A Description Logic Based Approach to Reasoning about Web Services

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

[1] Inst. for Theoretical CS
TU Dresden, Germany
lastname@tcs.inf.tu-dresden.de

[2] Department of CS
U. of Manchester, UK
sattler@cs.man.ac.uk

[3] Department of CS
U. 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.

1. INTRODUCTION

Description logics [3] play an important rôle in the Semantic Web since they are the basis of the W3C-recommended Web ontology language OWL [4, 13], which can be used to create semantic annotations describing the content of Web pages [32].

In addition to this static information, the Web also offers services, which allow their users to effect changes in the world, such as 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 [31] 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. However, the faithful representation of the dynamic behaviour of such 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. [27, 33]). Thus, it is not surprising that theories developed in these comunities have been applied in the context of Semantic Web services. For example, [18, 19] use the situation calculus [27] and GOLOG [15] to formalize the dynamic aspects of Web services and to describe their composition. In [30], OWL-S process models are translated into the planning language of the HTN planning system SHOP2 [20], 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, 37, 6, 9], 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 [26, 24, 14, 27] 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

WWW2005, May 10–14, 2005, Chiba, Japan.
.

Name	Syntax	Semantics
inverse role	s^-	$\{(y,x) \mid (x,y) \in s^{\mathcal{I}}\}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
at-least number restriction	$(\geqslant n\ r\ C)$	$\{x \mid \mathit{card}(\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}) \geq n\}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$

Table 1: Syntax and semantics of $\mathcal{ALCQIO}$.

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.

Because of the space constraints, all proofs and a more detailed discussion of the relationship to the situation calculus must be omitted. They can be found in [2].

2. DESCRIPTION LOGICS

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 6.

In DL, concepts are inductively defined with the help of a set of *constructors*, starting with a set $\mathsf{N_C}$ of *concept names*, a set $\mathsf{N_R}$ of *role names*, and a set $\mathsf{N_I}$ of *individual names*. The constructors determine the expressive power of the DL. Table 1 shows a minimal set of constructors from which all constructors of $\mathcal{ALCQIO}$ can be defined. The first row contains the only role constructor: in $\mathcal{ALCQIO}$, a *role* is either a role name $s \in \mathsf{N_R}$ or the inverse s^- of a role name. *Concepts* of $\mathcal{ALCQIO}$ are formed using the remaining constructors shown in Table 1, where r is a role, n a positive integer, and a an individual name. Using these constructors, several other constructors can be defined as abbreviations:

- $C \sqcup D := \neg(\neg C \sqcap \neg D)$ (disjunction),
- $\top := A \sqcup \neg A$ for a concept name A (top-concept),
- $\exists r.C := (\geqslant 1\ r\ C)$ (existential restriction),
- $\forall r.C := \neg\exists r.\neg C$ (value restriction),
- $(\leqslant n\ r\ C) := \neg(\geqslant (n+1)\ r\ C)$ (at-most restriction).

The DL that allows for negation, conjunction, and value 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 Table 2.

The semantics of $\mathcal{ALCQIO}$-concepts and roles is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps

- each concept name $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$,
- each role name $s \in \mathsf{N_R}$ to a binary relation $s^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, and
- each individual name $a \in \mathsf{N_I}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

The extension of $\cdot^{\mathcal{I}}$ to arbitrary concepts and roles is inductively defined, as shown in the third column of Table 1. Here, the function *card* yields the cardinality of the given set.

A *concept definition* is an identity of the form

$$A \equiv C,$$

where A is a concept name and C an $\mathcal{ALCQIO}$-concept. A *TBox* $\mathcal{T}$ is a finite set of concept definitions with unique left-hand sides. Concept names occurring on the left-hand side of a definition of $\mathcal{T}$ are called *defined in* $\mathcal{T}$ whereas the others are called *primitive in* $\mathcal{T}$. The TBox $\mathcal{T}$ is *acyclic* iff there are no cyclic dependencies between the definitions, i.e., the recursive substitution of defined concepts by their definitions always terminates. This process is called *expansion* of the TBox.

The *semantics* of TBox definitions is defined in the obvious way: the interpretation $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ iff it satisfies all its definitions, i.e.,

$$A^{\mathcal{I}} = C^{\mathcal{I}} \text{ holds for all } A \equiv C \text{ in } \mathcal{T}.$$

Any interpretation of the primitive concepts and of the role names can uniquely be extended to a model of the acyclic TBox $\mathcal{T}$. This is an easy consequence of the fact that acyclic TBoxes can be expanded [21].

An *ABox assertion* is of the form

$$C(a),\ s(a,b),\ \text{or } \neg s(a,b),$$

where $a, b \in \mathsf{N_I}$, C is a concept, and s a role name.[1] To improve readability, we will sometimes write the assertion $C(a)$ in the form $a : C$. An *ABox* is a finite set of ABox assertions. The interpretation $\mathcal{I}$ is a *model* of the ABox $\mathcal{A}$ iff it satisfies all its assertions, i.e., $a^{\mathcal{I}} \in C^{\mathcal{I}}$ $((a^{\mathcal{I}}, b^{\mathcal{I}}) \in s^{\mathcal{I}}$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin s^{\mathcal{I}})$ for all assertions $C(a)$ $(s(a,b)$, $\neg s(a,b))$ in $\mathcal{A}$. If φ is an assertion, then we write $\mathcal{I} \models \varphi$ iff $\mathcal{I}$ satisfies φ.

Various reasoning problems are considered for DLs. For the purpose of this paper, it suffices to introduce concept satisfiability and ABox consistency:

- the concept C is *satisfiable* w.r.t. the TBox $\mathcal{T}$ iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$;
- the ABox $\mathcal{A}$ 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 $\mathcal{A}$.

3. SERVICE DESCRIPTIONS

We now introduce the formalism for 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

[1] Negated role assertions are usually not considered in DL, but they are very useful as pre- and post-conditions. Disallowing inverse roles in ABox assertions is not a restriction since $s^-(a,b)$ can be expressed by $s(b,a)$.

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	

Table 2: Fragments of $\mathcal{ALCQIO}$.

viewed as a compact representation of all its ground instances. The handling of such parametric services takes place "outside" of our formalism and is not discussed in detail in the current paper. We may 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 1 (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. 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 [16, 28]. 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 who 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 involve 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:

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

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:

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

The meaning of the concepts used in S_1, S_2, and S_3 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, 37, 6, 9]. 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 [27]. We discuss this approach in Section 3. 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 2 (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

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

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 aber preceq sieht anders aus $\preccurlyeq_{\mathcal{I},S}$ instead of $\preccurlyeq_{\mathcal{I},S,\emptyset}$.

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 3 (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 $\Rightarrow_{S_1,\ldots,S_k}^{\mathcal{T}}$.

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 4. *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 6 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 even yields a well-defined semantics for these services (though not necessarily a satisfactory one).

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 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. [27]. In our setting, they can formally be defined as follows:

Definition 5 (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 conditions are true in all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$:
  - $\mathcal{I} \models \mathsf{pre}_1$ and
  - 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 4, 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

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

it is easily checked that the composite service $S = S_1, S_2, S_3$ is executable, and that $\texttt{receives\_c\_benef\_for}(a,d)$ 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 $\mathcal{A} \cup \{\neg\varphi\}$ is inconsistent w.r.t. $\mathcal{T}$ 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 6. *Executability and projection can be reduced to each other in polynomial time.*

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. □

Relationship with SitCalc

We have chosen a possible models approach to define the effects of our services. More established and widely used in the reasoning about action community is the situation calculus [27]. In contrast to the PMA, the situation calculus uses an axiomatic approach to define the effects of actions. However, if we consider services without occlusions, then our approach can be seen as an instance of the situation calculus.

Suppose an ABox $\mathcal{A}$, an acyclic TBox $\mathcal{T}$, and a composite service $S_1, \ldots, S_k$ are given. First, we can get rid of the TBox by expanding it and then replacing in $\mathcal{A}$ and the services $S_1, \ldots, S_k$ the defined concepts with their definitions.[2] Consider now the simple description of the relation $\Rightarrow_S^{\mathcal{T}}$ given in Lemma 4. By taking the standard translation of $\mathcal{ALCQIO}$ into first-order logic [3], we can easily translate this description into *action pre-conditions* and *successor state axioms* in the sense of [27]. In this setting, primitive concepts and role names are regarded as fluents. We take the first-order translation of the ABox as the initial state, and then we can show that our notions of executability and projection are instances of Reiter's definitions (see [2] for details).

The translation of our approach into a situation calculus axiomatization à la Reiter shows that our formalism is firmly based on research on reasoning about action. However, this does not mean that the inference problems introduced above can be solved using an implemented system for reasoning about action, such as GOLOG [15]. In fact, in Reiter's approach, *regression* [27] is used to solve the executability and the projection problem. However, when applied to (the translation of) our services, regression yields a standard first-order theory, which is not in the scope of what GOLOG can handle without calling a general first-order theorem prover. Thus, the translation into situation calculus does not directly provide us with decidability or complexity results for our reasoning problems.

[2] Alternatively, we could handle the TBox as state constraints.

4. DECISION PROCEDURES

We develop reasoning procedures for the reasoning services introduced in Section 3 and 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 6, we can restrict the attention to the projection problem. We solve this problem by an approach that is similar to the regression operation used in the situation calculus approach [27]: 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 single interpretations $\mathcal{I}$. Concerning the standard reasoning tasks, we consider two options:

Firstly, we show that the theory we obtain can again be expressed by a description logic TBox and ABox. This way, projection is reduced to the inconsistency of DL ABoxes, 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 show that 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 [23, 25], 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 7. *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 inconsistency of ABoxes in $\mathcal{LO}$, the extension of $\mathcal{L}$ with nominals.

Reduction to DL Reasoning

We reduce projection in fragments $\mathcal{L}$ of $\mathcal{ALCQIO}$ to ABox (in)consistency 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 8. *Let $\mathcal{L} \in \{\mathcal{ALC}, \mathcal{ALCI}, \mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQ}, \mathcal{ALCQO}, \mathcal{ALCQI}, \mathcal{ALCQIO}\}$. Then projection of composite services formulated in $\mathcal{L}$ can be polynomially reduced to ABox inconsistency in $\mathcal{LO}$ w.r.t. acyclic TBoxes.*

Let $\mathcal{L}$ be one of the languages listed in Theorem 8, 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

> φ_0 is a consequence of applying $S_1, \ldots, S_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}}$ is inconsistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.

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 the $\mathcal{I}_i$-part of $\mathcal{I}$ is a model of $\mathcal{T}$, for each $i \leq n$.
- 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 $\mathcal{I}_i$.
- 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 $\mathcal{I}_i$, *but only with respect to the named domain elements*. Since concept membership of unnamed elements never changes, the "unnamed part" of the interpretation of A in $\mathcal{I}_i$ can be found in $A^{(0)}$, for *any* $i \leq n$.
- 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 $\mathcal{I}_i$ *but only concerning those role relationships where both involved domain elements are named*. All other role relationships never change and are stored in $r^{(0)}$.
- We use a concept name N 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.
- Finally, we use 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 $\mathcal{I}_i$ as desired. Details are given in Figure 1, where $r^{-(i)}$ denotes $(r^{(i)})^-$ in the concept definitions for number restrictions. The first concept definition reflects the fact that concept names $A^{(i)}$ only represent the extension of A in $\mathcal{I}_i$ *for named domain elements*. To get $T_A^{(i)}$, the full extension of A in $\mathcal{I}_i$, 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 concept definitions.

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

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

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

Figure 1: The TBox $\mathcal{T}_{\mathsf{sub}}$.

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

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

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

We now formalize the minimization of changes on named elements. For $1 \leq i \leq n$ the ABox $\mathcal{A}_{\mathsf{min}}^{(i)}$ contains

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

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

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

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

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

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

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

$$
\begin{aligned}
\mathcal{A}_{\mathsf{red}} \quad := \quad & \mathcal{A}_{\mathsf{ini}} \cup \mathcal{A}_{\mathsf{aux}} \cup\\
& \mathcal{A}_{\mathsf{post}}^{(1)} \cup \cdots \cup \mathcal{A}_{\mathsf{post}}^{(n)} \cup\\
& \mathcal{A}_{\mathsf{min}}^{(1)} \cup \cdots \cup \mathcal{A}_{\mathsf{min}}^{(n)} \cup\\
& \{\neg T_{A_0}^{(n)}(a_0)\}.
\end{aligned}
$$

The proof of the following lemma can be found in [2].

Lemma 9. *$A_0(a_0)$ is a consequence of applying $S_1, \ldots, S_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}}$ is inconsistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

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 9 immediately yields Theorem 8. Thus, for the DLs $\mathcal{L}$ considered in Theorem 8, upper complexity bounds for ABox inconsistency 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 consistency in $\mathcal{ALCQO}$ w.r.t. acyclic TBoxes. For the sake of completeness, we prove that this problem is PSPACE-complete in Appendix A of [2]. Lower complexity bounds carry over from ABox inconsistency in a DL $\mathcal{L}$ to projection in the same DL: $\mathcal{A}$ is not consistent w.r.t. $\mathcal{T}$ iff $a : \bot$ 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 $\mathcal{L}$ that allow for nominals or where the addition of nominals does *not* increase the complexity of reasoning.

Corollary 10. *Executability and projection w.r.t. acyclic TBoxes are*

1. PSPACE*-complete for $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCQ}$, $\mathcal{ALCQO}$;*
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 8 and the following results: ABox (in)consistency in

- $\mathcal{ALC}$ w.r.t. acyclic TBoxes is PSPACE-hard [29] (yields lower bounds of Point 1);
- $\mathcal{ALCQO}$ w.r.t. acyclic TBoxes is in PSPACE, which is proved in Appendix A of [2] (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 [35] and [23] (yields Points 4 and 5).

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

In Section 5, we prove matching lower bounds for Points 2 and 4 of Corollary 7.

Reduction to C2

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 (otherwise, the last two lines of Figure 1 yield an exponential blow-up). However, we cannot get any PSPACE or EXPTIME upper bounds from the C^2-translation since satisfiability in C^2 is NEXPTIME-complete [23, 25].

The intuitions underlying the reduction to C^2 are very similar to those given 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. Full details are given in [2]. We obtain the following result:

Theorem 11. *Projection of composite services formulated in $\mathcal{ALCQIO}$ can be polynomially reduced to satisfiability in C^2.*

Together with the reduction from executability to projection, this yields the following result, which sharpens Points 4 and 5 of Corollary 10 to cover also the case of binary coding of numbers inside number restrictions.

Corollary 12. *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 10. As shown in the following section, Corollary 12 also yields a tight upper bound for the fragment $\mathcal{ALCQI}$ of $\mathcal{ALCQIO}$.

5. HARDNESS RESULTS

We show that the upper bounds for executability and projection obtained in the previous two sections cannot be improved. In Section 4, 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 6, these bounds carry over to executability, thus matching Points 2 and 4 of Corollary 7. 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 13. *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 ABox assertion φ,*

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

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

For the proof of Theorem 13, let $\mathcal{L} \in \{\mathcal{ALCIO}, \mathcal{ALCFIO}\}$ and C an $\mathcal{L}$-concept whose unsatisfiability 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, 35, 36]. For the reduction, we reserve a new 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 each occurrence of 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:

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

Let $\mathcal{I}$ and $\mathcal{I}'$ be models witnessing that $\neg D_C(a)$ is *not* a consequence of S, i.e., $\mathcal{I} \models \mathcal{A}_C$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models D_C(a)$. The reduction rests on the following ideas:

- By the first conjunct of (the concept in) $\mathcal{A}_C$, the post-condition, and Lemma 4, the only difference between $\mathcal{I}$ and $\mathcal{I}'$ is that $a^{\mathcal{I}} = a^{\mathcal{I}'} \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$;
- By the first conjunct of (the concept in) $\mathcal{A}_C$ and the post-condition, the only difference between $\mathcal{I}$ and $\mathcal{I}'$ is that $a^{\mathcal{I}} = a^{\mathcal{I}'} \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$;
- Using the first and third conjunct of $\mathcal{A}_C$ together with the post-condition and the second conjunct of D_C, it can be shown that $(a^{\mathcal{I}}, x) \in u^{\mathcal{I}} = u^{\mathcal{I}'}$ for each x from

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

the relevant part rel of $\Delta^{\mathcal{I}}$, where rel is defined as the smallest set that contains $a^{\mathcal{I}}$ and is closed under taking successors for the roles from $\mathsf{rol}(C)$;

- Thus, the second conjunct of $\mathcal{A}_C$ ensures that $O^{\mathcal{I}} \cap \mathsf{rel} = \emptyset$ and $O^{\mathcal{I}'} \cap \mathsf{rel} = \{a^{\mathcal{I}}\}$.
- Due to the first conjunct of D_C, $C[O/\{n\}]$ is satisfied in the relevant part of $\mathcal{I}'$. By the previous item, the concept name O behaves like a nominal.

In [2], we prove the following lemma, which immediately yields Theorem 13.

Lemma 14. *The following statements are equivalent:*

1. *C is satisfiable.*
2. *$\neg D_C(a)$ is not a consequence of applying S in $\mathcal{A}_C$.*
3. *the composite service $S, (\{\neg D_C(a)\}, \emptyset, \emptyset)$ is not executable in $\mathcal{A}_C$.*

6. PROBLEMATIC EXTENSIONS

In the DL framework for reasoning about services proposed in this paper, we have adopted several syntactic restrictions:

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, complex concepts.

The purpose of this section is to provide a justification for these restrictions: we show that removing the first restriction leads to *semantic problems*, while removing the second and third restriction leads to *both semantic and computational problems.*

Transitive Roles

Transitive roles are offered by most modern DL systems [10, 8], and also by the ontology languages OWL, DAML+OIL, and OIL [13, 11, 7]. They are added to $\mathcal{ALCQIO}$ by reserving a subset of roles N_{tR} of N_{R} such that all $r \in \mathsf{N}_{\mathsf{tR}}$ 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 4, 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 $\mathsf{has\text{-}part}$. Hence, in contrast to our intuition, $\mathsf{has\text{-}part}(\mathsf{engine}, \mathsf{valve})$ is not a cosequence of applying S in $\{\mathsf{has\text{-}part}(\mathsf{engine}, \mathsf{valve})\}$.

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 [17, 34]. Thus, we need a mechanism for eliminating unwanted outcomes or preferring the desired ones. We leave such extensions as future work.

Cyclic TBoxes and GCIs

Assume that we admit arbitrary (also cyclic) TBoxes as defined in Section 2. 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 $\preccurlyeq_{\mathcal{I},S,\mathcal{T}}$ 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 $\preccurlyeq_{\mathcal{I},S,\mathcal{T}}$, 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 $\preccurlyeq_{\mathcal{I},S,\mathcal{T}}$ 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 [16]. The second option seems more feasible: if we adopt the least or greatest fixpoint semantics for TBoxes as first proposed by Nebel [22], 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 $\preccurlyeq_{\mathcal{I},S,\mathcal{T}}$, the only choice is to minimize *all* concept names, which corresponds to the problematic minimization of complex concepts mentioned above. Second, the missing 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.

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 $S = (\emptyset, \emptyset, \{a : A \sqcup B\})$ are not deterministic and thus introduce similar complications as discussed for transitive roles. However, disjunction is not the only constructor to introduce non-determinism when allowed in post-conditions:

- 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.

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

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

  in the ABox $\{\mathsf{Female}(\mathsf{mary})\}$.

- 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., $\mathsf{Filled}(\mathsf{tire2})$ is not a consequence of applying the service

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

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

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 [17, 34], none of the mainstream action formalisms allows arbitrary formulas in post-conditions. Indeed, most formalisms such as the basic situation calculus restrict themselves to literals in post-conditions [27, 33]—just as our non-generalized services do.

Computational Problems

Executability and projection for generalized services easily become undecidable. To illustrate this, we prove undecidability of these reasoning tasks for the DL $\mathcal{ALCFI}$ that has been introduced in Section 5.[5] This result should be contrasted with the fact that, by Theorem 7, reasoning with non-generalized services is decidable even for powerful extensions of $\mathcal{ALCFI}$. Note that $\mathcal{ALCFI}$ may be viewed as a fragment of OWL light, the weakest OWL dialect [12].

Theorem 15. *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 $C(a)$ is a consequence of applying S in $\mathcal{A}$;*
- *decide whether the composite service S, S' is executable in $\mathcal{A}$, where $S' = (\{C(a)\}, \emptyset, \emptyset)$.*

The proof of Theorem 15 is by reduction of the domino problem to non-consequence and non-executability.

Definition 16. 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

[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.

[5] Recall that $\mathcal{ALCFI}$ is obtained from $\mathcal{ALCQI}$ by limiting numbers occurring in number restrictions to $\{0, 1\}$.

$$
\begin{aligned}
\mathcal{A} = \{ \quad & a : \neg A && (1)\\
& a : \forall u.\Big(\bigsqcap_{r\in\{x,y,u,x^-,y^-,u^-\}} \forall r.\neg A\Big) && (2)\\
& a : \forall u.\neg B \quad \} && (3)\\
\mathsf{post} = \{ \quad & a : \forall u.A && (4)\\
& a : \forall u.((\forall x^-.\forall y^-.\neg Q) \sqcup B) \quad \} && (5)\\
& \quad \text{with } Q := \forall x.\forall y.B \rightarrow \exists y.\exists x.B && (6)\\
C_{\mathcal{D}} = \quad & A \sqcap && (7)\\
& \forall u.\Big(\bigsqcap_{r\in\{x,y,u,x^-,y^-,u^-\}} \forall r.A\Big) \sqcap && (8)\\
& \forall u.B \sqcap && (9)\\
& \forall u.(\exists x.\top \sqcap \exists y.\top \sqcap \exists x^-.\top \sqcap \exists y^-.\top) \sqcap && (10)\\
& \forall u.((\leq 1\ x) \sqcap (\leq 1\ y)) \sqcap && (11)\\
& \forall u.((\leq 1\ x^-) \sqcap (\leq 1\ y^-)) \sqcap && (12)\\
& \forall u.\bigsqcap_{\substack{t,t'\in T\\ \text{with } t\neq t'}} \neg(D_t \sqcap D_{t'}) \sqcap && (13)\\
& \forall u.\bigsqcup_{(t,t')\in H} (D_t \sqcap \forall x.D_{t'}) \sqcap && (14)\\
& \forall u.\bigsqcup_{(t,t')\in V} (D_t \sqcap \forall y.D_{t'}) && (15)
\end{aligned}
$$

Figure 2: The Reduction Ingredients.

- if $\tau(x, y) = t$ and $\tau(x + 1, y) = t'$, then $(t, t') \in H$
- 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 $\mathcal{D}$.

It is well-known that the existence of a solution for a domino system is an undecidable problem [5]. 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 $(\leqslant 1\ r\ \top)$. As the proof is slightly involved, we only provide some core intuitions. To this end, let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_S \mathcal{I}'$, and $\mathcal{I}' \models C_{\mathcal{D}}(a)$. Then we have the following:

- Lines (1), (2), (4), (7), and (8) ensure that u connects $a^{\mathcal{I}} = a^{\mathcal{I}'}$ to every relevant domain element in both $\mathcal{I}$ and $\mathcal{I}'$, similar to what was done in the reduction presented in Section 5;
- Lines (10), (11), and (12) enforce that the roles x and y, which describe a grid structure encoding $\mathbb{Z} \times \mathbb{Z}$, are interpreted in $\mathcal{I}'$ as total functions with functional inverses;
- Lines (3), (5), and (9) guarantee that the concept Q is valid on the frame (in a modal logic sense) underlying $\mathcal{I}'$. Together with the previous item, this implies that, in $\mathcal{I}'$, the $x\circ y$-successor of any relevant object coincides with the $y \circ x$-successor;
- Lines (13) to (14) describe the existence of a tiling on the grid structure in $\mathcal{I}'$ that satisfies the horizontal and vertical matching conditions.

In [2], we prove the following lemma which immediately yields Theorem 15.

Lemma 17. *The following statements are equivalent:*

1. *The domino system $\mathcal{D}$ has a solution.*
2. *$\neg C_{\mathcal{D}}(a)$ is not a consequence of applying S in $\mathcal{A}$.*
3. *the composite service $S, (\{\neg C_{\mathcal{D}}(a)\}, \emptyset, \emptyset)$ is not executable in $\mathcal{A}$.*

7. 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 by nominals.

This is only a first proposal for a formalism describing the functionality of Web services, which must be extended in several directions. First, 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. Second, 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. Third, 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 Lemma 17. Frank Wolter was partly supported by EPSRC project GR/S63182/01 and Ulrike Sattler was partly supported by EPSRC project GR/S63168/01.

8. REFERENCES

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

[2] F. Baader, M. Milicic, C. Lutz, U. Sattler, and F. Wolter. Integrating description logics and action formalisms for reasoning about web services. LTCS-Report LTCS-05-02, Chair for Automata Theory, Institute for Theoretical Computer Science, TU Dresden, Germany, 2005. Available from http://lat.inf.tu-dresden.de/research/reports.html.

[3] 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.

[4] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jörg Siekmann*, LNAI. Springer-Verlag.

[5] R. Berger. The undecidability of the dominoe problem. *Memoirs of the American Mathematical Society*, 66, 1966.

[6] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *AIJ*, 57(2-3):227–270, October 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. and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In Bernhard Nebel, editor, *Proc. of IJCAI'01*, pages 161–166. Morgan Kaufmann, 2001.

[9] A. Herzig. The PMA revisited. In *Proc. of KR-96*. Morgan Kaufmann, 1996.

[10] I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proc. of KR98*, pages 636–647, 1998.

[11] I. Horrocks and P.F. Patel-Schneider. The generation of DAML+OIL. In *Proc. of DL2001*, number 49 in CEUR-WS (http://ceur-ws.org/), pages 30–35, 2001.

[12] Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4):345–357, 2004.

[13] 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.

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

[15] H.J. Levesque, R. Reiter, Y. 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.

[16] V. Lifschitz. Frames in the space of situations. *AIJ*, 46:365–376, 1990.

[17] F. Lin. Embracing causality in specifying the indeterminate effects of actions. In *Proc. of AAAI-96*, pages 670–676, Portland, OR, August 1996. MIT Press.

[18] S. McIlraith, T.C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[19] S. McIlraith and T.C. Son. Adapting golog for composition of semantic web services. In *Proc. of KR-02*, pages 482–496, 2002. Morgan Kaufmann.

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

[21] B. Nebel. Terminological reasoning is inherently intractable. *AIJ*, 43:235–249, 1990.

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

[23] 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, 2000.

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

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

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

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

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

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

[30] 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.

[31] The OWL-S Coalition. OWL-S 1.1 draft release, 2004. http://www.daml.org/services/owl-s/1.1/.

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

[33] M. Thielscher. Introduction to the Fluent Calculus. *ETAI*, 2(3–4):179–192, 1998.

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

[35] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *JAIR*, 12:199–217, 2000.

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

[37] M. Winslett. Epistemic aspects of databases. In *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 4 "Epistemic and Temporal Reasoning"*, pages 133–174. Oxford University Press, 1995.

[38] M. Winslett. Reasoning about action using a possible models approach. In *Proc. of AAAI-88*, pages 89–93, Saint Paul, MN, 1988.

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