show that ABox consequence with negated role assertions, i.e. assertions of the form $\neg r(a,b)$, can be polynomially reduced to ABox consistency without negated role assertions, and vice versa. From these reductions, it follows that ABox consistency and ABox consequence are of the same complexity. We proceed in two steps:

Firstly, we reduce ABox consequence with negated role assertions to ABox consistency with negated role assertions, and vice versa. For an assertion φ, let $\neg\varphi = \neg C(a)$ if $\varphi = C(a)$, $\neg\varphi = \neg r(a,b)$ if $\varphi = r(a,b)$, and $\neg\varphi = r(a,b)$ if $\varphi = \neg r(a,b)$. Then φ is a consequence of $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A} \cup \{\neg\varphi\}$ is inconsistent w.r.t. $\mathcal{T}$. Conversely, an ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ iff $\bot$ is not a consequence of $\mathcal{A}$ w.r.t. $\mathcal{T}$.

Second, we reduce ABox consistency with negated role assertions to ABox consistency without such assertions. Given an ABox $\mathcal{A}$, introduce a concept name X_a not used in $\mathcal{A}$ for each individual name a used in $\mathcal{A}$. Then replace each assertion $\neg r(a,b)$ with the two assertions $(\forall r.\neg X_b)(a)$ and $X_b(b)$. Clearly, the resulting ABox $\mathcal{A}'$ is consistent iff the original one is, and $\mathcal{A}'$ is of size linear in the size of $\mathcal{A}$.

2.2 Service Descriptions

We now introduce the formalism for the representation of and reasoning about Web services. For simplicity, we concentrate on *ground services*, i.e., services where the input parameters have already been instantiated by individual names. *Parametric* services, which contain variables in place of individual names, should be viewed as a compact representation of all its ground instances: a parametric service simply represents the set of all ground services obtained from the parametric service by replacing variables with individual names. The handling of such parametric services takes place "outside" of our formalism and is not discussed in detail in the current paper. We may savely restrict ourselves to ground services since all the reasoning tasks considered in this paper presuppose that parametric services have already been instantiated. For other tasks, such as planning, it may be more natural to work directly with parametric services.

Definition 6 (Service). Let $\mathcal{T}$ be an acyclic TBox. An *atomic service* $S = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ for an acyclic TBox $\mathcal{T}$ consists of

- a finite set pre of ABox assertions, the *pre-conditions*;
- a finite set occ of *occlusions* of the form $A(a)$ or $r(a, b)$, with A a primitive concept name w.r.t. $\mathcal{T}$, r a role name, and $a, b \in \mathsf{N_I}$;
- a finite set post of *conditional post-conditions* of the form φ/ψ, where φ is an ABox assertion and ψ is a *primitive literal for* $\mathcal{T}$, i.e., an ABox assertion $A(a)$, $\neg A(a)$, $s(a, b)$, or $\neg s(a, b)$ with A a primitive concept name in $\mathcal{T}$ and s a role name.

A *composite service* for $\mathcal{T}$ is a finite sequence $S_1, \ldots, S_k$ of atomic services for $\mathcal{T}$. A *service* is a composite or an atomic service. △

Intuitively, the pre-conditions specify under which conditions the service is applicable. The conditional post-conditions φ/ψ say that, if φ is true before executing the service, then ψ should be true afterwards. If φ is tautological, e.g. $\top(a)$ for some individual name a, then we write just ψ instead of φ/ψ. Also note that it is not a restriction to admit only role *names* in post-conditions $s(a, b)$ and $\neg s(a, b)$ since $s^-(a, b)$ can be replaced by $s(b, a)$ and similarly for $\neg s^-(a, b)$. By the law of inertia, only those facts that are forced to change by the post-conditions should be changed by applying the service. However, it is well-known in the reasoning about action community that enforcing this minimization of change strictly is sometimes too restrictive [18, 29]. The rôle of occlusions is to describe those primitive literals to which the minimization condition does not apply.

To illustrate the definition of services, consider a Web site offering services for people that move from Continental Europe to the United Kingdom. Among its services are getting a contract with an electricity provider, opening a bank account, and applying for child benefit. Obtaining an electricity contract b for customer a does not require any pre-conditions. It is described by the service S_1, which has an empty set of pre-conditions, an empty set of occlusions, and whose post-conditions are defined as follows:

$$\mathsf{post}_1 = \{\texttt{holds}(a, b), \texttt{electricity\_contract}(b)\}.$$

Suppose the pre-condition of opening a bank account is that the customer c is eligible for a bank account in the UK and holds a proof of address. Moreover, suppose that, if a letter from the employer is available, then the bank account comes with a credit card, otherwise not. This service can be formalised by the service description S_2, which has an empty set of occlusions and the following pre- and post-conditions:

$$\begin{aligned} \mathsf{pre}_2 = \ & \{\texttt{Eligible\_bank}(a), \exists \texttt{holds.Proof\_address}(a)\} \\ \mathsf{post}_2 = \ & \{\texttt{holds}(a, c), \\ & \exists \texttt{holds.Letter}(a)/\texttt{B\_acc\_credit}(c), \\ & \neg\exists \texttt{holds.letter}(a)/\texttt{B\_acc\_no\_credit}(c)\} \end{aligned}$$

Suppose that one can apply for child benefit in the UK if one has a child and a bank account. The service S_3 that offers this application then has the following pre- and post-conditions, and again an empty set of occlusions:

$$\begin{aligned} \mathsf{pre}_3 = \ & \{\texttt{parent\_of}(a, d), \exists \texttt{holds.B\_acc}(a)\} \\ \mathsf{post}_3 = \ & \{\texttt{receives\_c\_benef\_for}(a, d)\} \end{aligned}$$

The meaning of the concepts used in S_1 and S_2 are defined in the following acyclic TBox $\mathcal{T}$:

$$\begin{aligned} \mathcal{T} = \{\texttt{Eligible\_bank} \ &\equiv \ \exists \texttt{permanent\_resident.}\{\texttt{UK}\}, \\ \texttt{Proof\_address} \ &\equiv \ \texttt{Electricity\_contract}, \\ \texttt{B\_acc} \ &\equiv \ \texttt{B\_acc\_credit} \sqcup \texttt{B\_acc\_no\_credit}\} \end{aligned}$$

To define the semantics of services, we must first define how the application of an atomic service changes the world, i.e., how it transforms a given interpretation $\mathcal{I}$ into a new one $\mathcal{I}'$. Our definition follows the possible models approach (PMA) initially proposed in [38] and further elaborated e.g. in [39, 40, 6, 9, 5]. Equivalently, we could have translated description logic into first-order logic and then define executability and projection within Reiter's framework for reasoning about deterministic actions [28]. We discuss this approach in Section 2.4. The idea underlying PMA is that the interpretation of atomic concepts and roles should change as little as possible while still making the post-conditions true. Since the interpretation of defined concepts is uniquely determined by the interpretation of primitive concepts and role names, it is sufficient to impose this minimization of change condition on primitive concepts and roles names. We assume that neither the interpretation domain nor the interpretation of individual names is changed by the application of a service.

Formally, we define a precedence relation $\preccurlyeq_{\mathcal{I},S,\mathcal{T}}$ on interpretations, which characterizes their "proximity" to a given interpretation $\mathcal{I}$. We use $M_1 \triangledown M_2$ to denote the symmetric difference between the sets M_1 and M_2.

Definition 7 (Preferred Interpretations). Let $\mathcal{T}$ be an acyclic TBox, $S = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ a service for $\mathcal{T}$, and $\mathcal{I}$ a model of $\mathcal{T}$. We define the binary relation $\preccurlyeq_{\mathcal{I},S,\mathcal{T}}$ on models of $\mathcal{T}$ by setting $\mathcal{I}' \preccurlyeq_{\mathcal{I},S,\mathcal{T}} \mathcal{I}''$ iff

- $((A^{\mathcal{I}} \triangledown A^{\mathcal{I}'}) \setminus \{a^{\mathcal{I}} \mid A(a) \in \mathsf{occ}\}) \subseteq A^{\mathcal{I}} \triangledown A^{\mathcal{I}''}$;

- $((s^{\mathcal{I}} \nabla s^{\mathcal{I}'}) \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid s(a,b) \in \mathsf{occ}\}) \subseteq s^{\mathcal{I}} \nabla s^{\mathcal{I}''}$.

for all primitive concepts A, all role names s, and all domain elements $d, e \in \Delta^{\mathcal{I}}$. When $\mathcal{T}$ is empty, we write $\preceq_{\mathcal{I},S}$ instead of $\preceq_{\mathcal{I},S,\emptyset}$. $\triangle$

Intuitively, applying the service S transforms the interpretation $\mathcal{I}$ into the interpretation $\mathcal{I}'$ if $\mathcal{I}'$ satisfies the post-conditions and is closest to $\mathcal{I}$ (as expressed by $\preccurlyeq_{\mathcal{I},S,\mathcal{T}}$) among all interpretations satisfying the post-conditions. Since we consider *conditional* post-conditions, defining when they are satisfied actually involves both $\mathcal{I}$ and $\mathcal{I}'$. We say that the pair of interpretations $\mathcal{I}, \mathcal{I}'$ *satisfies the set of post-conditions* post ($\mathcal{I}, \mathcal{I}' \models \mathsf{post}$) iff the following holds for all post-conditions φ/ψ in post: $\mathcal{I}' \models \psi$ whenever $\mathcal{I} \models \varphi$.

Definition 8 (Service Application). Let $\mathcal{T}$ be an acyclic TBox, $S = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ a service for $\mathcal{T}$, and $\mathcal{I}, \mathcal{I}'$ models of $\mathcal{T}$ sharing the same domain and interpretation of all individual names. Then S *may transform* $\mathcal{I}$ to $\mathcal{I}'$ ($\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$) iff

1. $\mathcal{I}, \mathcal{I}' \models \mathsf{post}$, and

2. there does not exist a model $\mathcal{J}$ of $\mathcal{T}$ such that $\mathcal{I}, \mathcal{J} \models \mathsf{post}$, $\mathcal{J} \neq \mathcal{I}'$, and $\mathcal{J} \preccurlyeq_{\mathcal{I},S,\mathcal{T}} \mathcal{I}'$.

The composite service $S_1 \ldots, S_k$ *may transform* $\mathcal{I}$ to $\mathcal{I}'$ ($\mathcal{I} \Rightarrow_{S_1,\ldots,S_k}^{\mathcal{T}} \mathcal{I}'$) iff there are models $\mathcal{I}_0, \ldots, \mathcal{I}_k$ of $\mathcal{T}$ with $\mathcal{I} = \mathcal{I}_0$, $\mathcal{I}' = \mathcal{I}_k$, and $\mathcal{I}_{i-1} \Rightarrow_{S_i}^{\mathcal{T}} \mathcal{I}_i$ for $1 \leq i \leq k$. If $\mathcal{T}$ is empty, we write $\Rightarrow_{S_1,\ldots,S_k}$ instead of $\Rightarrow_{S_1,\ldots,S_k}^{\mathcal{T}}$. $\triangle$

Note that this definition does not check whether the service is indeed executable, i.e., whether the pre-conditions are satisfied. It just says what the result of applying the service is, irrespective of whether it is executable or not.

Because of our restriction to acyclic TBoxes and primitive literals in the consequence part of post-conditions, services without occlusions are *deterministic*, i.e., for any model $\mathcal{I}$ of $\mathcal{T}$ there exists at most one model $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$. First note that there are indeed cases where there is no successor model $\mathcal{I}'$. In this case, we say that the service is *inconsistent with* $\mathcal{I}$. It is easy to see that this is the case iff there are post-conditions $\varphi_1/\psi, \varphi_2/\neg\psi \in \mathsf{post}$ such that both φ_1 and φ_2 are satisfied in $\mathcal{I}$. Second, assume that S is consistent with $\mathcal{I}$. The fact that there is exactly one model $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$ is an easy consequence of the next lemma, whose proof we leave as an easy exercise.

Lemma 9. *Let $\mathcal{T}$ be an acyclic TBox, $S = (\mathsf{pre}, \emptyset, \mathsf{post})$ a service for $\mathcal{T}$, and $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$ for models $\mathcal{I}, \mathcal{I}'$ of $\mathcal{T}$. If A is a primitive concept and s a role name, then*

$$A^{\mathcal{I}'} := \big(A^{\mathcal{I}} \cup \{b^{\mathcal{I}} \mid \varphi/A(b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\}\big) \setminus \{b^{\mathcal{I}} \mid \varphi/\neg A(b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\},$$

$$s^{\mathcal{I}'} := \big(s^{\mathcal{I}} \cup \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/s(a,b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\}\big) \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/\neg s(a,b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\}.$$

Since the interpretation of the defined concepts is uniquely determined by the interpretation of the primitive concepts and the role names, it follows that there cannot exist more than one $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$.

In principle, we could have started with this more transparent definition of the relation $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$ (with some adaptations to deal with occlusions). However, in Section 4 we will discuss possible extensions of our approach: for example, to cyclic TBoxes or post-conditions φ/ψ with more complex ABox assertions ψ. In these cases, services are no longer deterministic, and thus the above lemma does not hold. The PMA approach also provides these extensions with a semantics (though not necessarily with a satisfactory one).

2.3 Reasoning about Services

Assume that we want to apply a composite service $S_1, \ldots, S_k$ for the acyclic TBox $\mathcal{T}$. Usually, we do not have complete information about the world (i.e., the model $\mathcal{I}$ of $\mathcal{T}$ is not known completely). All we know are some facts about this world, i.e., we have an ABox $\mathcal{A}$, and all models of $\mathcal{A}$ together with $\mathcal{T}$ are considered to be possible states of the world.

Before trying to apply the service, we want to know whether it is indeed executable, i.e., whether all necessary pre-conditions are satisfied. If the service is executable, we may want to know whether applying it achieves the desired effect, i.e., whether an assertion that we want to make true really holds after executing the service. These problems are basic inference problems considered in the reasoning about action community, see e.g. [28]. In our setting, they can formally be defined as follows:

Definition 10 (Reasoning Services). Let $\mathcal{T}$ be an acyclic TBox, $S_1, \ldots, S_k$ a service for $\mathcal{T}$ with $S_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$, and $\mathcal{A}$ an ABox.

- *Executability:* $S_1, \ldots, S_k$ is *executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$* iff the following condition is true for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$:
  - $\mathcal{I} \models \mathsf{pre}_1$
  - for all i with $1 \leq i < k$ and all interpretations $\mathcal{I}'$ with $\mathcal{I} \Rightarrow_{S_1,\ldots,S_i}^{\mathcal{T}} \mathcal{I}'$, we have $\mathcal{I}' \models \mathsf{pre}_{i+1}$.
- *Projection:* an assertion φ is a *consequence of applying $S_1, \ldots, S_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$* iff, for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$, and all $\mathcal{I}'$ with $\mathcal{I} \Rightarrow_{S_1,\ldots,S_k}^{\mathcal{T}} \mathcal{I}'$, we have $\mathcal{I}' \models \varphi$.

If $\mathcal{T}$ is empty, we simply drop the phrase "w.r.t. $\mathcal{T}$" instead of writing "w.r.t. the empty TBox $\emptyset$". △

Note that executability alone does not guarantee that we cannot get stuck while executing a composite service. It may also happen that the service to be applied is inconsistent with the current interpretation. This cannot happen if we additionally know that all services S_i are *consistent with* $\mathcal{T}$ in the following sense: S_i is not inconsistent with any model $\mathcal{I}$ of $\mathcal{T}$. Summing up, to achieve an effect φ (an ABox assertion) starting from a world description $\mathcal{A}$ and given a TBox $\mathcal{T}$, we need a service $S_1, \ldots, S_k$ such that

$S_1, \ldots, S_k$ is executable in $\mathcal{A}$ w.r.t $\mathcal{T}$, S_i is consistent with $\mathcal{T}$ for $1 \leq i \leq k$, and φ is a consequence of applying $S_1, \ldots, S_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$.

We do not view consistency with the considered TBox $\mathcal{T}$ as a reasoning task, but rather as a condition that we generally expect to be satisfied by all well-formed services. Still, we should be able to decide whether a service is consistent with a TBox. This can be done by a reduction to standard DL reasoning: given the characterization of consistency *with a model* stated above Lemma 9, it is not difficult to see that an atomic service S with post-conditions post_i is consistent with a TBox $\mathcal{T}$ iff $\{\varphi_1/\psi, \varphi_2/\neg\psi\} \subseteq \mathsf{post}_i$ implies that the ABox $\{\varphi_1, \varphi_2\}$ is inconsistent w.r.t. $\mathcal{T}$.

In our example, all three services are consistent with $\mathcal{T}$. Given the ABox

$$\mathcal{A} = \{\texttt{parent}(a, d), \texttt{permanent\_resident}(a, \texttt{UK})\},$$

the composite service $S = S_1, S_2, S_3$ is executable, and $\texttt{receives\_c\_benef\_for}(a, d)$ is a consequence of applying S in $\mathcal{A}$ w.r.t. $\mathcal{T}$. Note that the presence of the TBox is crucial for this result.

The main aim of this paper is to show how the two reasoning tasks executability and projection can be computed, and how their complexity depends on the description logic used within our framework. There is one particularly simple case: for atomic services S, computing executability boils down to standard DL reasoning: S is executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}, \mathcal{T} \models \varphi$ for all $\varphi \in \mathsf{pre}$. Executability for composite services is less trivial, and the same holds for projection of both atomic and composite services. We show now that the two reasoning services can be mutually polynomially reduced to each other. This allows us to concentrate on projection when proving decidability and complexity results.

Lemma 11. *Executability and projection can be reduced in polynomial time to each other.*

Proof. Let $S_1, \ldots, S_k$ with $S_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ be a composite service for the acyclic TBox $\mathcal{T}$. This service is executable in the ABox $\mathcal{A}$ iff

(i) pre_1 is satisfied in every model of $\mathcal{A}$ and $\mathcal{T}$ and, for $1 \leq i < k$,

(ii) all assertions in pre_{i+1} are consequences of applying $S_1, \ldots, S_i$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$.

Condition (ii) is obviously a projection problem. Condition (i) can also be seen as a projection problem for the empty service $(\emptyset, \emptyset, \emptyset)$.

Conversely, assume that we want to know whether φ is a consequence of applying $S_1, \ldots, S_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$. We consider the composite service $S'_1, \ldots, S'_k, S'$, where $S'_i = (\emptyset, \mathsf{occ}_i, \mathsf{post}_i)$ for $1 \leq i \leq k$, and $S' = (\{\varphi\}, \emptyset, \emptyset)$. Then φ is a consequence of applying $S_1, \ldots, S_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $S'_1, \ldots, S'_k, S'$ is executable. ❑

It is interesting to note that the reduction of projection of a composite service $S = S_1, \ldots, S_k$ is to executability for services of length $k+1$. Indeed, we shall later see that, for some description logics, projection of atomic services is computationally harder than executability of atomic services.

2.4 Relation to Situation Calculus

We compare our formalism for reasoning about services with one of the most prominent (families of) formalisms for reasoning about actions, the *situation calculus* [28]. We show how to translate the components of our formalism, i.e., ABoxes, TBoxes, and service descriptions, into the situation calculus. Based on this translation, we then establish a correspondence between the reasoning problems. This correspondence shows that the consequences of a service application computed in our framework are identical to the consequences that would be computed in the situation calculus. In particular, this means that our solution of the frame problem is identical to Reiter's as initially proposed in [27].

The basic notions of the situation calculus are *actions*, which correspond to our atomic services, and situations, which can be viewed as first-order structures and roughly correspond to interpretations in our framework. The main purpose of the situation calculus is to provide a framework for axiomatizing (i) to which applications an action can be applied and (ii) the effect that actions have on situations. The former is achieved through so-called *action pre-condition axioms* while the effects of actions are described using so-called *successor state axioms*. Formally, the situation calculus is a three-sorted second-order theory, with the three sorts being actions, situations, and objects. Properties whose truth depends on the situation are represented by predicates that have one additional parameter of type situation. For example, "likes(b, c, s)" would be read as "b likes c in situation s". Such situation dependant predicates are called *fluents*. In its most common form, the situation calculus is restricted to *deterministic* effects of actions in the sense discussed in Section 2.2. Therefore, in this section we restrict ourselves to deterministic services, i.e., to services without occlusions—c.f. Lemma 9. For more details on the situation calculus, see [28, 16, 25].

The foundation for translating our service formalism into the situation calculus is provided by the standard translation of description logics into first order logic [2, 4]. For our purposes, this translation needs to be slightly modified: in the standard translation, concept names correspond to unary predicates, concepts correspond to first-oder formulae in one free variable, and role names correspond to binary predicates. In the situation calculus, all concept names and role names correspond to *fluents* since their extension depends on the actual situation. Thus, we need to extend the predicates corresponding to concept names and role names by one additional argument of type situation.

We now describe the modified translation for $\mathcal{ALCQIO}$ concepts. The translation is based on two recursive mappings $\pi_{x,s}(\cdot)$ and $\pi_{y,s}(\cdot)$, which translate $\mathcal{ALCQIO}$ concepts into a formula in one free object variable x or y and one free situation variable s. For each concept name A, we introduce a binary predicate of the same name with one place for objects and one for situations. For each role name r, we introduce a ternary

predicate of the same name with two places for objects and one for situations. And for each individual variable a, we introduce an object-typed constant $\mathbf{a}$. The mapping $\pi_{x,s}$ is defined as follows, and the mapping $\pi_{y,s}$ is defined like $\pi_{x,s}$ with the roles of x and y swapped:

$$\begin{array}{rcll}
\pi_{x,s}(A) & = & A(x,s), & \text{for concept names } A \in \mathsf{N_C} \\
\pi_{x,s}(\{a\}) & = & (x = \mathbf{a}), & \text{for nominals } N \in \mathsf{N_I} \\
\pi_{x,s}(\neg D) & = & \neg \pi_{x,s}(D), & \\
\pi_{x,s}(C \sqcap D) & = & \pi_{x,s}(C) \wedge \pi_{x,s}(D), & \\
\pi_{x,s}(C \sqcup D) & = & \pi_{x,s}(C) \vee \pi_{x,s}(D), & \\
\pi_{x,s}(\bowtie\ n\, r\, C) & = & \exists^{\bowtie n} y.r(x,y,s) \wedge \pi_{y,s}(C), & \\
\pi_{x,s}(\bowtie\ n\, r^-\, C) & = & \exists^{\bowtie n} y.r(y,x,s) \wedge \pi_{y,s}(C). &
\end{array}$$

where $\bowtie \in \{\geq, \leq\}$.

The next step is to translate ABoxes into first-order logic. This is easily achieved using the mapping $\pi_{x,s}$ that we have just introduced. In the following, $\varphi[\mathbf{a}/x]$ denotes the result of replacing each free occurrence of x in φ with the object constant $\mathbf{a}$:

$$\pi_s(\mathcal{A}) = \bigwedge_{C(b) \in \mathcal{A}} \pi_{x,s}(C)[\mathbf{b}/x] \wedge \bigwedge_{r(b,c) \in \mathcal{A}} r(\mathbf{b}, \mathbf{c}, s) \wedge \bigwedge_{\neg r(\mathbf{b},\mathbf{c}) \in \mathcal{A}} \neg r(\mathbf{b}, \mathbf{c}, s)$$

To translate a service description from our framework into a description of an action in situation calculus form, we need to translate the pre-conditions into action pre-condition axioms and the post-conditions into successor state axioms. We begin with the former. For each service $S = (\mathsf{pre}, \emptyset, \mathsf{post})$, we introduce an action-typed constant $\mathbf{u}$ and define an action pre-condition axiom $\mathsf{Poss}(s) \equiv \Pi_{\mathbf{u}}(s)$, which specifies whether it is possible to carry out the action $\mathbf{u}$ in a situation s as follows:[1]

$$\mathsf{Poss}(\mathbf{u}, s) \equiv \Pi_{\mathbf{u}}(s) := \bigwedge_{C(b) \in \mathsf{pre}} \pi_{x,s}(C)[\mathbf{b}/x](s) \wedge \bigwedge_{r(b,c) \in \mathsf{pre}} r(\mathbf{b}, \mathbf{c}, s) \wedge \bigwedge_{\neg r(b,c) \in \mathsf{pre}} \neg r(\mathbf{b}, \mathbf{c}, s).$$

To define successor state axioms, we fix a finite set of services $S_1, \ldots, S_n$ with $S_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ and associated action constant $\mathbf{u}_i$, for $1 \leq i \leq n$. Then, for each concept name A and each role name r, we introduce successor state axioms as follows, where u denotes an action-typed variable

$$A(x, \mathsf{do}(u, s)) \equiv \Phi_A(x, u, s) \quad \text{and} \quad r(x, y, \mathsf{do}(u, s)) \equiv \Phi_r(x, y, u, s)$$

[1] We use $\mathbf{u}$ for action constants and u for action variables instead of the more common $\mathbf{a}$ and a to avoid confusion with individual names and corresponding object constants.

with $\Phi_A(x,u,s)$ and $\Phi_r(x,y,u,s)$ defined as follows:

$$\Phi_A(x,u,s) := \bigvee_{\{(\varphi,b,i) \mid \varphi/A(b)\in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = b \wedge u = \mathbf{u}_i\big) \vee$$
$$A(x,s) \wedge \neg \bigvee_{\{(\varphi,b,i) \mid \varphi/\neg A(b)\in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = b \wedge u = \mathbf{u}_i\big)$$

$$\Phi_r(x,y,u,s) := \bigvee_{\{(\varphi,b,c,i) \mid \varphi/r(b,c)\in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = b \wedge y = c \wedge u = \mathbf{u}_i\big) \vee$$
$$r(x,y,s) \wedge \neg \bigvee_{\{(\varphi,b,c,i) \mid \varphi/\neg r(b,c)\in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = b \wedge y = c \wedge u = \mathbf{u}_i\big)$$

It is easily seen that the syntactic form of the formulas Φ_A and Φ_r is as required for successor state axioms in the situation calculus.[2] Reiter identifies a special form of successor state axioms, so-called *context-free* ones, for which there exists a particularly simple algorithm for regression, the basic computational mechanism of the situation calculus. It is interesting to note that our successor state axioms are context-free iff only unconditional post-conditions are used in services.

The only component of our formalism for reasoning about services that we have not yet translated into a situation calculus form are TBoxes. Indeed, there is no need to translate them, which can be seen as follows. Since we assume TBoxes to by acyclic, we may completely eliminate TBoxes using a process called *unfolding*: first, exhaustively replace each defined concept name appearing on the right-hand side of a concept definition in the TBox $\mathcal{T}$ with its defining concept description as given by $A \equiv C \in \mathcal{T}$. Second, replace all defined concept names in the ABox and service descriptions by their defining concept descriptions and drop the TBox [2]. This unfolding process preserves executability and consequences. For example, if $\mathcal{A}'$, S', and φ' are the result of unfolding a TBox $\mathcal{T}$ given the ABox $\mathcal{A}$, the atomic service S, and the assertion φ, then φ is a consequence of applying S in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff φ' is a consequence of applying S' in $\mathcal{A}'$. Note that the unfolding of TBoxes may lead to an exponential blowup in the size of ABox and services. For our purposes, however, this is irrelevant: we only carry out the translation to compare reasoning in the two formalisms, and not to actually use it for practical reasoning.

Now that all components of our framework have been translated to counterparts in the situation calculus, we show how an ABox $\mathcal{A}$ and a set of atomic services $S_1, \ldots, S_n$ can be translated into a *basic action theory* as defined in [28], where all free variables are assumed to be universally quantified:

- Σ is the set of the four *foundational axioms* for situations, where *do* is the binary function symbol of type $\mathit{action} \times \mathit{situation} \rightarrow \mathit{situation}$ relating a situation with the situation that is reached by executing an action, $\mathbf{s_0}$ is the situation-type constant denoting the initial situation, $\sqsubset$ is the binary predicate for defining an ordering

[2] More precisely, these formulas are *uniform* in the situation variable s. See [28] for more information.

on situations, and $s \sqsubseteq s'$ abbreviates $s \sqsubset s' \vee s = s'$:

$$do(u_1, s_1) = do(u_2, s_2) \rightarrow u_1 = u_2 \wedge s_1 = s_2$$
$$\forall P.\big(P(\mathbf{s_0}) \wedge \forall u, s.(P(s) \rightarrow P(do(u, s))) \rightarrow \forall s.P(s)\big)$$
$$\neg s \sqsubset \mathbf{s_0}$$
$$s \sqsubset do(u, s') \leftrightarrow s \sqsubseteq s'$$

- $\mathcal{D}_{ss}$ is the set of *successor state axioms*, one for each concept and role name occurring in $\mathcal{A}$ as defined above,
- $\mathcal{D}_{ap}$ is the set of *action pre-condition axioms*, one for each action-typed constant $\mathbf{u}_1, \ldots, \mathbf{u}_n$ corresponding to the services $S_1, \ldots, S_n$,
- $\mathcal{D}_{una}$ is the set of *unique name axioms* for actions:

$$\bigwedge_{1 \leq i < j \leq n} \mathbf{u}_i \neq \mathbf{u}_j$$

- $\mathcal{D}_{\mathbf{s_0}} := \pi_{\mathbf{s_0}}(\mathcal{A})$ is the description of the initial situation.

We use $\mathcal{D}(\mathcal{A}, S_1, \ldots, S_n)$ to denote the basic action theory obtained from $\mathcal{A}$ and $S_1, \ldots, S_n$, i.e., $\Sigma \cup \mathcal{D}_{\mathbf{s_0}} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una}$.

Finally, we compare our framework with reasoning in the situation calculus. The counterparts of our reasoning tasks executability and projection are defined as follows:

Executability. We introduce the following abbreviation:

$$executable(s) := \forall u, s' : \big((do(u, s') \sqsubseteq s) \rightarrow Poss(u, s')\big)$$

In the situation calculus, a sequence of actions $u_1, \ldots, u_n$ is *executable* in a situation s w.r.t. the basic action theory $\mathcal{D}$ if $\mathcal{D} \models executable(do(u_n, do(u_{n-1}, \cdots, do(u_1, s) \cdots)$.

Projection. Let $\varphi(s)$ be a situation calculus formula with one free situation-typed variable s. Then φ is a *consequence* of applying the sequence of actions denoted by constants $u_1, \ldots, u_n$ in the initial situation $\mathbf{s_0}$ described by $\mathcal{D}_{\mathbf{s_0}}$ w.r.t. the basic action theory $\mathcal{D}$ iff we have that $\mathcal{D} \models \varphi(do(u_n, do(u_{n-1}, \cdots, do(u_1, \mathbf{s_0}) \cdots)$.

We now formulate the main theorem of this section stating that reasoning in our formalism coincides with reasoning in the situation calculus.

Theorem 12. *Let $\mathcal{A}$ be an ABox, $S = S_1, \ldots, S_n$ a composite service, and φ an assertion. Then*

1. *S is executable in $\mathcal{A}$ iff the sequence $u_1, \ldots, u_n$ is executable in $\mathbf{s_0}$ w.r.t. the basic action theory $\mathcal{D}(\mathcal{A}, S_1, \ldots, S_n)$.*
2. *φ is a consequence of applying S in $\mathcal{A}$ iff $\pi_s(\{\varphi\})$ is a consequence of applying the sequence $u_1, \ldots, u_n$ in $\mathbf{s_0}$ w.r.t. the basic action theory $\mathcal{D}(\mathcal{A}, S_1, \ldots, S_n)$.*

Thus, the framework for reasoning about services presented in this paper is fully compatible not only with ontology languages based on description logics, but also with the situation calculus.

We should like to note that there is a more explicite way for dealing with TBoxes than unfolding: TBoxes can be translated into so-called *state* or *integrity constraints* of the situation calculus:

$$\pi_s(\mathcal{T}) \quad = \quad \bigwedge_{A \equiv C \in \mathcal{T}} \forall x.\pi_{x,s}(A) \leftrightarrow \pi_{x,s}(C).$$

To obtain an analogue of Theorem 12, we can then devise successor state axioms only for primitive concepts and role names, but not for defined concepts—although the latter are fluents. This corresponds to not minimizing defined concepts in Definition 7. Note that, if we admit also cyclic TBoxes, then the unfolding approach cannot be used any more and we are forced to translate into state constraints. This poses semantic problems as discussed in more detail in Section 4.2, in Appendix B of [28], and in [18].

3 Deciding Executability and Projection

The purpose of this section is to develop reasoning procedures for the reasoning services introduced in Section 2.3, and to analyze the computational complexity of executability and projection of different fragments of $\mathcal{ALCQIO}$. Throughout this section, we assume that all services are consistent with their TBox, and that TBoxes are acyclic.

By Lemma 11, we can restrict the attention to the projection problem. Basically, we solve this problem by an approach that is similar to the regression operation used in the situation calculus approach [28]. The main idea is to reduce projection, which considers sequences of interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_k$ obtained by service application, to standard reasoning tasks for a single interpretation $\mathcal{I}$. For the standard reasoning tasks, we consider two options:

Firstly, we may take care that the theory we obtain can again be expressed by a description logic TBox and ABox. This way, projection is reduced to ABox consequence in DL, from which we obtain decidability results and upper complexity bounds. Interestingly, when taking this approach, we cannot always stay within the DL we started with since we need to introduce nominals in the reduction. We prove lower complexity bounds for projection showing that the increase in complexity that is sometimes obtained by introducing nominals cannot be avoided.

Secondly, we can express the resulting theory in C^2, the two-variable fragment of first-order logic extended with counting quantifiers. This way, projection is reduced to satisfiability in C^2. We obtain a simpler reduction, but less sharp complexity results since satisfiability in C^2 is NExpTime-complete [24, 26], and thus quite costly from a computational perspective. However, there are two exceptional cases where we obtain a tight upper bound using the second translation, but not the first: $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$ with numbers in number restrictions coded in binary, i.e., the size of $(\geqslant n\ r\ C)$ and $(\leqslant n\ r\ C)$ is assumed to be $\log(n) + 1$ plus the size of C.

The following results are proved in this section:

Theorem 13. *Executability and projection of composite services w.r.t. acyclic TBoxes are*

1. PSPACE*-complete for* $\mathcal{ALC}$*,* $\mathcal{ALCO}$*,* $\mathcal{ALCQ}$*, and* $\mathcal{ALCQO}$ *if numbers in number restrictions are coded in unary;*

2. EXPTIME*-complete for* $\mathcal{ALCI}$ *and* $\mathcal{ALCIO}$*;*

3. *co-*NEXPTIME*-complete for* $\mathcal{ALCQI}$ *and* $\mathcal{ALCQIO}$*, regardless of whether numbers in number restrictions are coded in unary or binary.*

Thus, in all cases considered, the complexity of executability and projection for a description logic $\mathcal{L}$ coincides with the complexity of ABox consequence in $\mathcal{LO}$, the extension of $\mathcal{L}$ with nominals.

3.1 Reduction to DL Reasoning

We reduce projection in fragments $\mathcal{L}$ of $\mathcal{ALCQIO}$ to ABox consequence in the extension $\mathcal{LO}$ of $\mathcal{L}$ with nominals. Here, we assume unary coding of numbers in number restrictions, i.e., the size of $(\leqslant n\ r\ C)$ and $(\geqslant n\ r\ C)$ is assumed to be $n+1$ plus the size of C.

Theorem 14. *Let* $\mathcal{L} \in \{\mathcal{ALC}, \mathcal{ALCI}, \mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQ}, \mathcal{ALCQI}, \mathcal{ALCQIO}\}$*. Then projection of composite services formulated in* $\mathcal{L}$ *can be polynomially reduced to ABox consequence in* $\mathcal{LO}$ *w.r.t. acyclic TBoxes.*

Let $\mathcal{L}$ be one of the languages listed in Theorem 14, and let $\mathcal{A}$ be an ABox, $S_1, \ldots, S_n$ a composite service with $S_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$, $\mathcal{T}$ an acyclic TBox, and φ_0 an assertion, all formulated in $\mathcal{L}$. We are interested in deciding whether φ_0 is a consequence of applying $S_1, \ldots, S_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. Without loss of generality, we assume that φ_0 is of the form $A_0(a_0)$, for a concept name A_0:

1. Assertions $r(a,b)$ and $\neg r(a,b)$ can be replaced with $(\exists r.\{b\})(a)$ and $(\forall r.\neg\{b\})(a)$, respectively. This presupposes nominals, but nominals will be used in our reduction, anyway.

2. If $\varphi = C(a)$ with C not a concept name, we add a concept definition $A_0 \equiv C$ to the TBox $\mathcal{T}$, and then consider $\varphi = A_0(a)$.

In the following, we call $\mathcal{A}$, $\mathcal{T}$, $S_1, \ldots, S_n$, and φ_0 *the input*. We devise a reduction ABox $\mathcal{A}_{\mathsf{red}}$, an (acyclic) reduction TBox $\mathcal{T}_{\mathsf{red}}$, and a reduction assertion φ_{red} such that

$$\varphi_0 \text{ is a consequence of applying } S_1, \ldots, S_n \text{ in } \mathcal{A} \text{ w.r.t. } \mathcal{T} \text{ iff } \mathcal{A}_{\mathsf{red}}, \mathcal{T}_{\mathsf{red}} \models \varphi_{\mathsf{red}}.$$

The main idea of the reduction is to define $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ such that each *single* model of them encodes a *sequence* of interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ obtained by applying $S_1, \ldots, S_n$ in $\mathcal{A}$ (and *all* such sequences are encoded by reduction models). To ensure this, we use the following intuitions:

- The reduction ABox states that (i) the "$\mathcal{I}_0$-part" of a reduction model $\mathcal{I}$ is a model of $\mathcal{A}$, and that (ii) the $\mathcal{I}_i$-part of $\mathcal{I}$ satisfies the post-conditions post_i, for $1 \leq i \leq n$.

- The reduction TBox states that each $\mathcal{I}_i$ part of $\mathcal{I}$ is a model of $\mathcal{T}$, for $i \leq n$.

- We need to describe the law of inertia, i.e., the fact that we want to minimize the changes that are performed when applying a service. This task is split among the reduction ABox and TBox.

To understand the splitting mentioned in the third item, it is important to distinguish two kinds of elements in interpretations: we call an element $d \in \Delta^{\mathcal{I}}$ *named* if $a^{\mathcal{I}} = d$ for some individual a used in the input, and *unnamed* otherwise. Intuitively, the minimization of changes on named elements can be described in a direct way through the ABox $\mathcal{A}_{\mathsf{red}}$, while the minimization of changes on unnamed elements is achieved through a suitable encoding of $\mathcal{T}$ in $\mathcal{T}_{\mathsf{red}}$. Indeed, minimizing changes on unnamed elements boils down to enforcing that changes in concept (non)membership and role (non)membership involving (at least) one unnamed domain element *never* occur: due to the restriction to primitive concept names in post-conditions, our services are not expressive enough to enforce such changes.

In the reduction, we use the following concept names, role names, and individual names:

- The smallest set that contains all concepts appearing in the input and is closed under taking subconcepts is denoted with Sub. For every $C \in \mathsf{Sub}$ and every $i \leq n$, we introduce a concept name $T_C^{(i)}$. It will be ensured by the TBox $\mathcal{T}_{\mathsf{red}}$ that the concept name $T_C^{(i)}$ stands for the interpretation of C in the i-th interpretation.

- We use a concept name $A^{(i)}$ for every primitive concept name A used in the input and every $i \leq n$. Intuitively, $A^{(i)}$ represents the interpretation of the concept name A in the i-th interpretation, *but only with respect to the named domain elements.* Since concept membership of unnamed elements never changes, the "unnamed part" of the interpretation of the concept name A can always be found in $A^{(0)}$.

- We use a role name $r^{(i)}$ for every role name r used in the input and every $i \leq n$. Similarly to concept names, $r^{(i)}$ stands for the interpretation of r in the i-th interpretation, *but only records role relationships where both involved domain elements are named.*

- We use a concept name N that will be used to denote "named elements" of interpretations.

- The set of individual names used in the input is denoted with Obj. For every $a \in \mathsf{Obj}$, we introduce an auxiliary role name r_a.

- An auxiliary individual name $a_{\mathsf{help}} \notin \mathsf{Obj}$.

The reduction TBox $\mathcal{T}_{\mathsf{red}}$ consists of several components. The first component simply states that N denotes exactly the named domain elements:

$$\mathcal{T}_N := \Big\{ N \equiv \bigsqcup_{a \in \mathsf{Obj}} \{a\} \Big\}.$$

The second component $\mathcal{T}_{\mathsf{sub}}$ contains one concept definition for every $i \leq n$ and every concept $C \in \mathsf{Sub}$ that is not a defined concept name in $\mathcal{T}$. These concept definitions ensure that $T_C^{(i)}$ stands for the interpretation of C in the i-th interpretation as desired:

$$\begin{aligned}
T_A^{(i)} &\equiv (N \sqcap A^{(i)}) \sqcup (\neg N \sqcap A^{(0)}) \quad \text{if } A \text{ primitive in } \mathcal{T} && (a)\\
T_{\neg C}^{(i)} &\equiv \neg T_C^{(i)} && (b)\\
T_{C \sqcap D}^{(i)} &\equiv T_C^{(i)} \sqcap T_D^{(i)} && (c)\\
T_{C \sqcup D}^{(i)} &\equiv T_C^{(i)} \sqcup T_D^{(i)} && (d)\\
T_{(\geqslant m\ r\ C)}^{(i)} &\equiv \Big(N \sqcap \bigsqcup_{0 \leq j \leq m} \big((\geqslant j\ r^{(i)}\ (N \sqcap T_C^{(i)})) \sqcap (\geqslant (m-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)})) \big) \Big) && (e)\\
&\quad \sqcup \Big(\neg N \sqcap (\geqslant m\ r^{(0)}\ T_C^{(i)}) \Big)\\
T_{(\leqslant m\ r\ C)}^{(i)} &\equiv \Big(N \sqcap \bigsqcup_{0 \leq j \leq m} \big((\leqslant j\ r^{(i)}\ (N \sqcap T_C^{(i)})) \sqcap (\leqslant (m-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)})) \big) \Big) && (f)\\
&\quad \sqcup \Big(\neg N \sqcap (\leqslant m\ r^{(0)}\ T_C^{(i)}) \Big)
\end{aligned}$$

where $r^{-(i)}$ denotes $(r^{(i)})^-$ in the concept definitions for number restrictions. Line (a) reflects the fact that concept names $A^{(i)}$ only represent the extension of A in the i-th interpretation *for named domain elements*. To get $T_A^{(i)}$, the full extension of A in the i-th interpretation, we use $A^{(i)}$ for named elements and $A^{(0)}$ for unnamed ones. A similar splitting of role relationships into a named part and an unnamed part is reflected in the translation of number restrictions given in the last two lines.

Now we can assemble the reduction TBox $\mathcal{T}_{\mathsf{red}}$:

$$\mathcal{T}_{\mathsf{red}} \quad := \quad \mathcal{T}_{\mathsf{sub}} \cup \mathcal{T}_N \ \cup \{ T_A^{(i)} \equiv T_E^{(i)} \mid A \equiv E \in \mathcal{T},\ i \leq n \}$$

The last summand of $\mathcal{T}_{\mathsf{red}}$ ensures that all definitions from the input TBox $\mathcal{T}$ are satisfied by all interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$.

The reduction ABox $\mathcal{A}_{\mathsf{red}}$ also consists of several components. The first component ensures that, for each individual a occurring in the input, the auxiliary role r_a connects each individual (including a_{help}) with a, and only with a. This construction will simplify the definition of the other components of $\mathcal{A}_{\mathsf{red}}$:

$$\mathcal{A}_{\mathsf{aux}} := \big\{ a : \big(\exists r_b.\{b\} \sqcap \forall r_b.\{b\} \big) \mid a \in \mathsf{Obj} \cup \{a_{\mathsf{help}}\},\ b \in \mathsf{Obj} \big\}.$$

To continue, we first introduce the following abbreviations, for $i \leq n$:

$$\begin{aligned}
\mathsf{p}_i(C(a)) &:= \forall r_a.T_C^{(i)}\\
\mathsf{p}_i(r(a,b)) &:= \forall r_a.\exists r^{(i)}.\{b\}\\
\mathsf{p}_i(\neg r(a,b)) &:= \forall r_a.\forall r^{(i)}.\neg\{b\}.
\end{aligned}$$

The next component of $\mathcal{A}_{\mathsf{red}}$ formalizes satisfaction of the post-conditions. Note that its formulation relies on $\mathcal{A}_{\mathsf{aux}}$. For $1 \leq i \leq n$, we define

$$\mathcal{A}^{(i)}_{\mathsf{post}} := \{a_{\mathsf{help}} : \big(\mathsf{p}_{i-1}(\varphi) \to \mathsf{p}_i(\psi)\big) \mid \varphi/\psi \in \mathsf{post}_i\}.$$

The following component formalizes the minimization of changes on named elements. For $1 \leq i \leq n$ the ABox $\mathcal{A}^{(i)}_{\mathrm{min}}$ contains

1. the following assertions for every $a \in \mathsf{Obj}$ and every primitive concept name A with $A(a) \notin \mathsf{occ}_i$:

$$\begin{array}{rcl} a & : & \Big(\big(A^{(i-1)} \sqcap \bigsqcap_{\varphi/\neg A(a) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \to A^{(i)}\Big) \\ a & : & \Big(\big(\neg A^{(i-1)} \sqcap \bigsqcap_{\varphi/A(a) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \to \neg A^{(i)}\Big); \end{array}$$

2. the following assertions for all $a, b \in \mathsf{Obj}$ and every role name r with $r(a,b) \notin \mathsf{occ}_i$:

$$\begin{array}{rcl} a & : & \Big(\big(\exists r^{(i-1)}.\{b\} \sqcap \bigsqcap_{\varphi/\neg r(a,b) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \to \exists r^{(i)}.\{b\}\Big) \\ a & : & \Big(\big(\forall r^{(i-1)}.\neg\{b\} \sqcap \bigsqcap_{\varphi/r(a,b) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \to \forall r^{(i)}.\neg\{b\}\Big). \end{array}$$

The ABox $\mathcal{A}_{\mathsf{ini}}$ ensures that the first interpretation of the encoded sequence is a model of the input ABox $\mathcal{A}$:

$$\begin{array}{rcl} \mathcal{A}_{\mathsf{ini}} & := & \{T_C^{(0)}(a) \mid C(a) \in \mathcal{A}\} \cup \\ & & \{r^{(0)}(a,b) \mid r(a,b) \in \mathcal{A}\} \cup \\ & & \{\neg r^{(0)}(a,b) \mid \neg r(a,b) \in \mathcal{A}\}. \end{array}$$

We can now assemble $\mathcal{A}_{\mathsf{red}}$:

$$\begin{array}{rcl} \mathcal{A}_{\mathsf{red}} & := & \mathcal{A}_{\mathsf{ini}} \cup \mathcal{A}_{\mathsf{aux}} \cup \\ & & \mathcal{A}^{(1)}_{\mathsf{post}} \cup \cdots \cup \mathcal{A}^{(n)}_{\mathsf{post}} \cup \\ & & \mathcal{A}^{(1)}_{\mathsf{min}} \cup \cdots \cup \mathcal{A}^{(n)}_{\mathsf{min}}. \end{array}$$

Finally, the reduction assertion φ_{red} is defined as $T^{(n)}_{A_0}(a_0)$. Then we have the following.

Lemma 15. *φ is a consequence of applying $S_1, \ldots, S_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}}, \mathcal{T}_{\mathsf{red}} \models \varphi_{\mathsf{red}}$.*

Proof. We first introduce a few notions that we are going to use in the proof. With Con, we denote the set of concept names that appear in the input, with Prim concept names from the input which are primitive in $\mathcal{T}$, and with Rol the set of role names that appear in the input. Moreover, if $\mathcal{I}$ is an interpretation, we denote with $\mathsf{Obj}^{\mathcal{I}}$ the set $\{a^{\mathcal{I}} \mid a \in \mathsf{Obj}\}$. Finally, Assert will denote the set of assertions that appear in the input.

"$\Rightarrow$" We prove this direction by contraposition. Assume that $\mathcal{A}_{\mathsf{red}}, \mathcal{T}_{\mathsf{red}} \not\models \varphi_{\mathsf{red}}$. This means that there is an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, and

$\mathcal{J} \not\models T^{(n)}_{A_0}(a_0)$. In order to show that $\varphi = A_0(a_0)$ is not a consequence of applying $S_1, \ldots, S_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$, we have to find interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{S_i} \mathcal{I}_i$ for $1 \leq i \leq n$, and $\mathcal{I}_n \not\models A_0(a_0)$.

Let us define interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$, based on $\mathcal{J}$, in the following way:

- $\Delta^{\mathcal{I}_i} := \Delta^{\mathcal{J}}$
- $a^{\mathcal{I}_i} := a^{\mathcal{J}}$ for $a \in \mathsf{Obj}$
- $A^{\mathcal{I}_i} := (T^{(i)}_A)^{\mathcal{J}}$ for $A \in \mathsf{Con}$
- $r^{\mathcal{I}_i} := (r^{(i)})^{\mathcal{J}} \cap (N^{\mathcal{J}} \times N^{\mathcal{J}}) \cup (r^{(0)})^{\mathcal{J}} \cap \big(\Delta^{\mathcal{J}} \times (\neg N)^{\mathcal{J}} \cup (\neg N)^{\mathcal{J}} \times \Delta^{\mathcal{J}}\big)$ for $r \in \mathsf{Rol}$

Claim 1. For $i \leq n$, the following holds:

(a) If $a \in \mathsf{Obj}$, then $a^{\mathcal{I}_i} \in A^{\mathcal{I}_i}$ iff $a^{\mathcal{J}} \in (A^{(i)})^{\mathcal{J}}$, for all $A \in \mathsf{Prim}$
If $x \notin \mathsf{Obj}^{\mathcal{J}}$, then $x \in A^{\mathcal{I}_i}$ iff $x \in (A^{(0)})^{\mathcal{J}}$, for all $A \in \mathsf{Prim}$

(b) If $a, b \in \mathsf{Obj}$ then, for all $r \in \mathsf{Rol}$:

$$(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}) \in r^{\mathcal{I}_i} \text{ iff } (a^{\mathcal{J}}, b^{\mathcal{J}}) \in (r^{(i)})^{\mathcal{J}}$$

If $x \notin \mathsf{Obj}^{\mathcal{J}}$ or $y \notin \mathsf{Obj}^{\mathcal{J}}$ then, for all $r \in \mathsf{Rol}$:

$$(x, y) \in r^{\mathcal{I}_i} \text{ iff } (x, y) \in (r^{(0)})^{\mathcal{J}}$$

(c) $E^{\mathcal{I}_i} = (T^{(i)}_E)^{\mathcal{J}}$ for every $E \in \mathsf{Sub}$

(d) $\mathcal{I}_i \models \varphi$ iff $\mathcal{J} \models a : \mathsf{p}_i(\varphi)$ for all $\varphi \in \mathsf{Assert}$ and $a \in \mathsf{Obj} \cup \{a_{\mathsf{help}}\}$

Proof.

(a) follows from the fact that $A^{\mathcal{I}_i} = (T^{(i)}_A)^{\mathcal{J}} = ((A^{(i)})^{\mathcal{J}} \cap N^{\mathcal{J}}) \cup ((A^{(0)})^{\mathcal{J}} \cap (\neg N)^{\mathcal{J}})$ and $N^{\mathcal{J}} = \{a^{\mathcal{J}} \mid a \in \mathsf{Obj}\}$ (due to $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$ and the definition of $A^{\mathcal{I}_i}$).

(b) follows directly from the definition of $r^{\mathcal{I}_i}$.

(c) is proved by structural induction on E:

- $E = A$, where $A \in \mathsf{Con}$. We have that $A^{\mathcal{I}_i} = (T^{(i)}_A)^{\mathcal{J}}$ by definition of $\mathcal{I}_i$
- $E = \neg C$: $(\neg C)^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \setminus C^{\mathcal{I}_i} = \Delta^{\mathcal{J}} \setminus (T^{(i)}_C)^{\mathcal{J}} = (\neg T^{(i)}_C)^{\mathcal{J}} = (T^{(i)}_{\neg C})^{\mathcal{J}}$ holds since $C^{\mathcal{I}_i} = (T^{(i)}_C)^{\mathcal{J}}$ by induction, and since $\mathcal{J}$ satisfies (b) of $\mathcal{T}_{\mathsf{sub}}$.
- $E = C \sqcap D$: $(C \sqcap D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} = (T^{(i)}_C)^{\mathcal{J}} \cap (T^{(i)}_D)^{\mathcal{J}} = (T^{(i)}_C \sqcap T^{(i)}_D)^{\mathcal{J}} = (T^{(i)}_{C \sqcap D})^{\mathcal{J}}$ holds since $C^{\mathcal{I}_i} = (T^{(i)}_C)^{\mathcal{J}}$ and $D^{\mathcal{I}_i} = (T^{(i)}_D)^{\mathcal{J}}$ by induction, and since $\mathcal{J}$ satisfies (c) of $\mathcal{T}_{\mathsf{sub}}$.

– $E = C \sqcup D$ is similar to the previous case.

– $E = (\geq m\ r\ C)$: since $\mathcal{J}$ satisfies (e) of $\mathcal{T}_{\mathsf{sub}}$, we have that $x \in (T^{(i)}_{(\geq m\ r\ C)})^{\mathcal{J}}$ iff one of the following holds:

$$x \in \left(N \sqcap \bigsqcup_{0 \leq j \leq m} \left(\left(\geq j\ r^{(i)}\ (N \sqcap T_C^{(i)}) \right) \sqcap \left(\geq (m-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)}) \right) \right) \right)^{\mathcal{J}}$$

$$\text{or} \quad x \in \left(\neg N \sqcap \left(\geq m\ r^{(0)}\ T_C^{(i)} \right) \right)^{\mathcal{J}}$$

Thus, we obtain that:

$$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq m} \left(\begin{array}{l} \#\{y \mid (x,y) \in (r^{(i)})^{\mathcal{J}} \wedge y \in \left(N^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\right)\} \geq j \wedge \\ \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\right)\} \geq (m-j) \end{array} \right)$$

$$\text{or} \quad x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\right)\} \geq m$$

By induction, we have that $C^{\mathcal{I}_i} = (T_C^{(i)})^{\mathcal{J}}$. Thus, using the definition of $r^{\mathcal{I}_i}$, we have that the above disjunction holds iff:

$$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq m} \left(\begin{array}{l} \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \left(N^{\mathcal{J}} \cap C^{\mathcal{I}_i}\right)\} \geq j \wedge \\ \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap C^{\mathcal{I}_i}\right)\} \geq (m-j) \end{array} \right)$$

$$\text{or} \quad x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap C^{\mathcal{I}_i}\right)\} \geq m$$

By the semantics, this is equivalent to:

$$x \in (\geq m\ r\ C)^{\mathcal{I}_i}.$$

– $E = (\leq m\ r\ C)$: similar to the previous case

(d) follows directly from (b),(c), and the fact that

$$\mathcal{J} \models \{a : (\exists r_b.\{b\} \sqcap \forall r_b.\{b\}) \mid a \in \mathsf{Obj} \cup \{a_{\mathsf{help}}\},\ b \in \mathsf{Obj}\}.$$

This finishes the proof of the claim. Next, we will show that $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{S_i} \mathcal{I}_i$ for all $1 \leq i \leq n$, and $\mathcal{I}_n \not\models A_0(a_0)$:

- $\mathcal{I}_0 \models \mathcal{A}$: this follows immediately from Claim 1 ((b) and (c)) and $\mathcal{J} \models \mathcal{A}_{\mathsf{ini}}$.
- $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{S_i} \mathcal{I}_i$ for all $1 \leq i \leq n$ is split into three sub-tasks:
  - $\mathcal{I}_{i-1}, \mathcal{I}_i \models \mathsf{post}_i$: due to Claim 1(d), we have that:

    $$\text{If } \mathcal{J} \models a_{\mathsf{help}} : \left(\mathsf{p}_{i-1}(\varphi) \rightarrow \mathsf{p}_i(\psi)\right) \text{ then } \mathcal{I}_{i-1} \models \varphi \text{ implies } \mathcal{I}_i \models \psi$$

    Thus, $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{post}}$ implies $\mathcal{I}_{i-1}, \mathcal{I}_i \models \mathsf{post}_i$.

– $\mathcal{I}_i$ is minimal: assume, to the contrary of what is to be shown, that there is an interpretation $\mathcal{I}_i' \neq \mathcal{I}_i$ such that $\mathcal{I}_{i-1} \Rightarrow_{S_i}^{\mathcal{T}} \mathcal{I}_i'$ and $\mathcal{I}_i' \preccurlyeq_{\mathcal{I}_{i-1},S_i,\mathcal{T}} \mathcal{I}_i$. The latter can be split into two cases:

(i) $(A^{\mathcal{I}_{i-1}} \triangledown A^{\mathcal{I}_i'}) \setminus \{a^{\mathcal{I}_i} \mid A(a) \in \mathsf{occ}_i\} \subsetneq A^{\mathcal{I}_{i-1}} \triangledown A^{\mathcal{I}_i}$ for some $A \in \mathsf{Prim}$

Claim 1 (a) implies, for all $x \notin \mathsf{Obj}^{\mathcal{J}}$, that $x \in A^{\mathcal{I}_i}$ iff $x \in (A^{(0)})^{\mathcal{J}}$ iff $x \in A^{\mathcal{I}_{i-1}}$. Thus, there is some $a \in \mathsf{Obj}$ with $A(a) \notin \mathsf{occ}_i$, $a^{\mathcal{I}_i} \in A^{\mathcal{I}_{i-1}} \triangledown A^{\mathcal{I}_i}$, and $a^{\mathcal{I}_i} \notin A^{\mathcal{I}_{i-1}} \triangledown A^{\mathcal{I}_i'}$. Let us assume that $\mathcal{I}_{i-1} \models A(a)$, $\mathcal{I}_i' \models A(a)$ and $\mathcal{I}_i \not\models A(a)$. By Claim 1 (a), we have that $\mathcal{J} \models A^{(i-1)}(a)$ and $\mathcal{J} \not\models A^{(i)}(a)$. Then, due to $\mathcal{J} \models \mathcal{A}_{\mathrm{min}}^{(i)}$, we have that

$$a^{\mathcal{J}} \notin \Big(\bigsqcap_{\varphi/\neg A(a) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi) \Big)^{\mathcal{J}},$$

i.e. there is a $\varphi/\neg A(a) \in \mathsf{post}_i$ such that $\mathcal{J} \models a : \mathsf{p}_{i-1}(\varphi)$. By Claim 1(d) we obtain that $\mathcal{I}_{i-1} \models \varphi$. But then $\mathcal{I}_i' \models A(a)$ implies $\mathcal{I}_{i-1}, \mathcal{I}_i' \not\models \mathsf{post}_i$ in contradiction to $\mathcal{I}_{i-1} \Rightarrow_{S_i}^{\mathcal{T}} \mathcal{I}_i'$.
The case $\mathcal{I}_{i-1} \models \neg A(a)$, $\mathcal{I}_i' \models \neg A(a)$ and $\mathcal{I}_i \models A(a)$ is analogous.

(ii) $(r^{\mathcal{I}_{i-1}} \triangledown r^{\mathcal{I}_i'}) \setminus \{(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}) \mid r(a,b) \in \mathsf{occ}_i\} \subsetneq r^{\mathcal{I}_{i-1}} \triangledown r^{\mathcal{I}_i}$ for some $r \in \mathsf{Rol}$ is analogous to (i).

– $\mathcal{I}_i \models \mathcal{T}$, $i \leq n$, is an immediate consequence of Claim 1(c): Let $A \equiv E$ be a concept definition in $\mathcal{T}$. We have that $T_A^{(i)} \equiv T_E^{(i)} \in \mathcal{T}_{\mathsf{red}}$ and, since $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, obtain $(T_A^{(i)})^{\mathcal{J}} = (T_E^{(i)})^{\mathcal{J}}$. Claim 1(c) gives us that $A^{\mathcal{I}_i} = (T_A^{(i)})^{\mathcal{J}} = (T_E^{(i)})^{\mathcal{J}} = E^{\mathcal{I}_i}$.

- Finally, it is obvious that Claim 1(c) and $\mathcal{J} \not\models T_{A_0}^{(n)}(a_0)$ imply $\mathcal{I}_n \not\models A_0(a_0)$.

"$\Leftarrow$": For this direction, we also show the contrapositive. Assume that $\varphi = A_0(a_0)$ is not a consequence of applying $S_1, \ldots, S_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$. Then there are interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow_{S_i}^{\mathcal{T}} \mathcal{I}_i$ for $1 \leq i \leq n$, and $\mathcal{I}_n \not\models A_0(a_0)$. We show that, then, $T_{A_0}^{(n)}(a_0)$ is not a consequence of $\mathcal{A}_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$.

Claim 2 The interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ satisfy the following:

(a) For all $a \in \mathsf{Obj}$ and $A \in \mathsf{Prim}$ such that $A(a) \notin \mathsf{occ}_i$, the following holds:

if $\mathcal{I}_{i-1} \models A(a)$ and, for each $\varphi/\neg A(a) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models A(a)$, and

if $\mathcal{I}_{i-1} \models \neg A(a)$ and, for each $\varphi/A(a) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models \neg A(a)$.

For all $a, b \in \mathsf{Obj}$ and $r \in \mathsf{Rol}$ such that $r(a,b) \notin \mathsf{occ}_i$, the following holds:

if $\mathcal{I}_{i-1} \models r(a,b)$ and, for each $\varphi/\neg r(a,b) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models r(a,b)$

if $\mathcal{I}_{i-1} \models \neg r(a,b)$ and, for each $\varphi/r(a,b) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models \neg r(a,b)$.

(b) If $x \notin \mathsf{Obj}^{\mathcal{I}_0}$, then $x \in A^{\mathcal{I}_i}$ iff $x \in A^{\mathcal{I}_0}$, for all $A \in \mathsf{Prim}$

(c) If $x \notin \mathsf{Obj}^{\mathcal{I}_0}$ or $y \notin \mathsf{Obj}^{\mathcal{I}_0}$, then $(x, y) \in r^{\mathcal{I}_i}$ iff $(x, y) \in r^{\mathcal{I}_0}$, for all $r \in \mathsf{Rol}$

Proof. Assume that **(a)** does not hold, i.e., that, for example,

$$\mathcal{I}_{i-1} \models A(a) \text{ and, for each } \varphi/\neg A(a) \in \mathsf{post}_i,\ \mathcal{I}_{i-1} \not\models \varphi \text{ and } \mathcal{I}_i \models \neg A(a)$$

for an $a \in \mathsf{Obj}$ and an $A \in \mathsf{Prim}$ such that $A(a) \notin \mathsf{occ}_i$. Define an interpretation $\mathcal{I}_i'$ such that it interpretes all primitive concept and role names in the same way as $\mathcal{I}_i$ with the exception that $a^{\mathcal{I}_i'} \in A^{\mathcal{I}_i'}$. Since the TBox $\mathcal{T}$ is acyclic we can define $B^{\mathcal{I}_i'}$ for defined concept names B, such that $\mathcal{I}_i' \models \mathcal{T}$. Then we have that $\mathcal{I}_i' \neq \mathcal{I}_i$, $\mathcal{I}_i' \preccurlyeq_{\mathcal{I}_{i-1}, S_i, \mathcal{T}} \mathcal{I}_i$, $\mathcal{I}_i' \models \mathcal{T}$, and $\mathcal{I}_{i-1}, \mathcal{I}_i' \models \mathsf{post}$. But this implies that $\mathcal{I}_{i-1} \not\Rightarrow^{\mathcal{T}}_{S_i} \mathcal{I}_i$.

(b) In an analogous way to (a), we can show that $x \in A^{\mathcal{I}_i}$ iff $x \in A^{\mathcal{I}_{i-1}}$ for all $x \notin \mathsf{Obj}^{\mathcal{I}_0}$ and $1 \leq i \leq n$. As an immediate consequence, we obtain (b).
(c) Analogous to (b).

This finishes the proof of Claim 2.

We define an interpretation $\mathcal{J}$ in the following way:

- $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}_0}$ $(= \Delta^{\mathcal{I}_1} = \cdots = \Delta^{\mathcal{I}_n})$
- $a^{\mathcal{J}} := a^{\mathcal{I}_0}$ $(= a^{\mathcal{I}_1} = \cdots = a^{\mathcal{I}_n})$ for $a \in \mathsf{Obj}$
- $a^{\mathcal{J}}_{\mathsf{help}} := d$, for an arbitrary $d \in \Delta^{\mathcal{J}}$
- $N^{\mathcal{J}} := \{a^{\mathcal{J}} \mid a \in \mathsf{Obj}\}$
- $r^{\mathcal{J}}_b := \{(a^{\mathcal{J}}, b^{\mathcal{J}}) \mid a \in \mathsf{Obj} \cup \{a_{\mathsf{help}}\}\}$, for all $b \in \mathsf{Obj}$
- $(A^{(i)})^{\mathcal{J}} := A^{\mathcal{I}_i}$ for $A \in \mathsf{Con}$ and $i \leq n$
- $(r^{(i)})^{\mathcal{J}} := r^{\mathcal{I}_i}$ for $r \in \mathsf{Rol}$ and $i \leq n$
- $(T^{(i)}_C)^{\mathcal{J}} := C^{\mathcal{I}_i}$ for all $C \in \mathsf{Sub}$ and $i \leq n$

Please note that the definition of $\mathcal{J}$ implies that, for all $i \leq n$, $\varphi \in \mathsf{Assert}$, and $a \in \mathsf{Obj} \cup \{a_{\mathsf{help}}\}$, we have the following:

$$\mathcal{I}_i \models \varphi \quad \text{iff} \quad \mathcal{J} \models a : \mathsf{p}_i(\varphi) \tag{$*$}$$

We will show now that $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, and $\mathcal{J} \not\models T_{A_0}(a_0)$.

(i) $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$:

- $\mathcal{J} \models \mathcal{A}_{\mathsf{ini}}$ follows directly from $\mathcal{I}_0 \models \mathcal{A}$ and the definition of $\mathcal{J}$.
- $\mathcal{J}$ models $\mathcal{A}_{\mathsf{aux}}$ by definition of $r^{\mathcal{J}}_b$.

- $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{post}}$, for each $1 \leq i \leq n$: by $(*)$, we obtain that:

$$\text{If } \mathcal{I}_{i-1} \models \varphi \text{ implies } \mathcal{I}_i \models \psi, \text{ then } \mathcal{J} \models a_{\mathsf{help}} : (\mathsf{p}_1(\varphi) \to \mathsf{p}_2(\psi))$$

  for every $\varphi/\psi \in \mathsf{post}_i$. Since $\mathcal{I}_{i-1}, \mathcal{I}_i \models \mathsf{post}_i$, we have that $\mathcal{J}$ satisfies $\mathcal{A}^{(i)}_{\mathsf{post}}$.

- $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{min}}$, for each $1 \leq i \leq n$: by $(*)$ and definitions of $(A^{(i)})^{\mathcal{J}}$, we have that

$$\begin{array}{rl} \text{If} & \mathcal{I}_{i-1} \models A(a) \text{ and, for all } \varphi/\neg A(a) \in \mathsf{post}_i,\ \mathcal{I}_{i-1} \not\models \varphi \text{ imply } \mathcal{I}_i \models A(a), \\ \text{then} & \mathcal{J} \models a : \Big(\big(A^{(i-1)} \sqcap \bigsqcap_{\varphi/\neg A(a) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \to A^{(i)} \Big) \end{array}$$

  The symmetric case for $(\neg A)^{\mathcal{J}}$ and the cases for $(r^{(i)})^{\mathcal{J}}$ can be considered in a similar way. Thus, by Claim 2(a), we have that $\mathcal{J}$ satisfies $\mathcal{A}^{(i)}_{\mathsf{min}}$.

(ii) $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$:

- $\mathcal{J}$ satisfies the concept definition $N \equiv \bigsqcup_{a \in \mathsf{Obj}} \{a\}$ by definition of $N^{\mathcal{J}}$.

- $\mathcal{J}$ satisfies every concept definition $T^{(i)}_A \equiv T^{(i)}_C$, where $A \equiv C \in \mathcal{T}$ and $i \leq n$: by definition of $(T^{(i)}_A)^{\mathcal{J}}$ and since $\mathcal{I}_i \models \mathcal{T}$, we have that $(T^{(i)}_A)^{\mathcal{J}} = A^{\mathcal{I}_i} = C^{\mathcal{I}_i} = (T^{(i)}_C)^{\mathcal{J}}$.

- Finally, we will show that $\mathcal{J} \models \mathcal{T}_{\mathsf{sub}}$. By structural induction on $E \in \mathsf{Sub}$, we show that $\mathcal{J}$ satisfies every concept definition with $T^{(i)}_E$ on the left-hand side:

  - $E = A$, where $A \in \mathsf{Prim}$. We have:

$$\begin{array}{rcl} (T^{(i)}_A)^{\mathcal{J}} & = & A^{\mathcal{I}_i} = N^{\mathcal{J}} \cap A^{\mathcal{I}_i} \cup (\neg N)^{\mathcal{J}} \cap A^{\mathcal{I}_i} \\ & = & N^{\mathcal{J}} \cap A^{\mathcal{I}_i} \cup (\neg N)^{\mathcal{J}} \cap A^{\mathcal{I}_0} = N^{\mathcal{J}} \cap (A^{(i)})^{\mathcal{J}} \cup (\neg N)^{\mathcal{J}} \cap (A^{(0)})^{\mathcal{J}} = \\ & = & ((N \sqcap A^{(i)}) \sqcup (\neg N \sqcap A^{(0)}))^{\mathcal{J}} \end{array}$$

    The first equality holds by definition of $(T^{(i)}_A)^{\mathcal{J}}$, the second one by the semantics, the third one by Claim 2 (b) and the definition of $N^{\mathcal{J}}$, the fourth one by definition of $(A^{(j)})^{\mathcal{J}}$, and the last one by the semantics.

  - $E = \neg C$. By definition of $(T^{(i)}_{\neg C})^{\mathcal{J}}$ and $(T^{(i)}_C)^{\mathcal{J}}$, we have the following:

$$(T^{(i)}_{\neg C})^{\mathcal{J}} = (\neg C)^{\mathcal{I}_i} = \neg C^{\mathcal{I}_i} = \neg (T^{(i)}_C)^{\mathcal{J}}.$$

  - $E = C \sqcap D$. By definition of $(T^{(i)}_{C \sqcap D})^{\mathcal{J}}$, $(T^{(i)}_C)^{\mathcal{J}}$ and $(T^{(i)}_D)^{\mathcal{J}}$, we have the following:

$$(T^{(i)}_{C \sqcap D})^{\mathcal{J}} = (C \sqcap D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} = (T^{(i)}_C)^{\mathcal{J}} \cap (T^{(i)}_D)^{\mathcal{J}} = (T^{(i)}_C \sqcap T^{(i)}_D)^{\mathcal{J}}$$

  - $E = C \sqcup D$ is similar to the previous case.

– $E = (\geq\ m\ r\ C)$. By definition of $\mathcal{J}$, we have that $x \in (T^{(i)}_{(\geq m\ r\ C)})^{\mathcal{J}}$ iff $x \in (\geq\ m\ r\ C)^{\mathcal{I}_i}$. Due to the definition of the semantics, the latter is the case iff $\#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\} \geq m$.
By Claim 2(c) and $N^{\mathcal{J}} = \{a^{\mathcal{J}} \mid a \in \mathsf{Obj}\}$, the above expression is equivalent to the following disjunction:

$$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq m} \Big(\begin{array}{l} \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \big(N^{\mathcal{J}} \cap C^{\mathcal{I}_i}\big)\} \geq j \wedge \\ \#\{y \mid (x,y) \in r^{\mathcal{I}_0} \wedge y \in \big((\neg N)^{\mathcal{J}} \cap C^{\mathcal{I}_i}\big)\} \geq (m-j) \end{array} \Big)$$

or
$x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in r^{\mathcal{I}_0} \wedge y \in C^{\mathcal{I}_i}\} \geq m$

Using the definitions of $(T^{(i)}_C)^{\mathcal{J}}$ and $(r^{(i)})^{\mathcal{J}}$, we obtain:

$$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq m} \Big(\begin{array}{l} \#\{y \mid (x,y) \in (r^{(i)})^{\mathcal{J}} \wedge y \in \big(N^{\mathcal{J}} \cap (T^{(i)}_C)^{\mathcal{J}}\big)\} \geq j \wedge \\ \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in \big((\neg N)^{\mathcal{J}} \cap (T^{(i)}_C)^{\mathcal{J}}\big)\} \geq (m-j) \end{array} \Big)$$

or
$x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in (T^{(i)}_C)^{\mathcal{J}}\} \geq m,$

which is equivalent to:

$$x \in \Big[\Big(N \sqcap \bigsqcup_{0 \leq j \leq m} \Big(\big(\geq j\ r^{(i)}\ (N \sqcap T^{(i)}_C)\big) \sqcap \big(\geq (m-j)\ r^{(0)}\ \neg(N \sqcap T^{(i)}_C)\big) \Big) \Big) \sqcup \Big((\neg N)^{\mathcal{J}} \sqcap \big(\geq m\ r^{(0)}\ T^{(i)}_C\big) \Big) \Big]^{\mathcal{J}}.$$

– $E = (\leq\ m\ r\ C)$ is similar to the previous case.

Hence $\mathcal{J}$ satisfies $\mathcal{T}_{\mathsf{sub}}$.

(iii) Finally, it is easy to see that $(A_0)^{\mathcal{I}_n} = (T^{(n)}_{A_0})^{\mathcal{J}}$ and $\mathcal{I}_n \not\models A_0(a_0)$ imply $\mathcal{J} \not\models T^{(n)}_{A_0}(a_0)$.

❑

Since the size of $\mathcal{A}_{\mathsf{red}}$, $\mathcal{T}_{\mathsf{red}}$, and φ_{red} are clearly polynomial in the size of the input (recall that we assume unary coding of numbers in number restrictions), Lemma 15 immediately yields Theorem 14. Thus, for the DLs $\mathcal{L}$ considered in Theorem 14, upper complexity bounds for ABox consequence in $\mathcal{LO}$ carry over to projection in $\mathcal{L}$. Many such upper bounds are available from the literature. Indeed, there is only one case where we cannot draw upon existing results: the complexity of ABox consequence in $\mathcal{ALCQO}$ w.r.t. acyclic TBoxes. For the sake of completeness, we prove that this problem is PSPACE-complete in Appendix A. Lower complexity bounds carry over from ABox consequence in a DL $\mathcal{L}$ to projection in the same DL: $\mathcal{A}, \mathcal{T} \models \varphi$ iff φ is a consequence of applying the empty service $(\emptyset, \emptyset, \emptyset)$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$. Thus, we obtain tight bounds for projection in those DLs $\mathcal{L}$ where the addition of nominals does *not* increase the complexity of reasoning.

Corollary 16. *Executability and projection w.r.t. acyclic TBoxes are*

1. PSPACE*-complete for* $\mathcal{ALC}$*,* $\mathcal{ALCO}$*,* $\mathcal{ALCQ}$*, and* $\mathcal{ALCQO}$*;*
2. *in* EXPTIME *for* $\mathcal{ALCI}$*;*
3. EXPTIME*-complete for* $\mathcal{ALCIO}$*;*
4. *in co-*NEXPTIME *for* $\mathcal{ALCQI}$*;*
5. *co-*NEXPTIME*-complete for* $\mathcal{ALCQIO}$*.*

Points 1, 4, and 5 presuppose that numbers in number restrictions are coded in unary.

Proof. The corollary is a consequence of Theorem 14 and the following results: ABox consequence in

- $\mathcal{ALC}$ w.r.t. acyclic TBoxes is PSPACE-hard [30] (yields lower bounds of Point 1);
- $\mathcal{ALCQO}$ w.r.t. acyclic TBoxes is in PSPACE, which is proved in Appendix A (yields upper bounds of Point 1);
- $\mathcal{ALCIO}$ w.r.t. acyclic TBoxes is EXPTIME-complete, as follows from results in [1] (yields Points 2 and 3);
- $\mathcal{ALCQIO}$ is co-NEXPTIME-complete as follows from results in [36] and [24] (yields Points 4 and 5).

The bounds for executability are then obtained by the reductions of executability to projection and vice versa. ❑

In Section 3.3, we prove matching lower bounds for Points 2 and 4 of Corollary 13. Note that the stated upper bounds increase by one exponential if numbers in number restrictions are coded in binary: in this case, the size of the reduction TBox $\mathcal{T}_{\mathsf{red}}$ is exponential in the size of the input.

3.2 Reduction to C^2

Alternatively to reducing to standard DL reasoning, we can reduce projection to satisfiability in C^2. This yields a simpler translation and a co-NEXPTIME upper bound for projection in $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$ with numbers in number restrictions coded in binary—in contrast to the reduction given in the previous section which requires unary coding to yield co-NEXPTIME upper bounds. However, we cannot get any PSPACE or EXPTIME upper bounds from the C^2-translation since satisfiability in C^2 is NEXPTIME-complete [24, 26].

We assume that the two variables of C^2 are called x and y, and write $\exists^{\leq n} x.\varphi(x)$ and $\exists^{\geq n} x.\varphi(x)$ for the counting quantifiers. We show the following.

Theorem 17. *Projection of composite services formulated in* $\mathcal{ALCQIO}$ *can be polynomially reduced to satisfiability in* C^2*.*

As in the previous section, we assume that *an input* is given, namely an acyclic TBox $\mathcal{T}$, an ABox $\mathcal{A}$, a composite service with $S_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$, and an assertion φ_0, all formulated in $\mathcal{ALCQIO}$, and that we are interested in deciding whether φ_0 is a consequence of applying S in $\mathcal{A}$ w.r.t. $\mathcal{T}$. As in Section 3.1, we assume that φ_0 is of the form $A_0(a_0)$ with A_0 a concept name. The idea underlying the reduction is very similar to that underlying the reduction presented in the previous section, apart from one significant simplification: since C^2 is more expressive than $\mathcal{ALCQIO}$, it is not necessary to split the interpretations of concept and role names into a named part and an unnamed part. We introduce the following signature for the reduction formula φ_{red} that we are about to craft:

- Again, the smallest set that contains all concepts appearing in the input and that is closed under taking subconcepts is denoted by Sub. We introduce a unary predicate $P_C^{(i)}$ for every concept $C \in \mathsf{Sub}$. Intuitively, $P_C^{(i)}$ represents the extension of C in the i-th interpretation.
- Again, let $\mathsf{RoI} := \{r, r^- \mid \text{ the role name } r \text{ occurs in the input}\}$. We introduce a binary predicate $P_r^{(i)}$ for every $r \in \mathsf{RoI}$ and every $i \leq n$;
- Again, the set of individual names used in the input is denoted with Obj. For every $a \in \mathsf{Obj}$, we introduce a constant c_a.

We start with translating the TBox $\mathcal{T}$ into a C^2-formula. The formula φ_{sub} contains one conjunct for every $i \leq n$ and every concept $C \in \mathsf{Sub}$ that is not a concept name:

$$\begin{array}{lrcl}
\forall x. & P^{(i)}_{\neg C}(x) & \leftrightarrow & \neg P^{(i)}_C(x) \\
\forall x. & P^{(i)}_{C \sqcap D}(x) & \leftrightarrow & P^{(i)}_C(x) \wedge P^{(i)}_D(x) \\
\forall x. & P^{(i)}_{C \sqcup D}(x) & \leftrightarrow & P^{(i)}_C(x) \vee P^{(i)}_D(x) \\
\forall x. & P^{(i)}_{(\geqslant m\ r\ C)}(x) & \leftrightarrow & \exists^{\geq m} y. P^{(i)}_r(x,y) \wedge P^{(i)}_C(y) \\
\forall x. & P^{(i)}_{(\leqslant m\ r\ C)}(x) & \leftrightarrow & \exists^{\leq m} y. P^{(i)}_r(x,y) \wedge P^{(i)}_C(y)
\end{array}$$

Next, the formula $\varphi_{\mathcal{T}}$ is defined as follows (recall that $\mathsf{N_R}$ is the set of all role names):

$$\begin{aligned}
\varphi_{\mathcal{T}} := \varphi_{\mathsf{sub}} \wedge \quad & \bigwedge_{A \equiv E \in \mathcal{T}} \bigwedge_{i \leq n} \forall x. (P^{(i)}_A(x) \leftrightarrow P^{(i)}_C(x)) \wedge \\
& \bigwedge_{r \in \mathsf{RoI} \cap \mathsf{N_R}} \bigwedge_{i \leq n} \forall x, y. (P^{(i)}_r(x,y) \leftrightarrow P^{(i)}_{r^-}(y,x))
\end{aligned}$$

Observe that the last line ensures a correct interpretation of inverse roles.

Next, we translate the input ABox and input services. Similar to the previous reduction, we introduce the following abbreviations, for $i \leq n$:

$$\begin{array}{rcl}
\mathsf{p}_i(C(a)) & := & P^{(i)}_C(c_a) \\
\mathsf{p}_i(r(a,b)) & := & P^{(i)}_r(c_a, c_b) \\
\mathsf{p}_i(\neg r(a,b)) & := & \neg P^{(i)}_r(c_a, c_b)
\end{array}$$

The formula $\varphi_{\mathcal{A},S}$ is defined as follows:

$$\varphi_{\mathcal{A},S} := \bigwedge_{\varphi \in \mathcal{A}} p_0(\varphi) \wedge \bigwedge_{1 \leq i \leq n} \bigwedge_{\varphi/\psi \in \mathsf{post}_i} \mathsf{p}_{i-1}(\varphi) \rightarrow \mathsf{p}_i(\psi)$$

Next, we construct a formula that ensures the minimality of changes made when applying a service. For $1 \leq i \leq n$, let $\varphi^{(i)}_{\mathsf{min}}$ be the conjunction of the following formulas:

1. for every $a \in \mathsf{Obj}$ and every primitive concept name A with $A(a) \notin \mathsf{occ}_i$,

$$\big(P_A^{(i-1)}(c_a) \wedge \bigwedge_{\varphi/\neg A(a) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \rightarrow P_A^{(i)}(c_a)$$

$$\big(\neg P_A^{(i-1)}(c_a) \wedge \bigwedge_{\varphi/A(a) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \rightarrow \neg P_A^{(i)}(c_a)$$

2. for all $a, b \in \mathsf{Obj}$ and every role name r with $r(a,b) \notin \mathsf{occ}_i$:

$$\big(P_r^{(i-1)}(c_a, c_b) \wedge \bigwedge_{\varphi/\neg r(a,b) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \rightarrow P_r^{(i)}(c_a, c_b)$$

$$\big(\neg P_r^{(i-1)}(c_a, c_b) \wedge \bigwedge_{\varphi/r(a,b) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \rightarrow \neg P_r^{(i)}(c_a, c_b)$$

3. for every concept name A occurring in the input:

$$\forall x. \big(\neg(P_A^{(i-1)}(x) \leftrightarrow P_A^{(i)}(x)) \rightarrow \bigvee_{a \in \mathsf{Obj}} x = c_a\big)$$

4. for every role name r occurring in the input:

$$\forall x, y. \big(\neg(P_r^{(i-1)}(x,y) \leftrightarrow P_r^{(i)}(x,y)) \rightarrow (\bigvee_{a \in \mathsf{Obj}} x = c_a \vee \bigvee_{a \in \mathsf{Obj}} y = c_a)\big)$$

Observe that the last two items have no direct counterpart in the reduction given in the previous section. Intuitively, the formula from Item 3 says that a service application will not add or delete an element d to or from a set $A^{\mathcal{I}}$ if d is unnamed. The formula from Item 4 makes the analogous statement for role names. These statements cannot be expressed in $\mathcal{ALCQIO}$ which necessitated the splitting of the interpretations of concept and role names into a named and an unnamed part in the previous reduction.

The formula φ_{min} is defined as follows:

$$\varphi_{\mathsf{min}} := \bigwedge_{1 \leq i \leq n} \varphi^{(i)}_{\mathsf{min}}.$$

Finally, φ_0^* is defined as $P_{A_0}^{(n)}(c_{a_0})$ and

$$\varphi_{\mathsf{red}} := \varphi_{\mathcal{A},S} \wedge \varphi_{\mathsf{min}} \wedge \varphi_{\mathcal{T}} \wedge \neg \varphi_0^*$$

The following lemma can be proved analogously to Lemma 15.

Lemma 18. *φ_0 is a consequence of applying $S_1, \ldots, S_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff φ_{red} is unsatisfiable.*

Together with the reduction from executability to projection, we obtain the following result.

Corollary 19. *Executability and projection w.r.t. acyclic TBoxes are in co-*NExpTime *for $\mathcal{ALCQIO}$ even if the numbers in number restrictions are coded in binary.*

A matching lower bound for $\mathcal{ALCQIO}$ is obtained from Point 5 of Corollary 13. As shown in the following section, Corollary 19 also yields a tight upper bound for the fragment $\mathcal{ALCQI}$ of $\mathcal{ALCQIO}$.

3.3 Hardness Results

We show that the upper bounds for executability and projection obtained in the previous two sections cannot be improved. In Section 3.1, we have already obtained matching lower bounds for DLs $\mathcal{L}$ where the complexity of ABox inconsistency coincides in $\mathcal{L}$ and $\mathcal{LO}$ ($\mathcal{L}$'s extension with nominals). It thus remains to consider cases where ABox inconsistency in $\mathcal{LO}$ is harder than in $\mathcal{L}$: we prove an ExpTime lower bound for projection in $\mathcal{ALCI}$ and a co-NExpTime lower bound for projection in $\mathcal{ALCQI}$ with numbers coded in unary. By Lemma 11, these bounds carry over to executability. They match Points 2 and 4 of Corollary 13 and, together with Corollary 19, establish co-NExpTime-completeness of projection in $\mathcal{ALCQI}$. The results established in this section show that the additional complexity that is obtained by introducing nominals in the reduction of projection to ABox consequence cannot be avoided.

The idea for proving the lower bounds is to reduce, for $\mathcal{L} \in \{\mathcal{ALCI}, \mathcal{ALCQI}\}$, unsatisfiability of $\mathcal{LO}$ concepts to projection in $\mathcal{L}$. In the case of $\mathcal{ALCQI}$, we can even obtain a slightly stronger result by reducing concept unsatisfiability in $\mathcal{ALCFIO}$ to projection in $\mathcal{ALCFI}$, where $\mathcal{ALCFIO}$ is $\mathcal{ALCQIO}$ with numbers occurring in number restrictions limited to $\{0, 1\}$, and $\mathcal{ALCFI}$ is obtained from $\mathcal{ALCFIO}$ by dropping nominals.[3] Observe that the coding of numbers, i.e. unary vs. binary, is not an issue in $\mathcal{ALCFIO}$ and $\mathcal{ALCFI}$, and thus a lower bound for projection in $\mathcal{ALCFI}$ implies the same bound for projection in $\mathcal{ALCQI}$ with unary coding of numbers. Our aim is to prove the following.

Theorem 20. *There exists an ABox $\mathcal{A}$ and an atomic service S formulated in $\mathcal{ALCI}$ ($\mathcal{ALCFI}$) such that the following tasks are* ExpTime*-hard (co-*NExpTime*-hard): given an assertion φ,*

- *decide whether φ is a consequence of applying S in $\mathcal{A}$;*
- *decide whether $S, (\{\varphi\}, \emptyset, \emptyset)$ is executable in $\mathcal{A}$.*

[3] We admit the number 0 to preserve the abbreviation $\forall r.C$ that stands for $(\leqslant 0\ r\ \neg C)$.

Note that we cannot obtain the same hardness results for executability of *atomic* services for the following reasons: (i) executability of atomic services in any DL $\mathcal{L}$ can trivially be reduced to ABox consequence in $\mathcal{L}$, and (ii) the complexity of ABox consequence is identical to the complexity of concept unsatisfiability in $\mathcal{ALCI}$ and $\mathcal{ALCFI}$.

For the proof of Theorem 20, let $\mathcal{L} \in \{\mathcal{ALCIO}, \mathcal{ALCFIO}\}$ and C an $\mathcal{L}$-concept whose (un)satisfiability is to be decided. For simplicity, we assume that C contains only a single nominal $\{n\}$. This can be done w.l.o.g. since the complexity of unsatisfiability in $\mathcal{ALCIO}$ (resp. $\mathcal{ALCFIO}$) is already ExpTime-hard (resp. co-NExpTime-hard) if only a single nominal is available and TBoxes are not admitted [1, 36, 37]. We reserve a concept name O and a role name u that do not occur in C. Let

$$\mathsf{rol}(C) := \{r, r^- \mid r \in \mathsf{N_R} \text{ used in } C\}$$

and let $C[O/\{n\}]$ denote the result of replacing the nominal $\{n\}$ in C with the concept name O. We define an ABox $\mathcal{A}$, an atomic service $S = (\emptyset, \emptyset, \mathsf{post}_S)$, and a concept D_C as follows:

$$\begin{array}{rcl} \mathcal{A}_C & := & \{a : (\neg O \sqcap \forall u.\neg O \sqcap \forall u. \bigsqcap_{r \in \mathsf{rol}(C)} \forall r.\exists u^-.\neg O)\} \\ \mathsf{post}_S & := & a : O \\ D_C & := & \exists u.C[O/\{n\}] \sqcap (\forall u. \bigsqcap_{r \in \mathsf{rol}(C)} \forall r.\forall u^-.O) \end{array}$$

The following lemma immediately yields Theorem 20.

Lemma 21. *The following statements are equivalent:*

1. *C is satisfiable.*
2. *there are interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}_C$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : D_C$.*
3. *$a : \neg D_C$ is not a consequence of applying S in $\mathcal{A}_C$.*
4. *the composite service $S, (\{a : \neg D_C\}, \emptyset, \emptyset)$ is not executable in $\mathcal{A}_C$.*

Proof. We only prove that (1) and (2) are equivalent since the other equivalences are immediate consequences of the definitions of prediction and executability.

(2) implies (1). Assume that there are interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}_C$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : D_C$. By the first conjunct of (the concept in the only assertion of) $\mathcal{A}_C$, by post_S, and Lemma 9, we have that $\mathcal{I}$ is identical to $\mathcal{I}'$ with the only exception that $a^{\mathcal{I}} = a^{\mathcal{I}'} \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$. For simplicity, we will call this relationship *quasi-identity* of $\mathcal{I}$ and $\mathcal{I}'$ in what follows. By the first conjunct of D_C, there is an $x_0 \in \Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ such that $(a^{\mathcal{I}'}, x_0) = (a^{\mathcal{I}}, x_0) \in u^{\mathcal{I}'} = u^{\mathcal{I}}$, and $x_0 \in C[O/\{n\}]^{\mathcal{I}'}$. We first identify the "relevant part" of $\mathcal{I}$ and $\mathcal{I}'$: set

$$\begin{array}{rcl} \mathsf{rel}_0 & := & \{x_0\} \\ \mathsf{rel}_{i+1} & := & \mathsf{rel}_i \cup \{x \in \Delta^{\mathcal{I}} \mid (y, x) \in r^{\mathcal{I}} \text{ for some } y \in \mathsf{rel}_i \text{ and } r \in \mathsf{rol}(C)\} \\ \mathsf{rel} & := & \bigcup_{i \geq 0} \mathsf{rel}_i \end{array}$$

The relevant part of $\mathcal{I}'$ can be defined analogously. Due to the quasi-identity of $\mathcal{I}$ and $\mathcal{I}'$, it is identical to the relevant part of $\mathcal{I}$. We now show the following:

Claim 1. For all $x \in \mathsf{rel}$, we have $(a^{\mathcal{I}}, x) = (a^{\mathcal{I}'}, x) \in u^{\mathcal{I}} = u^{\mathcal{I}'}$.

The proof of the claim is by induction on the smallest i such that $x \in \mathsf{rel}_i$. First let $i = 0$. Then $x = x_0$ and the claim holds since $(a^{\mathcal{I}}, x_0) \in u^{\mathcal{I}}$ by choice of x_0. Now let $i > 0$. Since i is smallest with $x \in \mathsf{rel}_i$, there is a $y \in \mathsf{rel}_{i-1}$ such that $(y, x) \in r^{\mathcal{I}} = r^{\mathcal{I}'}$ for some $r \in \mathsf{rol}(C)$. By induction, we have $(a^{\mathcal{I}}, y) \in u^{\mathcal{I}} = u^{\mathcal{I}'}$. Thus, the third conjunct of $\mathcal{A}_C$ implies that $y \in (\forall r.\exists u^-.\neg O)^{\mathcal{I}}$, and thus $x \in (\exists u^-.\neg O)^{\mathcal{I}}$. Let z be the witness for this, i.e. $(z, x) \in u^{\mathcal{I}} = u^{\mathcal{I}'}$ and $z \notin O^{\mathcal{I}}$. Since $(a^{\mathcal{I}}, y) \in u^{\mathcal{I}'}$, $(y, x) \in r^{\mathcal{I}'}$, and $(z, x) \in u^{\mathcal{I}'}$, we have $z \in O^{\mathcal{I}'}$ by the second conjunct of D_C. Since the only difference between $\mathcal{I}$ and $\mathcal{I}'$ is $a^{\mathcal{I}} \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$, $z \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$ implies $z = a^{\mathcal{I}}$. Since $(z, x) \in u^{\mathcal{I}}$, we are done.

This finishes the proof of Claim 1. Now we show that the concept name O may serve as a nominal on the relevant part of $\mathcal{I}'$ extended with $a^{\mathcal{I}'}$.

Claim 2. $O^{\mathcal{I}'} \cap (\mathsf{rel} \cup \{a^{\mathcal{I}'}\})$ is a singleton.

Proof: By post_S, we have $a^{\mathcal{I}'} \in O^{\mathcal{I}'}$. Thus, $O^{\mathcal{I}'}$ is non-empty. Now let $x \in O^{\mathcal{I}'} \cap \mathsf{rel}$. By Claim 1, we have $(a^{\mathcal{I}'}, x) \in u^{\mathcal{I}'}$. Then, $(a^{\mathcal{I}}, x) \in u^{\mathcal{I}}$. Thus, the second conjunct of $\mathcal{A}_C$ yields $x \notin O^{\mathcal{I}}$. Since the only difference between $\mathcal{I}$ and $\mathcal{I}'$ is $a^{\mathcal{I}'} \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$, this implies that $x = a^{\mathcal{I}'}$. Hence, $O^{\mathcal{I}'} \cap (\mathsf{rel} \cup \{a^{\mathcal{I}'}\})$ is a singleton.

Now define an interpretation $\mathcal{J}$ as $\mathcal{I}'$ extended with $n^{\mathcal{J}} := x$ if $O^{\mathcal{I}'} \cap (\mathsf{rel} \cup \{a^{\mathcal{I}'}\}) = \{x\}$ (such an x exists and is unique by Claim 2). It is standard to prove the following claim by structural induction. The only interesting aspects are the case of the nominal $\{n\}$ where we use that $\{n\}^{\mathcal{J}} = O^{\mathcal{I}'}$, and the fact that all domain elements encountered during the proof are from rel.

Claim 3. For all $x \in \mathsf{rel}$ and all subconcepts D of C, we have $x \in D[O/\{n\}]^{\mathcal{J}}$ iff $x \in D^{\mathcal{J}}$.

Finally, $x_0 \in C[O/\{n\}]^{\mathcal{I}'}$ yields $x_0 \in C^{\mathcal{J}}$ by Claim 3, and thus C is satisfiable.

"only if". Let $\mathcal{J}$ be a model of C, and let $x_0 \in C^{\mathcal{J}}$. Let $\mathcal{I}$ be the interpretation that is identical to $\mathcal{J}$, but for the following modifications:

- $O^{\mathcal{I}} = \emptyset$;
- $a^{\mathcal{I}} = n^{\mathcal{J}}$;
- $u^{\mathcal{I}} = \{(a^{\mathcal{I}}, x) \mid x \in \Delta^{\mathcal{I}}\}$.

Moreover, let $\mathcal{I}'$ be the interpretation that is identical to $\mathcal{I}$ except that $O^{\mathcal{I}'} = \{n\}^{\mathcal{J}}$. It is readily checked that $\mathcal{I} \models \mathcal{A}_C$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : D_C$. ❑

4 Syntactic Restrictions

The purpose of this section is to provide a justification for the syntactic restrictions that we have adopted in our formalism for describing services:

1. we do not allow for transitive roles, which are available in OWL-DL;

2. we only allow for acyclic TBoxes rather than arbitrary (also cyclic) ones or even so-called general concept inclusions (GCIs), which are also available in OWL-DL;

3. in post-conditions $\varphi/C(a)$, we require C to be a primitive concept or its negation, rather than admitting arbitrary concepts.

We will show that removing the first restriction leads to *semantic problems*, while removing the second and third restriction leads to *both semantic and computational problems*.

4.1 Transitive Roles

Transitive roles are offered by most modern DL systems [10, 8], and also by the ontology languages OWL, DAML+OIL, and OIL [12, 11, 7]. They are added to $\mathcal{ALCQIO}$ by reserving a subset of roles $\mathsf{N_{tR}}$ of $\mathsf{N_R}$ such that all $r \in \mathsf{N_{tR}}$ are interpreted as transitive relations $r^{\mathcal{I}}$ in all models $\mathcal{I}$. We show that admitting the use of transitive roles in post-conditions yields semantic problems.

By Lemma 9, services without occlusions $S = (\mathsf{pre}, \emptyset, \mathsf{post})$ are deterministic in the sense that $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$, and $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}''$ implies $\mathcal{I}' = \mathcal{I}''$. This is not any more the case for services referring to transitive roles: consider the service $S = (\emptyset, \emptyset, \{\mathsf{has\text{-}part}(\mathsf{car}, \mathsf{engine})\})$ that adds an engine to a car. Let $\mathsf{has\text{-}part}$ be a transitive role and take the model

$$\begin{aligned} \Delta^{\mathcal{I}} &:= \{\mathsf{car}, \mathsf{engine}, \mathsf{valve}\} \\ \mathsf{has\text{-}part}^{\mathcal{I}} &:= \{(\mathsf{engine}, \mathsf{valve})\} \\ z^{\mathcal{I}} &:= z \text{ for } z \in \Delta^{\mathcal{I}}. \end{aligned}$$

Then we have both $\mathcal{I} \Rightarrow_S \mathcal{I}'$ and $\mathcal{I} \Rightarrow_S \mathcal{I}''$, where $\mathcal{I}'$ is obtained from $\mathcal{I}$ by setting

$$\mathsf{has\text{-}part}^{\mathcal{I}'} := \{(\mathsf{car}, \mathsf{engine}), (\mathsf{engine}, \mathsf{valve}), (\mathsf{car}, \mathsf{valve})\}$$

and $\mathcal{I}''$ is obtained from $\mathcal{I}$ by setting

$$\mathsf{has\text{-}part}^{\mathcal{I}''} := \{(\mathsf{car}, \mathsf{engine})\}.$$

Observe that, in $\mathcal{I}''$, the valve is no longer part of the engine since adding only $(\mathsf{car}, \mathsf{engine})$ to $\mathsf{has\text{-}part}^{\mathcal{I}}$ violates the transitivity of $\mathsf{has\text{-}part}^{\mathcal{I}}$.

In the area of reasoning about actions, it is well-known that non-determinism of this kind requires extra effort to obtain sensible consequences of action/service executions [19, 35]. In the above example, it is unlikely that both outcomes of the service application are equally desirable. Thus, we need a mechanism for eliminating unwanted outcomes or preferring the desired ones. We leave such extensions as future work.

4.2 Cyclic TBoxes and GCIs

Assume that we admit arbitrary (also cyclic) TBoxes as defined in Section 2.1. Then semantic problems arise due to a crucial difference between cyclic and acyclic TBoxes:

for acyclic TBoxes, the interpretation of primitive concepts *uniquely* determines the extension of the defined ones, while this is not the case for cyclic ones. Together with the fact that the preference relation between interpretations $\preceq_{\mathcal{I},S,\mathcal{T}}$ only takes into account primitive concepts, this means that the minimization of changes induced by service application does not work as expected. To see this, consider the following example:

$$\begin{aligned} \mathcal{A} &:= \{\mathsf{Dog}(a)\} \\ \mathcal{T} &:= \{\mathsf{Dog} \equiv \exists \mathsf{parent}.\mathsf{Dog}\} \\ \mathsf{post} &:= \{\mathsf{Cat}(b)\} \end{aligned}$$

Then, $\mathsf{Dog}(a)$ is *not* a consequence of applying $S = (\emptyset, \emptyset, \mathsf{post})$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$, as one would intuitively expect. This is due to the following countermodel. Define an interpretation $\mathcal{I}$ as follows:

$$\begin{aligned} \Delta^{\mathcal{I}} &:= \{b\} \cup \{d_0, d_1, d_2, \ldots\} \\ \mathsf{Dog}^{\mathcal{I}} &:= \{d_0, d_1, d_2, \ldots\} \\ \mathsf{Cat}^{\mathcal{I}} &:= \emptyset \\ \mathsf{parent}^{\mathcal{I}} &:= \{(d_i, d_{i+1}) \mid i \in \mathbb{N}\} \\ a^{\mathcal{I}} &:= d_0 \\ b^{\mathcal{I}} &:= b \end{aligned}$$

The interpretation $\mathcal{I}'$ is defined as $\mathcal{I}$, with the exception that $\mathsf{Cat}^{\mathcal{I}'} = \{b\}$ and $\mathsf{Dog}^{\mathcal{I}'} := \emptyset$. Using the fact that Dog is a defined concept and thus not considered in the definition of $\preceq_{\mathcal{I},S,\mathcal{T}}$, it is easy to see that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S^{\mathcal{T}} \mathcal{I}'$, and $\mathcal{I}' \not\models \mathsf{Dog}(a)$.

There appear to be two possible ways to solve this problem: either include defined concepts in the minimization of changes, i.e., treat them in the definition of $\preceq_{\mathcal{I},S,\mathcal{T}}$ in the same way as primitive concepts, or use a semantics that regains the "definitorial power" of acyclic TBoxes, namely that an interpretation of the primitive concepts *uniquely* determines the interpretation of defined concepts. The first option is infeasible since minimizing a defined concept A with TBox definition $A \equiv C$ corresponds to minimizing the complex concept C, and it is well-known that even the minimization of arbitrary Boolean concepts (in particular of disjunctions) induces technical problems and counterintuitive results [18]. The second option seems more feasible: if we adopt the least or greatest fixpoint semantics for TBoxes as first proposed by Nebel [23], it is indeed the case that primitive concepts uniquely determine defined concepts. Thus, it may be interesting to analyze services with cyclic TBoxes under fixpoint semantics as future work.

Even more general than admitting cyclic TBoxes is to allow general concept inclusions (GCIs). A *GCI* is an expression $C \sqsubseteq D$, with C and D (possibly complex) concepts. An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. As we can rewrite a concept equation $A \equiv C$ as two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$, it should be obvious that (sets of) GCIs strictly generalize (also cyclic) TBoxes. When admitting GCIs in connection with services, we thus run into the same problems as with cyclic TBoxes. However, the problems are even more serious in the case of GCIs: first, GCIs do not allow an obvious

partitioning of concept names into primitive and defined ones. Thus, in the definition of $\preceq_{\mathcal{I},S,\mathcal{T}}$, the only choice is to minimize *all* concept names, which corresponds to the problematic minimization of complex concepts mentioned above. Second, the vanished distinction between primitive and defined concepts means that we can no longer restrict concepts C in post-conditions $\varphi/C(a)$ to literals over *primitive* concept names. The best we can do is to restrict such concepts to literals over arbitrary concept names. However, together with the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$ with C a complex concept, the literal post-condition $\varphi/A(a)$ is equivalent to the complex one $\varphi/C(a)$. Thus, it seems that GCIs cannot be admitted without simultaneously admitting arbitrarily complex concepts in post-conditions. As we will discuss in the following section, this step induces additional semantic problems as well as computational problems.

4.3 Complex Concepts in Post-Conditions

Let a *generalized* service be a service where post-conditions are of the form φ/ψ for arbitrary assertions φ and ψ. In other words, ψ is no longer restricted to be a literal over primitive concepts. For simplicity, further assume that occlusions are disallowed and that neither TBoxes nor GCIs are admitted. As we shall discuss in the following, there are both semantic and computational problems with generalized services: firstly, they offer an expressivity that is difficult to control and often yields unexpected consequences. Secondly, reasoning with generalized services easily becomes undecidable.

Semantic Problems

Clearly, generalized services such as the trivial $S = (\emptyset, \emptyset, \{a : A \sqcup B\})$ are not deterministic and thus introduce similar complications as discussed for transitive roles in Section 4.1. However, disjunction is not the only constructor to introduce non-determinism when allowed in post-conditions. An even "higher degree" of non-determinsm is introduced by existential and universal value restrictions:

- If a post-condition contains $a : \exists r.A$ and this assertion was not already satisfied before the execution of the service, then the non-determinism lies in the choice of a witness object, i.e., *any* domain element $x \in \Delta^{\mathcal{I}}$ may be chosen to satisfy $(a^{\mathcal{I}}, x) \in r^{\mathcal{I}}$ and $x \in A^{\mathcal{I}}$ after execution of the service. Note that some such x may already satisfy the former condition, some may satisfy the latter, and some neither.

  The fact that *any* domain element is a potential witness object implies that, e.g., $\mathsf{mary} : \mathsf{Female}$ is not a consequence of applying the service

  $$(\emptyset, \emptyset, \{\mathsf{mary} : \exists \mathsf{has\text{-}child}.\neg\mathsf{Female}\})$$

  in the ABox $\{\mathsf{mary} : \mathsf{Female}\}$—an effect that may not be intended.

- If a post-condition contains $a : \forall r.A$ and this assertion was not already satisfied before the execution of the service, we also have a non-deterministic situation: for each object $x \in \Delta^{\mathcal{I}}$ such that $(a^{\mathcal{I}}, x) \in r^{\mathcal{I}}$ and $x \notin A^{\mathcal{I}}$ holds before the execution

of the service, we have to decide whether $(a^{\mathcal{J}}, x) \notin r^{\mathcal{J}}$ or $x \in A^{\mathcal{J}}$ should be satisfied after execution of the service.[4]

Similarly to the existential case, we may obtain surprising results due to the fact that *any* domain element $x \in \Delta^{\mathcal{I}}$ may satisfy $(a^{\mathcal{I}}, x) \in r^{\mathcal{I}}$ and $x \in A^{\mathcal{I}}$ unless explicitly stated otherwise. This means that, e.g., $\mathsf{tire2}{:}\neg\mathsf{Filled}$ is not a consequence of applying the service

$$(\emptyset, \emptyset, \{\mathsf{car1}{:}\forall \mathsf{tire}.\mathsf{Filled}\})$$

in the ABox $\{\mathsf{tire}(\mathsf{car2}, \mathsf{tire2}),\ \mathsf{tire2}{:}\neg\mathsf{Filled}\}$.

Complex concepts with many nested operators may obviously introduce a rather high degree of non-determinism. While simple non-determinism such as the one introduced by transitive roles or post-conditions $a : C \sqcup D$ may be dealt with in a satisfactory way [19, 35], none of the mainstream action formalisms allows arbitrary formulas in post-conditions to avoid having to deal with the resulting massive degree of non-determinism. Indeed, most formalisms such as the basic situation calculus restrict themselves to literals in post-conditions [28, 34]—just as our non-generalized services do.

Computational Problems

Executability and projection for generalized services easily becomes undecidable. To illustrate this, we prove undecidability of these reasoning tasks for the DL $\mathcal{ALCFI}$ that has been introduced in Section 3.3. Recall that $\mathcal{ALCFI}$ is obtained from $\mathcal{ALCQI}$ by limiting numbers occurring in number restrictions to $\{0, 1\}$. This result should be contrasted with the fact that, by Theorem 13, reasoning with non-generalized services is decidable even for powerful extensions of $\mathcal{ALCFI}$. We leave it as an open problem whether the presented undecidability result can be strengthened to simpler description logics, in particular $\mathcal{ALC}$.

Theorem 22. *There exists a generalized atomic service S and an ABox $\mathcal{A}$ formulated in $\mathcal{ALCFI}$ such that the following problems are undecidable: given a concept C,*

- *decide whether the assertion $a : C$ is a consequence of applying S in $\mathcal{A}$;*
- *decide whether the composite service S, S' is executable in $\mathcal{A}$, where $S' = (\{a : C\}, \emptyset, \emptyset)$.*

The proof of Theorem 22 is by reduction of the domino problem to non-consequence and non-executability.

Definition 23. Let $\mathcal{D} = (T, H, V)$ be a *domino system*, where T is a finite set of *tile types* and $H, V \subseteq T \times T$ represent the horizontal and vertical matching conditions. We say that $\mathcal{D}$ *tiles the plane* iff there exists a mapping $\tau : \mathbb{Z} \times \mathbb{Z} \to T$ such that, for all $(x, y) \in \mathbb{Z} \times \mathbb{Z}$, we have

- if $\tau(x, y) = t$ and $\tau(x + 1, y) = t'$, then $(t, t') \in H$

[4] There may even be cases where it is intended that both conditions are satisfied after service execution; this is, however, not justified by the PMA semantics of generalized services.

$$
\begin{aligned}
\mathcal{A} = \{ \quad & a : \neg A && (1)\\
& a : \forall u.\Big(\bigsqcap_{r \in \{x,y,u,x^-,y^-,u^-\}} \forall r.\neg A\Big) && (2)\\
& a : \forall u.\neg B \quad \} && (3)\\
\mathsf{post} = \{ \quad & a : \forall u.A && (4)\\
& a : \forall u.((\forall x^-.\forall y^-.\neg Q) \sqcup B) \quad \} \text{ with } Q := \forall x.\forall y.B \to \exists y.\exists x.B && (5)\\
C_{\mathcal{D}} = \quad & A \sqcap && (6)\\
& \forall u.\Big(\bigsqcap_{r \in \{x,y,u,x^-,y^-,u^-\}} \forall r.A\Big) \sqcap && (7)\\
& \forall u.B \sqcap && (8)\\
& \forall u.(\exists x.\top \sqcap \exists y.\top \sqcap \exists x^-.\top \sqcap \exists y^-.\top) \sqcap && (9)\\
& \forall u.((\leq 1\ x) \sqcap (\leq 1\ y) \sqcap (\leq 1\ x^-) \sqcap (\leq 1\ y^-)) \sqcap && (10)\\
& \forall u.\Big(\bigsqcap_{\substack{t,t' \in T \\ \text{with } t \neq t'}} \neg(D_t \sqcap D_{t'})\Big) \sqcap && (11)\\
& \forall u.\Big(\bigsqcup_{(t,t') \in H} (D_t \sqcap \forall x.D_{t'})\Big) \sqcap && (12)\\
& \forall u.\Big(\bigsqcup_{(t,t') \in V} (D_t \sqcap \forall y.D_{t'}) && (13)
\end{aligned}
$$

Figure 2: The ABox $\mathcal{A}$, the post-conditions of S, and the concept $C_{\mathcal{D}}$.

- if $\tau(x, y) = t$ and $\tau(x, y + 1) = t'$, then $(t, t') \in V$

Such a mapping τ is called a *solution* for $\mathcal{D}$. △

For a domino system $\mathcal{D} = (T, H, V)$, the ABox $\mathcal{A}$, the service $S = (\emptyset, \emptyset, \mathsf{post})$ and the concept $C_{\mathcal{D}}$ are defined in Figure 2, where A, B, B', C, and C' are concept names, D_t is a concept name for each $t \in T$, and x, y, and u are role names. For a better readibility, we write $(\leqslant 1\ r)$ instead of $(\leqslant 1\ r\ \top)$. Our aim is to prove the following lemma, which immediately yields Theorem 22.

Lemma 24. *The following statements are equivalent:*

1. *The domino system $\mathcal{D}$ has a solution.*
2. *There are interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$.*
3. *$a : \neg C_{\mathcal{D}}$ is not a consequence of applying S in $\mathcal{A}$.*
4. *the composite service $S, (\{a : \neg C_{\mathcal{D}}\}, \emptyset, \emptyset)$ is not executable in $\mathcal{A}$.*

We only prove the equivalence of (1) and (2) since the other equivalences hold by definition of projection and executability. It is convenient to first establish a series of lemmas.

Lemma 25. *Let $\mathcal{I}$, $\mathcal{I}'$ be interpretations such that $\mathcal{I} \Rightarrow_S \mathcal{I}'$. Then, for all $r \in \{x, y, u, x^-, y^-, u^-\}$, we have that $r^{\mathcal{I}'} \subseteq r^{\mathcal{I}}$;*

Proof. We concentrate on $r = x$ as the cases $r = y$ and $r = u$ are analogous and the cases for inverse roles follow immediately by definition of the semantics. Thus let $(d, d') \in x^{\mathcal{I}'}$ and suppose that $(d, d') \notin x^{\mathcal{I}}$, to the contrary of what is to be shown. Let $\mathcal{J}$ be the interpretation that is identical to $\mathcal{I}'$ except that $(d, d') \notin x^{\mathcal{J}}$. Clearly, $\mathcal{J} \neq \mathcal{I}$ and $\mathcal{J} \preceq_{\mathcal{I}} \mathcal{I}'$. We show that $\mathcal{J}$ satisfies all post-conditions, thus contradicting $\mathcal{I} \Rightarrow_S \mathcal{I}'$. Since only the interpretation of x has changed and Line (4) does not mention x, we only need to consider Line (5): it is satisfied by $\mathcal{J}$ since it is satisfied by $\mathcal{I}'$ and it is easily seen that this line cannot be invalidated by shrinking x (only by growing it). ❑

Let $\mathcal{I}$ be an interpretation and $d, d' \in \Delta^{\mathcal{I}}$. Then

- d' is *reachable* from d if there exists a sequence of elements $d_1, \ldots, d_k \in \Delta^{\mathcal{I}}$ such that $d_1 = d$, $d_k = d'$, and $(d_i, d_{i+1}) \in x^{\mathcal{I}} \cup y^{\mathcal{I}} \cup u^{\mathcal{I}} \cup (x^-)^{\mathcal{I}} \cup (y^-)^{\mathcal{I}} \cup (u^-)^{\mathcal{I}}$ for $1 \leq i < k$; such as sequence is called a *path* from d to d'; as we may have $k = 1$, every $d \in \Delta^{\mathcal{I}}$ is reachable from itself;
- we use $\delta^{\mathcal{I}}(d, d')$ to denote the length of the shortest path from d to d'; if d' is not reachable from d, then $\delta^{\mathcal{I}}(d, d')$ is undefined.

Lemma 26. *Let $\mathcal{I}$, $\mathcal{I}'$ be interpretations such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$. Then we have $(a^{\mathcal{I}'}, d) \in u^{\mathcal{I}} \cap u^{\mathcal{I}'}$ if d is reachable from $a^{\mathcal{I}'}$.*

Proof. The proof is by induction on $\delta^{\mathcal{I}'}(a^{\mathcal{I}'}, d)$ which we abbreviate by $\delta(d)$ for convenience. Note that $\delta(d)$ is defined for all d that are reachable from $a^{\mathcal{I}'}$.

First for the induction start, i.e. $\delta(d) = 0$. Then $d = a^{\mathcal{I}'}$. By Line (1) of Figure 2,[5] we have $a^{\mathcal{I}'} \notin A^{\mathcal{I}}$. By Line (6), $a^{\mathcal{I}'} \in A^{\mathcal{I}'}$. We first show that $(a^{\mathcal{I}'}, a^{\mathcal{I}'}) \in u^{\mathcal{I}'}$. Assume to the contrary that $(a^{\mathcal{I}'}, a^{\mathcal{I}'}) \notin u^{\mathcal{I}'}$ holds. Then let $\mathcal{J}$ be the interpretation that is defined as $\mathcal{I}'$ except that $a^{\mathcal{I}'} \notin A^{\mathcal{J}}$. Clearly, $\mathcal{J} \neq \mathcal{I}'$ and $\mathcal{J} \preceq_{\mathcal{I}} \mathcal{I}'$. Moreover, it is easily verified that $\mathcal{J}$ satisfies all post-conditions (only Line (4) needs to be considered): contradiction to the fact that $\mathcal{I} \Rightarrow_S \mathcal{I}'$. Finally, $(a^{\mathcal{I}'}, a^{\mathcal{I}'}) \in u^{\mathcal{I}}$ is an immediate consequence of Lemma 25.

Now for the induction step, i.e. $\delta(d) > 0$. Then there is a $d' \in \Delta^{\mathcal{I}'}$ such that d' is reachable from $a^{\mathcal{I}'}$, $\delta(d') = \delta(d) - 1$, and

$$(d, d') \in x^{\mathcal{I}'} \cup y^{\mathcal{I}'} \cup u^{\mathcal{I}'} \cup (x^-)^{\mathcal{I}'} \cup (y^-)^{\mathcal{I}'} \cup (u^-)^{\mathcal{I}'}. \qquad (*)$$

By induction, we have $(a^{\mathcal{I}'}, d') \in u^{\mathcal{I}} \cap u^{\mathcal{I}'}$. This together with $(*)$ and Line (7) yields that $d \in A^{\mathcal{I}'}$. Moreover, $d \notin A^{\mathcal{I}}$: With $(*)$, Lemma 25 yields

$$(d, d') \in x^{\mathcal{I}} \cup y^{\mathcal{I}} \cup u^{\mathcal{I}} \cup (x^-)^{\mathcal{I}} \cup (y^-)^{\mathcal{I}} \cup (u^-)^{\mathcal{I}'}$$

[5] In the remainder of this proof, we use Line (j) as an abbreviation for Line (j) of Figure 2.

and thus $(a^{\mathcal{I}'}, d') \in u^{\mathcal{I}}$ together with Line (2) yields $d \notin A^{\mathcal{I}}$. We can now continue exactly as in the induction start to show first that $(a^{\mathcal{I}'}, d) \in u^{\mathcal{I}'}$, and then conclude that $(a^{\mathcal{I}'}, d) \in u^{\mathcal{I}}$. ❑

Let $\mathcal{I}$ be an interpretation. For $a \in \mathsf{N_I}$, we define

$$\mathsf{reach}_{\mathcal{I}}(a) := \{d \in \Delta^{\mathcal{I}} \mid d \text{ is reachable from } a^{\mathcal{I}} \text{ in } \mathcal{I}\}.$$

For two interpretations $\mathcal{I}$ and $\mathcal{I}'$ and $a \in \mathsf{N_I}$, we say that $\mathcal{I}$ and $\mathcal{I}'$ *agree on the a-rooted part* if the following conditions are satisfied:

- $a^{\mathcal{I}} = a^{\mathcal{I}'}$;
- $\mathsf{reach}_{\mathcal{I}}(a) = \mathsf{reach}_{\mathcal{I}'}(a)$;
- $d \in A^{\mathcal{I}}$ iff $d \in A^{\mathcal{I}'}$ for all $d \in \mathsf{reach}_{\mathcal{I}}(a)$ and concept names A;
- $(d, e) \in r^{\mathcal{I}}$ iff $(d, e) \in r^{\mathcal{I}'}$ for all $d, e \in \mathsf{reach}_{\mathcal{I}}(a)$ and $r \in \{x, y, u\}$.

The following lemma can be proved by structural induction in a straightforward way. Details are left to the reader.

Lemma 27. *Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations that agree on the a-rooted part for some $a \in \mathsf{N_I}$. Then $d \in C^{\mathcal{I}}$ iff $d \in C^{\mathcal{I}'}$ for all $d \in \mathsf{reach}_{\mathcal{I}}(a)$ and $\mathcal{ALCFI}$ concepts C using only the roles x, y, u, and their inverses.*

We say that $\mathcal{I}$ is *a-rooted* if every $d \in \Delta^{\mathcal{I}}$ is reachable from $a^{\mathcal{I}}$.

Lemma 28. *If there exist interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$, then there exist a-rooted interpretations $\mathcal{J}, \mathcal{J}'$ such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{J} \Rightarrow_S \mathcal{J}'$, and $\mathcal{J}' \models a : C_{\mathcal{D}}$.*

Proof. Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations such that $\mathcal{I} \Rightarrow_S \mathcal{I}'$ and $\mathcal{I}' \models a : C_{\mathcal{D}}$. Let $\mathcal{J}$ be the restriction of $\mathcal{I}$ to $\mathsf{reach}_{\mathcal{I}}(a)$, i.e.

- $\Delta^{\mathcal{J}} := \mathsf{reach}_{\mathcal{I}}(a)$;
- $A^{\mathcal{J}} := A^{\mathcal{I}} \cap \mathsf{reach}_{\mathcal{I}}(a)$; for all concept names A;
- $r^{\mathcal{J}} := r^{\mathcal{I}} \cap (\mathsf{reach}_{\mathcal{I}}(a) \times \mathsf{reach}_{\mathcal{I}}(a))$ for all role names r;
- $b^{\mathcal{J}} := b^{\mathcal{I}}$ if $b^{\mathcal{I}} \in \mathsf{reach}_{\mathcal{I}}(a)$, and $b^{\mathcal{J}} := a^{\mathcal{I}}$ otherwise, for all individual names b.

Similarly, let $\mathcal{J}'$ be the restriction of $\mathcal{I}'$ to $\mathsf{reach}_{\mathcal{I}}(a)$. Then, $\mathcal{I}$ and $\mathcal{J}$ agree on the a-rooted part, and so do $\mathcal{I}'$ and $\mathcal{J}'$. It is thus not hard to show that $\mathcal{J} \models \mathcal{A}$, $\mathcal{J} \Rightarrow_S \mathcal{J}'$, and $\mathcal{J}' \models a : C_{\mathcal{D}}$ as required:

- $\mathcal{J} \models \mathcal{A}$ and $\mathcal{J}' \models a : C_{\mathcal{D}}$ is an immediate consequence of Lemma 27.
- To show $\mathcal{J} \Rightarrow_S \mathcal{J}'$, we have to prove that $\mathcal{J}' \models \mathsf{post}$ and there is no $\mathcal{J}''$ such that $\mathcal{J}' \neq \mathcal{J}''$, $\mathcal{J}'' \preceq_{\mathcal{J}} \mathcal{J}'$, and $\mathcal{J}'' \models \mathsf{post}$. Since the former is again an immediate consequence of Lemma 27, we concentrate on the latter.

  Suppose to the contrary of what is to be shown that there is a $\mathcal{J}''$ such that $\mathcal{J}' \neq \mathcal{J}''$, $\mathcal{J}'' \preceq_{\mathcal{J}} \mathcal{J}'$, and $\mathcal{J}'' \models \mathsf{post}$. Define an interpretation $\mathcal{I}''$ by setting

- $\Delta^{\mathcal{I}''} := \Delta^{\mathcal{I}'}$;
- $A^{\mathcal{I}''} := A^{\mathcal{J}''} \cup (A^{\mathcal{I}'} \setminus \mathsf{reach}_{\mathcal{I}}(a))$; for all concept names A;
- $r^{\mathcal{I}''} := r^{\mathcal{J}''} \cup (r^{\mathcal{I}'} \setminus (\mathsf{reach}_{\mathcal{I}}(a) \times \mathsf{reach}_{\mathcal{I}}(a)))$ for all role names r;
- $b^{\mathcal{I}''} := b^{\mathcal{I}'}$ for all individual names b.

It is not hard to verify that $\mathcal{I}'' \neq \mathcal{I}'$, $\mathcal{I}'' \preceq_{\mathcal{I}} \mathcal{I}'$, and $\mathcal{I}'' \models \mathsf{post}$. Thus, we have established a contradiction to $\mathcal{I} \Rightarrow_S \mathcal{I}'$.

❑

A *frame* is a pair $(\Delta^{\mathcal{F}}, \cdot^{\mathcal{F}})$, where $\Delta^{\mathcal{F}}$ is a non-empty set and $\cdot^{\mathcal{F}}$ maps each role name r to a binary relation $r^{\mathcal{F}} \subseteq \Delta^{\mathcal{F}} \times \Delta^{\mathcal{F}}$. An interpretation $\mathcal{I}$ is said to be *based on* $\mathcal{F}$ if $\Delta^{\mathcal{F}} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{F}}$ and $\cdot^{\mathcal{I}}$ agree on the interpretation of all role names. A frame $\mathcal{F}$ *validates* a concept C if we have $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$, for all interpretations $\mathcal{I}$ based on $\mathcal{F}$.

Lemma 29. *Let $\mathcal{I}$ and $\mathcal{I}'$ be a-rooted interpretations such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$. Then $\mathcal{I}'$ is based on a frame $\mathcal{F}$ that satisfies the following:*

1. *for each $d \in \Delta^{\mathcal{F}}$, there is a unique d' with $(d, d') \in x^{\mathcal{F}}$ and a unique d'' with $(d, d'') \in (x^-)^{\mathcal{F}}$;*
2. *for each $d \in \Delta^{\mathcal{F}}$, there is a unique d' with $(d, d') \in y^{\mathcal{F}}$ and a unique d'' with $(d, d'') \in (y^-)^{\mathcal{F}}$;*
3. *$\mathcal{F}$ validates the following concepts:*
   - $\forall x.\forall y.B \rightarrow \exists y.\exists x.B$;
   - $\forall y.\forall x.B \rightarrow \exists x.\exists y.B$.

Proof. Points 1 and 2 are an immediate consequence of a-rootedness of $\mathcal{I}'$, Lemma 26, and Lines (9) and (10). Concerning Point 3, it suffices to prove validity of the first listed concept: the second concept can be obtained from the first one by contraposition and replacing B with $\neg B$. Thus, validity of the first concept implies validity of the second one. For convenience, we abbreviate the first listed concept with Q as in Figure 2.

Let $\mathcal{I}$ and $\mathcal{I}'$ be a-rooted interpretations such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$, and let $\mathcal{F}$ be the frame that $\mathcal{I}'$ is based upon. Assume that $\mathcal{F}$ does not validate Q, to the contrary of what is to be shown. Hence there is an interpretation $\mathcal{I}^*$ based on $\mathcal{F}$ such that $Q^{\mathcal{I}^*} \neq \Delta^{\mathcal{I}^*}$ $(= \Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}})$. Fix a $d_0 \in (\neg Q)^{\mathcal{I}^*}$. Let d_0' be the unique element such that $(d_0, d) \in y^{\mathcal{I}'}$ and $(d, d_0') \in x^{\mathcal{I}'}$ for some element d—such a d_0' exists (and is unique) by Points 1 and 2. Then define a new interpretation $\mathcal{J}$ that is defined as $\mathcal{I}'$, but with the following difference:

$$B^{\mathcal{J}} = \Delta^{\mathcal{I}^*} \setminus \{d_0'\}.$$

By Lemma 26, a-rootedness of $\mathcal{I}'$, and Line (8), we have $B^{\mathcal{I}'} = \Delta^{\mathcal{I}'}$. Thus $B^{\mathcal{J}} \subsetneq B^{\mathcal{I}'}$. By Lemma 26, a-rootedness of $\mathcal{I}$, and Line (3) we have $B^{\mathcal{I}} = \emptyset$. Thus, $\mathcal{J} \preceq_{\mathcal{I}} \mathcal{I}'$.

Moreover, $\mathcal{J}$ satisfies all the post-conditions. This is trivial for Line (4) but less trivial for Line (5). Thus, let us show that every $d \in \Delta^{\mathcal{J}}$ is in the $\mathcal{J}$-extension of the concept

$$(\forall x^-.\forall y^-.\neg Q) \sqcup B \qquad (*)$$

Since we clearly have $d \in B^{\mathcal{J}}$ for all $d \in \Delta^{\mathcal{J}} \setminus \{d_0'\}$, it remains to deal with d_0'. This is done in what follows. Since $d_0 \in (\neg Q)^{\mathcal{I}^*}$, we have $d_0 \in (\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B)^{\mathcal{I}^*}$. Let d_0'' be the unique element such that $(d_0, d) \in x^{\mathcal{I}'}$ and $(d, d_0'') \in y^{\mathcal{I}'}$ for some element d. Due to the fact that $d_0 \in (\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B)^{\mathcal{I}^*}$, we have $d_0' \neq d_0''$. Thus, $d_0'' \in B^{\mathcal{J}}$ by definition of $\mathcal{J}$. Since we also have $d_0' \notin B^{\mathcal{J}}$, we get $d_0 \in (\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B)^{\mathcal{J}} = (\neg Q)^{\mathcal{J}}$. Thus, d_0' is in the $\mathcal{J}$-extension of the first disjunct of $(*)$.

We have shown that $\mathcal{J}$ indeed satisfies all post-conditions. Together with $\mathcal{J} \preceq_{\mathcal{I}} \mathcal{I}'$, we have a contradiction to $\mathcal{I} \Rightarrow_S \mathcal{I}'$. ❏

Definition 30. Let $\mathcal{F}$ be a frame and $d \in \Delta^{\mathcal{F}}$. We say that a frame $\mathcal{F}$ *contains a morphic image of* $\mathbb{Z} \times \mathbb{Z}$ *with origin* d if there exists a total function $\pi : \mathbb{Z} \times \mathbb{Z} \to \Delta^{\mathcal{F}}$ such that, for all i, j:

1. $(0, 0) \mapsto d$
2. $(\pi(i, j), \pi(i + 1, j)) \in x^{\mathcal{F}}$;
3. $(\pi(i, j), \pi(i, j + 1)) \in y^{\mathcal{F}}$.

△

Lemma 31. *Let $\mathcal{I}$ and $\mathcal{I}'$ be a-rooted interpretations such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$. Then $\mathcal{I}'$ is based on a frame $\mathcal{F}$ that contains a morphic image of $\mathbb{Z} \times \mathbb{Z}$ with origin $a^{\mathcal{I}'}$.*

Proof. We construct the desired function π in an incremental way:

- The first step is to define $\pi(i, j)$ for a "staircase" through $(0, 0)$, namely for each

  $$(i, j) \in \{\ldots, (-2, -1), (-1, -1), (-1, 0), (0, 0), (0, 1), (1, 1), (2, 1), \ldots\}.$$

  This is done by induction on $|i + j|$:

  start. Then $i = j = 0$. Set $\pi(i, j) = a^{\mathcal{I}'}$;

  step. Let $\pi(i, j)$ be already defined for all i, j with $|i + j| < n$ for some $n \in \mathbb{N}$. Intuitively, we extend the staircase in both directions. First for going up the staircase. Let $i, j \in \mathbb{N}$ such that $i + j = n$. We distinguish two cases:
  - $j > i$. By Point 2 of Lemma 29, there is a d such that $(\pi(i, j-1), d) \in y^{\mathcal{I}'}$. Set $\pi(i, j) = d$;
  - $j = i$. By Point 1 of Lemma 29, there is a d such that $(\pi(i-1, j), d) \in x^{\mathcal{I}'}$. Set $\pi(i, j) = d$.

  Now for going "down". Let $i, j \in \mathbb{Z}$ such that $i + j = -n$. Again, we distinguish two cases:

- $j > i$. By Point 2 of Lemma 29, there is a d such that $(d, \pi(i, j+1)) \in y^{\mathcal{I}'}$. Set $\pi(i, j) = d$;
- $j = i$. By Point 1 of Lemma 29, there is a d such that $(d, \pi(i+1, j)) \in x^{\mathcal{I}'}$. Set $\pi(i, j) = d$.

- Starting from the staircase, we can now exhaustively "fill up" the mapping π by repeating the following steps ad infinitum:

  Suppose that $\pi(i, j)$, $\pi(i, j+1)$, and $\pi(i+1, j+1)$ are already defined (i.e. one "positive stair" of the staircase). We show that there is a $d \in \Delta^{\mathcal{I}'}$ with $(\pi(i, j), d) \in x^{\mathcal{I}'}$ and $(d, \pi(i+1, j+1)) \in y^{\mathcal{I}'}$. Assume to the contrary that no such d exists. Then let $\mathcal{J}$ be an interpretation based on the same frame as $\mathcal{I}'$ with $B^{\mathcal{J}} = \{\pi(i+1, j+1)\}$. As $x^{\mathcal{J}}$ and $y^{\mathcal{J}}$ are functional by Points (1) and (2) of Lemma 29, we have $\pi(i, j) \in (\forall y.\forall x.B)^{\mathcal{J}}$. If there is no d with $(\pi(i, j), d) \in x^{\mathcal{I}'}$ and $(d, \pi(i+1, j+1)) \in y^{\mathcal{I}'}$, then $B^{\mathcal{J}} = \{\pi(i+1, j+1)\}$ implies that $\pi(i, j) \notin (\exists x.\exists y.B)^{\mathcal{J}}$, contradicting that $\mathcal{F}$ validates $\forall y.\forall x.B \to \exists x.\exists y.B$ as stated by Point 3 of Lemma 29. Thus, we find a d as stated. Set $\pi(i+1, j) = d$.

  Suppose that $\pi(i, j)$, $\pi(i+1, j)$, and $\pi(i+1, j+1)$ are already defined (i.e. one "negative stair" of the staircase). This is completely analogous to the first case, using the fact that $\mathcal{F}$ validates the concept $\forall x.\forall y.B \to \exists y.\exists x.B$.

Hence we have defined the mapping π as required. ❑

Proving Lemma 24 is now simple. First assume that there are interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$. By Lemma 31, $\mathcal{I}'$ is based on a frame $\mathcal{F}$ that contains a morphic image of $\mathbb{Z} \times \mathbb{Z}$ with origin $a^{\mathcal{I}'}$. Thus, there exists a total function $\pi : \mathbb{Z} \times \mathbb{Z} \to \Delta^{\mathcal{F}}$ such that Conditions 1 to 3 from Definition 30 are satisfied. Now define a mapping $\tau : \mathbb{Z} \times \mathbb{Z} \to T$ by setting $\tau(i, j) := t$ if $\pi(i, j) \in D_t^{\mathcal{I}'}$. By referring to Lines (11) to (13) and Conditions 1 to 3 from Definition 30, it is straightforward to verify that τ is a solution to $\mathcal{D}$.

Conversely, from a solution τ to $\mathcal{D}$, we define two interpretations $\mathcal{I}$ and $\mathcal{I}'$ as follows:

$$
\begin{array}{rcl}
\Delta^{\mathcal{I}} & := & \Delta^{\mathcal{I}'} := \mathbb{Z} \times \mathbb{Z}; \\
a^{\mathcal{I}} & := & a^{\mathcal{I}'} := (0, 0); \\
A^{\mathcal{I}} & := & B^{\mathcal{I}} := \emptyset; \\
A^{\mathcal{I}'} & := & B^{\mathcal{I}'} := \Delta^{\mathcal{I}}; \\
x^{\mathcal{I}} & := & x^{\mathcal{I}'} := \{((i, j), (i+1, j)) \mid i, j \in \mathbb{Z}\}; \\
y^{\mathcal{I}} & := & y^{\mathcal{I}'} := \{((i, j), (i, j+1)) \mid i, j \in \mathbb{Z}\}; \\
u^{\mathcal{I}} & := & u^{\mathcal{I}'} := \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}; \\
D_t^{\mathcal{I}} & := & D_t^{\mathcal{I}'} := \{(i, j) \mid \tau(i, j) = t\}, \text{ for each } t \in T.
\end{array}
$$

It is straightforward to verify that $\mathcal{I} \models \mathcal{A}$, and $\mathcal{I}' \models a : C_{\mathcal{D}}$. We now show that $\mathcal{I} \Rightarrow_S \mathcal{I}'$. First, it is easily verified that $\mathcal{I}'$ satisfies the post-conditions of S. Now assume that there is an interpretation $\mathcal{J}$ such that $\mathcal{J}$ satisfies the post-conditions of S, $\mathcal{J} \preceq_{\mathcal{I}} \mathcal{I}'$,

and $\mathcal{J} \neq \mathcal{I}'$. Observe that, since $\mathcal{I}$ and $\mathcal{I}'$ are based on the same frame, $\mathcal{J}$ must also be based on the same frame. This means that one of the following two cases applies:

- $A^{\mathcal{J}} \subsetneq A^{\mathcal{I}'}$. Then we have a contradiction to the assumption that $\mathcal{J}$ satisfies the post-condition, in particular Line (4).

- $B^{\mathcal{J}} \subsetneq B^{\mathcal{I}'}$. Let $(i,j) \in (\neg B)^{\mathcal{J}} \cap B^{\mathcal{I}'}$. Since $\mathcal{J}$ satisfies Post-condition (6), we have $d \in (\forall x^{-}.\forall y^{-}.\neg Q)^{\mathcal{J}}$, which implies that $(i-1, j-1) \notin Q^{\mathcal{J}}$. Thus, $(i-1, j-1) \in (\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B)^{\mathcal{J}}$, which implies $(i,j) \in B^{\mathcal{J}}$, thus contradicting $(i,j) \in (\neg B)^{\mathcal{J}}$.

This finishes the proof of Lemma 24.

5 Conclusion

The main technical result of this paper is that standard problems in reasoning about action (projection, executability) become decidable if one restricts the logic for describing pre- and post-conditions as well as the state of the world to certain decidable description logics $\mathcal{L}$. The complexity of these inferences is determined by the complexity of standard DL reasoning in $\mathcal{L}$ extended with nominals.

The framework presented here is a first proposal for a formalism describing the functionality of Web services that takes into account the research undertaken both in the field of knowledge representation and reasoning for ontology engineering, and in the field of reasoning about actions. Clearly, this framework can be extended in several directions. Firstly, instead of using an approach similar to regression to decide the projection problem, one could also try to apply *progression*, i.e., to calculate a successor ABox that has as its models all the successors of the models of the original ABox. Secondly, the expressiveness of the basic action formalism introduced by Reiter has been extended in several directions, and we need to check for which of these extensions our results still hold. Thirdly, we have used only composition to construct composite services, whereas OWL-S proposes also more complex operators. These could, for example, be modeled by appropriate GOLOG programs. Finally, to allow for automatic composition of services, one would need to look at how planning can be done in our formalism.

Acknowledgements

Our thanks go to Alexei Lisitsa and Michael Thielscher for fruitful discussions, and to Evgeny Zolin for pointing out a mistake in an earlier version of the proof of Theorem 22.

References

[1] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in Lecture Notes in Computer Science, pages 307–321. Springer-Verlag, 1999.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003. To appear.

[4] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1 - 2):353–367, 1996.

[5] G. Brewka and J. Hertzberg. How to do things with worlds. *Journal of Logic and Computation*, 3:517–532, 1993.

[6] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57(2-3):227–270, Oct. 1992.

[7] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[8] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 161–166. Morgan-Kaufmann, 2001.

[9] A. Herzig. The pma revisited. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*. Morgan Kaufmann, 1996.

[10] I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR98)*, pages 636–647, 1998.

[11] I. Horrocks and P. Patel-Schneider. The generation of DAML+OIL. In C. Goble, D. L. McGuinness, R. Möller, and P. F. Patel-Schneider, editors, *Proceedings of the International Workshop in Description Logics 2001 (DL2001)*, number 49 in CEUR-WS (http://ceur-ws.org/), pages 30–35, 2001.

[12] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[13] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In D. MacAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, number 1831 in Lecture Notes in Computer Science. Springer-Verlag, 2000.

[14] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[15] R. Lara, H. Lausen, S. Arroyo, J. de Bruijn, and D. Fensel. Semantic web services: description requirements and current technologies. In *Proceedings of the International Workshop on Electronic Commerce, Agents, and Semantic Web Services*, 2003.

[16] H. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1988.

[17] H. J. Levesque, R. Reiter, L. Lesperance, F. Lin, and R. B. Scherl. GOLOG: a logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.

[18] V. Lifschitz. Frames in the space of situations. *Artificial Intelligence Journal*, 46:365–376, 1990.

[19] F. Lin. Embracing causality in specifying the indeterminate effects of actions. In B. Clancey and D. Weld, editors, *Proc. of the 14th Nat. Conf. on Artificial Intelligence (AAAI-96)*, pages 670–676, Portland, OR, Aug. 1996. MIT Press.

[20] S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*, 16(2):46–53, 2001.

[21] S. A. McIlraith and T. C. Son. Adapting golog for composition of semantic web services. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, pages 482–496, San Francisco, CA, 2002. Morgan Kaufmann Publishers.

[22] D. S. Nau, T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[23] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann, 1991.

[24] L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM Journal on Computing*, 29(4):1083–1117, Aug. 2000.

[25] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46:325 – 361, 1999.

[26] I. Pratt-Hartmann. Counting quantifiers and the stellar fragment. Available at The Mathematics Preprint Server, `www.mathpreprints.com`, 2003.

[27] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[28] R. Reiter. *Knowledge in Action*. MIT Press, 2001.

[29] E. Sandewall. *Features and Fluents*. Oxford University Press, 1994.

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

[31] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Journal of Web Semantics*, 1(4):377–396, 2004.

[32] The OWL-S Coalition. Owl-s 1.1 (beta) draft release, 2004. http://www.daml.org/services/owl-s/1.1/.

[33] The W^{3}C Consortium. The web ontology language (owl), 2005. http://www.w3.org/2004/OWL/.

[34] M. Thielscher. Introduction to the Fluent Calculus. *Electronic Transactions on Artificial Intelligence*, 2(3–4):179–192, 1998.

[35] M. Thielscher. Nondeterministic actions in the fluent calculus: Disjunctive state update axioms. In S. Hölldobler, editor, *Intellectics and Computational Logic*, pages 327–345. Kluwer Academic, 2000.

[36] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[37] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.

[38] M. Winslett. Reasoning about action using a possible models approach. In *AAAI*, pages 89–93, Saint Paul, MN, 1988.

[39] M. Winslett. *Updating Logical Databases*. Cambridge University Press, Cambridge, England, 1990.

[40] M. Winslett. Epistemic aspects of databases. In D. Gabbay, A. Galton, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 4 "Epistemic and Temporal Reasoning"*, pages 133–174. Oxford University Press, 1995.

A Complexity of $\mathcal{ALCQO}$ with acyclic TBoxes

We show that, in $\mathcal{ALCQO}$, ABox consequence w.r.t. acyclic TBoxes can be decided in PSpace. For simplicity, we presuppose that numbers in number restrictions are coded in unary: the same result could be proved for the binary coding case, but this would necessitate the introduction of binary counters and further complicate the presentation of the algorithm. We proceed in two steps: first, we reduce ABox consequence to satisfiability, and second, we develop a K-worlds style algorithm, as known from modal logics [14], for deciding satisfiability in $\mathcal{ALCQO}$ w.r.t. acyclic TBoxes.

First for the reduction. Let $\mathcal{A}$ an ABox, $\mathcal{T}$ an acyclic TBox, and φ an ABox assertion. As noted at the beginning of Section 3.1, we may w.l.o.g. assume that $\varphi = A_0(a_0)$ for some concept name A_0. Then $\mathcal{A}, \mathcal{T} \models \varphi$ iff the following concept is unsatisfiable w.r.t. $\mathcal{T}$, where u is a role name not occurring in $\mathcal{A}$ and $\mathcal{T}$:

$$\bigsqcap_{C(a)\in\mathcal{A}} \exists u.(\{a\} \sqcap C) \sqcap \bigsqcap_{r(a,b)\in\mathcal{A}} \exists u.(\{a\} \sqcap \exists r.\{b\}) \sqcap \bigsqcap_{\neg r(a,b)\in\mathcal{A}} \exists u.(\{a\} \sqcap \forall r.\neg\{b\}) \sqcap \neg A_0(a_0).$$

Now for the K-worlds style algorithm for deciding satisfiability. Let C be a concept and $\mathcal{T}$ an acyclic TBox. We start with introducing a number of notions.

- We use $\mathsf{ind}(C, \mathcal{T})$ to denote the number of individual names occurring in C or $\mathcal{T}$, and $a_1, \ldots, a_{\mathsf{ind}(C,\mathcal{T})}$ to denote these names.
- $\max_{C,\mathcal{T}}$ denotes the sum of over all numbers that occur in number restrictions in C and $\mathcal{T}$, i.e.,
$$\max_{C,\mathcal{T}} := \sum_{(\bowtie n r D) \text{ occurs in } C \text{ or } \mathcal{T}} n.$$
- We use $\dot{\neg} D$ to denote the *negation normal form (NNF)* of $\neg D$: $\dot{\neg} D$ is obtained from $\neg D$ by pushing negation inwards using de Morgan's laws and the duality between atleast and atmost number restrictions [13]. It is well-known that $\dot{\neg} D$ can be computed in polynomial time and that it is of length linear in the length of $\neg D$.
- With $\mathsf{cl}_\mathcal{T}(C)$, we denote the smallest set of concepts S such that $C \in S$, S is closed under taking negation normal forms, and
$$\{D \mid A \doteq D \in \mathcal{T} \text{ and some } D' \in S \text{ uses } A \text{ in } \mathcal{T}\} \subseteq S,$$
where "uses" is defined as in Definition 3. For X a set of concepts, we set $\mathsf{cl}_\mathcal{T}(X) := \bigcup_{D\in X} \mathsf{cl}_\mathcal{T}(D)$.

We now define a crucial notion underlying the algorithm. Let X be a set of concepts. A set $S \subseteq \mathsf{cl}_\mathcal{T}(X)$ is called a *type for X and $\mathcal{T}$* , written $S \in \mathrm{Type}(X, \mathcal{T})$, if S satisfies the following conditions:

1. S does not contain both D and $\dot{\neg} D$, for all concepts C,
2. for each $D \in X$, either $D \in S$ or $\dot{\neg} D \in S$,

3. if $D_1 \sqcap D_2 \in S$, then $\{D_1, D_2\} \subseteq S$,

4. if $D_1 \sqcup D_2 \in S$, then $\{D_1, D_2\} \cap S \neq \emptyset$, and

5. if $A \doteq D \in \mathcal{T}$ and $\{A, D\} \subseteq \mathsf{cl}_{\mathcal{T}}(X)$, then $A \in S$ iff $D \in S$.

Finally, we use a function RandomType$(X, \mathcal{T})$ which non-deterministically returns a type $S \in \mathrm{Type}(X, \mathcal{T})$.

We are now ready to formulate the algorithm SatNom$(C, \mathcal{T})$ for deciding satisfiability in $\mathcal{ALCQO}$. Together with its auxiliary function Sat$(S, S_1, \ldots, S_\ell)$, the algorithm is given in pseudo code in Figure 3. The following lemma states that it is indeed a PSPACE decision procedure for satisfiability in $\mathcal{ALCQO}$ w.r.t. acyclic TBoxes. We use $|C|$ to denote the *size* of C, i.e., the number of symbols used to write C. The size $|\mathcal{T}|$ of $\mathcal{T}$ is defined analogously.

Lemma 32. *Let C be an $\mathcal{ALCQO}$-concept and $\mathcal{T}$ be an acyclic $\mathcal{ALCQO}$ TBox.*

1. *The recursion depth of Sat$(\cdot)$ is bounded by $|C| + |\mathcal{T}|$.*

2. *SatNom$(C, \mathcal{T})$ uses space bounded polynomially by $|C| + |\mathcal{T}|$.*

3. *SatNom$(C, \mathcal{T})$ returns "satisfiable" iff C is satisfiable w.r.t. $\mathcal{T}$.*

Proof. Let C be an $\mathcal{ALCQO}$-concept and $\mathcal{T}$ be an acyclic $\mathcal{ALCQO}$ TBox. To prove Point 1 of Lemma 32, it is convenient to first introduce some additional notions: the depth of concepts is defined inductively as follows:

$$\begin{array}{rcl} \mathsf{depth}(A) & := & \mathsf{depth}(\neg A) := 0 \text{ for } A \in \mathsf{N_C} \\ \mathsf{depth}(D_1 \sqcap D_2) & := & \mathsf{depth}(D_1 \sqcup D_2) := \max\{\mathsf{depth}(D_1), \mathsf{depth}(D_2)\} \\ \mathsf{depth}(\geqslant n\ r\ D) & := & \mathsf{depth}(\leqslant n\ r\ D) := 1 + \mathsf{depth}(D) \end{array}$$

The unfolding depth of a concept C is defined as $\mathsf{udepth}(C) := \mathsf{depth}(\mathsf{unfold}(C, \mathcal{T}))$, where $\mathsf{unfold}(C, \mathcal{T})$ denotes the result of recursively and exhaustively replacing each concept A used by C with D for each $A \doteq D \in \mathcal{T}$. For a set of concepts S, the depth of S is defined as the maximum depth of concepts in S, and the unfolding depth of S is defined analogously.

To show Point 1, consider the random types T_i guessed by RandomType$(X, \mathcal{T})$ in Sat$(S, S_1, \ldots, S_\ell)$. Each $E \in X$ is a sub-concept of some $(\bowtie\ m\, r\, E) \in S$, and thus $\mathsf{depth}(X) < \mathsf{depth}(S)$. Since each subconcept of a concept in E is also a subconcept of a concept in S, this yields $\mathsf{udepth}(X) < \mathsf{udepth}(S)$. Since $T_i \subseteq \mathsf{cl}_{\mathcal{T}}(X)$, we have $\mathsf{udepth}(T_i) \leq \mathsf{udepth}(\mathsf{cl}_{\mathcal{T}}(X))$ implying $\mathsf{udepth}(T_i) \leq \mathsf{udepth}(X)$ since $\mathsf{udepth}(X) = \mathsf{udepth}(\mathsf{cl}_{\mathcal{T}}(X))$. Taking together $\mathsf{udepth}(T_i) \leq \mathsf{udepth}(X)$ and $\mathsf{udepth}(X) < \mathsf{udepth}(S)$, we obtain

$$\mathsf{udepth}(T_i) < \mathsf{udepth}_(S).$$

As this holds in every recursion call, the recursion depth of Sat$(\cdot)$ is bounded by $\mathsf{udepth}(C) \leq |C| + |\mathcal{T}|$.

The second point is then an easy consequence of the first point and the fact that the information we need to store in each call to Sat$(\cdot)$ needs space $\mathcal{O}((|C| + |\mathcal{T}|)^2)$.

```
DEFINE PROC SatNom(C, T)
    Guess some ℓ ≤ ind(C, T) and a mapping π : {a_1, ..., a_ind(C,T)} → {0, ..., ℓ}
    FOR EACH 0 ≤ i ≤ ℓ DO
        S_i := RandomType({C, {a_1}, ..., {a_ind(C,T)}}, T)
    OD
    IF (FORSOME 0 ≤ i ≤ ℓ, {{a_j} | π(a_j) = i} ≠ {{a_j} | {a_j} ∈ S_i})
       THEN RETURN "unsatisfiable"
    IF (FORALL 0 ≤ i ≤ ℓ, C ∉ S_i)
       THEN RETURN "unsatisfiable"
    IF (FORALL 0 ≤ i ≤ ℓ, Sat(S_i, S_1, ..., S_ℓ) = "satisfiable")
       THEN RETURN "satisfiable"
    RETURN "unsatisfiable"


DEFINE PROC Sat(S, S_1, ..., S_ℓ)
    FOR EACH r with (⋈ n r D) ∈ S DO
        GUESS some n_r with 0 ≤ n_r ≤ max_{C,T}
        FOR EACH 1 ≤ i ≤ n_r DO
            T_i := RandomType({E | (⋈ m r E) ∈ S}, T) OD
        FOR EACH (≤ n r D) ∈ S DO
            IF (there are more than n T_i with D ∈ T_i)
            THEN RETURN "unsatisfiable" OD
        FOR EACH (≥ n r D) ∈ S DO
            IF (there are less than n T_i with D ∈ T_i)
            THEN RETURN "unsatisfiable" OD
        FOR EACH 1 ≤ j ≤ ind(C, T) DO
            IF (there more than 2 T_i with {a_j} ∈ T_i) OR
               (there is a T_i with {a_j} ∈ T_i and T_i ⊈ S_π(a_j) )
            THEN RETURN "unsatisfiable" OD
        FOR EACH i with 1 ≤ i ≤ n_r and
                EACH T_i with T_i ∩ {{a_1}, ..., {a_ind(C,T)}} = ∅ DO
            IF Sat(T_i, S_1, ..., S_ℓ) = "unsatisfiable"
            RETURN "unsatisfiable" OD
    OD
    RETURN "satisfiable"
```

Figure 3: The algorithms SatNom$(C, \mathcal{T})$ and Sat$(S, S_1, \ldots, S_\ell)$

For the "if" direction of the third point, we use a model $\mathcal{I}$ of C w.r.t. $\mathcal{T}$ to "guide" the non-deterministic guesses of SatNom and Sat such that "satisfiable" is returned. Since $\mathcal{I}$ is a model of C, we find a sequence of pairwise distinct elements $x_0, \dots, x_\ell \in \Delta^{\mathcal{I}}$, for some $\ell \leq \mathsf{ind}(C, \mathcal{T})$, such that

1. there is an $i < \ell$ such that $x_i \in C^{\mathcal{I}}$;
2. there is a mapping $\pi : \{a_1, \dots, a_{\mathsf{ind}(C,\mathcal{T})}\} \to \{0, \dots, \ell\}$ such that, for $0 \leq i \leq \ell$, we have $x_i \in \{\pi(a_i)\}^{\mathcal{I}}$.

Use the number ℓ and the mapping π obtained in this way for the guess made by SatNom$(C, \mathcal{T})$ in Line 2. Next, we determine SatNom's remaining guesses $S_0, \dots, S_\ell$: for $0 \leq i \leq \ell$, set

$$S_i := \mathrm{type}(x_i, \{C, \{a_1\}, \dots, \{a_{\mathsf{ind}(C,\mathcal{T})}\}\}, \mathcal{T}).$$

where $\mathrm{type}(x, X, \mathcal{T})$ denotes the largest set $S \in \mathrm{Type}(X, \mathcal{T})$ such that $x \in D^{\mathcal{I}}$ for all $D \in S$.

It remains to determine the numbers n_r and the types T_i guessed in the function Sat. To determine them, we simultaneously define a mapping

$$\Pi : \mathbb{N} \to \Delta^{\mathcal{I}}$$

that associates, with the i-th call to Sat, a domain element $\Pi(i) \in \Delta^{\mathcal{I}}$ such that

$$\Pi(i) \in D^{\mathcal{I}} \text{ for each } D \text{ in the first argument of the } i\text{th call to Sat} \qquad (*)$$

Intuitively, the domain element $\Pi(i)$ realizes the type that is passed as the first argument to the ith call of Sat. To determine the remaining guesses and define Π, we distinguish two cases:

1. The ith call $\mathrm{Sat}(S, S_1, \dots, S_\ell)$ is performed by NomSat. Set $\Pi(i) := x_j$ if S is the set S_j guessed by NomSat, for $0 \leq j \leq \ell$. It is easily checked that $(*)$ is satisfied. Now fix an r with $(\bowtie\ n\, r\, D) \in S$. The guess n_r is defined to be the minimum of

   - $\#\{y \in \Delta^{\mathcal{I}} \mid (\Pi(i), y) \in r^{\mathcal{I}}\}$ and
   - $\max_{C,\mathcal{T}}$.

   Since $\Pi(i)$ satisfies $(*)$ and by definition of $\max_{C,\mathcal{T}}$, we find a sequence $\langle y_1, \dots, y_{n_r} \rangle$ of pairwise distinct elements of $\Delta^{\mathcal{I}}$ such that $(\Pi(i), y_j) \in r^{\mathcal{I}}$ for $1 \leq j \leq n_r$ and, for each $(\geqslant n\ r\ D) \in S$, there are at least n y_j in P with $y_j \in D^{\mathcal{I}}$.

   We set, for each $1 \leq j \leq n_r$,

   $$T_j := \mathrm{type}(y_j, \{E \mid (\bowtie\ m\, r\, E) \in S\}, \mathcal{T}). \qquad (\dagger)$$

2. Let the ith call $\mathrm{Sat}(S, S_1, \dots, S_\ell)$ be performed by Sat itself. Then $S = T_j$ has been defined in the ith call to Sat using some domain element y_j in $(\dagger)$. We set $\Pi(i) := y_j$, which clearly preserves $(*)$. The guesses n_r and $T_1, \dots, T_{n_r}$ are then defined as in the previous case.

We prove that, if guided in the indicated way, Sat never returns "unsatisfiable". Due to Point 1, it thus terminates returning "satisfiable". There are several ways in which the algorithm may return "unsatisfiable":

- In SatNom, we may have $\{\{a_j\} \mid \pi(a_j) = i\} \neq \{\{a_j\} \mid \{a_j\} \in S_i\}$ for some i with $0 \leq i \leq \ell$.

  Clearly, this is impossible by our choice of π and S_i.

- In the ith call to Sat, there is a role r and a $(\leqslant n\ r\ D) \in S$ such that there are more than n T_j with $D \in T_j$.

  By $(*)$, we have $\Pi(i) \in (\leqslant n\ r\ D)^{\mathcal{I}}$, and thus the definition of the semantics implies that there are at most n $y \in \Delta^{\mathcal{I}}$ with $(\Pi(i), y) \in r^{\mathcal{I}}$ and $y \in D^{\mathcal{I}}$. Hence, by definition of T_j and due to the first property of types, there cannot be more than n T_j with $D \in T_j$.

- In the ith call to Sat, there is a role r and a $(\geqslant n\ r\ D) \in S$ such that there are less than n T_j with $D \in T_j$.

  Impossible by our definition of the types $T_1, \ldots, T_{n_r}$.

- In the ith call to Sat, there is a role r and a j with $1 \leq j \leq \mathsf{ind}(C, \mathcal{T})$ such that there are more than 2 T_k with $\{a_j\} \in T_k$ or there is a T_k with $\{a_j\} \in T_k$ and $T_k \not\subseteq S_{\pi(j)}$.

  The former is impossible by the semantics and our choice of types $T_1, \ldots, T_{n_r}$, and the latter is impossible by the semantics, our choice of $T_1, \ldots, T_{n_r}$, and our choice of $S_{\pi(j)}$.

This finishes the proof of the "if" direction.

For the "only if" direction of the third point, we use a run of $\mathrm{SatNom}(C, \mathcal{T})$ returning "satisfiable" to build a model $\mathcal{I}$ of C w.r.t. $\mathcal{T}$. Let ℓ be the number, π the mapping, and S_j, $0 \leq j \leq \ell$, the types guessed by $\mathrm{SatNom}(C, \mathcal{T})$. Moreover, let n_r^i be the number guessed in the i-th call to the Sat function for the role r, and let $T_1^{i,r}, \ldots, T_{n_r^i}^{i,r}$ be the types guessed for r in that call. Finally, we use T_i to denote the type that was passed as the first argument to the ith call of SatNom and write $T_j^{i,r} \rightsquigarrow i'$ if the i'th call of Sat was performed during the $i$the call to Sat with argument $T_j^{i,r}$. Define an interpretation $\mathcal{I}$ as follows:

$$
\begin{array}{rcl}
\Delta^{\mathcal{I}} & := & \{x_j \mid 0 \leq j \leq \ell\} \cup \{y_j^{i,r} \mid T_j^{i,r} \cap \{\{a_1\}, \ldots, \{a_{\mathsf{ind}(C,\mathcal{T})}\}\} = \emptyset\} \\
A^{\mathcal{I}} & := & \{x_j \mid A \in S_j\} \cup \{x_j^{i,r} \mid A \in T_j^{i,r}\} \\
r^{\mathcal{I}} & := & \{(x_j, x_k) \mid \exists i, n, m : S_j = T_i \text{ and } \{a_n\} \in T_m^{i,r} \cap S_k\} \cup \\
 & & \{(x_j, y_k^{i,r}) \mid S_j = T_i \text{ and } y_k^{i,r} \in \Delta^{\mathcal{I}}\} \cup \\
 & & \{(y_j^{i,s}, y_k^{i',r}) \mid T_j^{i,s} \rightsquigarrow i' \text{ and } y_k^{i,r} \in \Delta^{\mathcal{I}}\} \cup \\
 & & \{(y_j^{i,s}, x_k) \mid \exists i', n, m : T_j^{i,s} \rightsquigarrow i' \text{ and } \{a_n\} \in T_m^{i',r} \cap S_k\}
\end{array}
$$

By induction on the structure of concepts, we can show that $x_j \in D^{\mathcal{I}}$ for all $D \in S_j$ and all $x_j \in \Delta^{\mathcal{I}}$, and that $y_j^{i,r} \in D^{\mathcal{I}}$ for all $D \in T_j^{i,r}$ and all $y_j^{i,r} \in \Delta^{\mathcal{I}}$. Since $C \in S_k$ for

some $0 \leq k \leq \ell$ and due to Property 5 in the definition of types, we thus have that $\mathcal{I}$ is indeed a model of C w.r.t. $\mathcal{T}$. ❑

As an immediate consequence of Lemma 32 and PSpace-hardness of $\mathcal{ALC}$ [30], we thus have the following result.

Theorem 33. *Satisfiability of $\mathcal{ALCQO}$-concepts w.r.t. acyclic TBoxes is PSpace-complete if numbers inside number restrictions are coded in unary.*