Reasoning about Actions using Description Logics with general TBoxes (Revised Version)

Hongkai Liu[1], Carsten Lutz[1], Maja Miličić[1], Frank Wolter[2]

[1] Institut für Theoretische Informatik
TU Dresden, Germany
lastname@tcs.inf.tu-dresden.de
[2] Department of Computer Science
University of Liverpool, UK
frank@csc.liv.ac.uk

Abstract. Action formalisms based on description logics (DLs) have recently been introduced as decidable fragments of well-established action theories such as the Situation Calculus and the Fluent Calculus. However, existing DL action formalisms fail to include general TBoxes, which are the standard tool for formalising ontologies in modern description logics. We define a DL action formalism that admits general TBoxes, propose an approach to addressing the ramification problem that is introduced in this way, and perform a detailed investigation of the decidability and computational complexity of reasoning in our formalism.

1 Introduction

Action theories such as the Situation Calculus (SitCalc) and the Fluent Calculus aim at describing actions in a semantically adequate way [10, 12]. They are usually formulated in first- or higher-order logic and do not admit decidable reasoning. For reasoning about actions in practical applications, such theories are thus not directly suited. There are two obvious ways around this problem: the first one is to accept undecidability and replace reasoning by programming. This route is taken by the inventors of action-oriented programming languages such as Golog [5] and Flux [13], whose semantics is based on the SitCalc and Fluent Calculus, respectively. The second one is to try to identify fragments of action theories such as SitCalc that are sufficiently expressive to be useful in applications, but nevertheless admit decidable reasoning. For example, a simple such fragment is obtained by allowing only propositional logic for describing the state of the world and pre- and post-conditions of actions. A much more expressive formalism was identified in our recent paper [2], where we define action formalisms that are based on description logics (DLs) [3]. More precisely, we use DL ABoxes to describe the state of the world and pre- and post-conditions of actions and prove that reasoning in the resulting formalism is decidable [2]. We also show in [2] that, in this way, we actually get a decidable fragment of SitCalc.

In description logic, TBoxes are used as an ontology formalism, i.e., to define concepts and describe relations between them. For example, a TBox may describe relevant concepts from the domain of universities such as lecturers, students, courses, and libraries. From the reasoning about actions perspective, TBoxes correspond to state constraints. For example, a TBox for the university domain could state that every student that is registered for a course has access to a university library. If we execute an action that registers the student Dirk for a computer science course, then after the action Dirk should also have access to a university library to comply with the state constraint imposed by the TBox. Thus, general TBoxes as state constraints induce a ramificiation problem which we henceforth call the *TBox ramification problem*.

Regarding TBoxes, the DL action formalism defined in [2] has two major limitations: first, we only admit *acyclic TBoxes* which are a much more lightweight ontology formalism than the *general TBoxes* that can be found in all state-of-the-art DL reasoners [17]. For example, the DL formulation of the above ontology statement regarding access to libraries requires a general concept inclusion (GCIs) as offered by general TBoxes. Second, we allow only concept names (but no complex concepts) in post-conditions and additionally stipulate that these concept names are *not* defined in the TBox. In the present paper, we present an approach to overcoming these limitations while retaining decidability of the most important reasoning tasks. In particular, we show how to incorporate general TBoxes into DL action formalisms. This implies dropping the second restriction as well since there is no clear notion of a concept name "being defined" in a general TBox.

The main reason for adopting the mentioned restrictions in [2] was that they disarm the TBox ramification problem. Attempts to *automatically* solve the TBox ramificiation problem, e.g. by adopting a Winslett-style PMA semantics [16], lead to semantic and computational problems: we show in [2] that counter-intuitive results and undecidability of reasoning are the consequence of adopting such a semantics. Since there appears to be no general automated solution to the TBox ramification problem other than resorting to very inexpressive DLs [4], we propose to leave it to the designer of an action description to fine-tune the ramifications of the action. This is similar to the approach taken in the SitCalc and the Fluent Calculus to address the ramification problem. There, the designer of an action description can control the ramifications of the action by specifying causal relationships between predicates [6, 11]. While causality appears to be a satisfactory approach for addressing the ramification problem that is induced by Boolean state constraints, it seems not powerful enough for attacking the ramifications introduced by general TBoxes, which usually involve complex quantification patterns. We therefore advocate a different approach: when describing an action, the user can specify the predicates that can change through the execution of the action, as well as those that cannot change. To allow an adequate fine-tuning of ramifications, we admit complex statements about the change of predicates such as "the concept name A can change from positive to negative only at the individual a, and from negative to positive only where the complex concept C was satisfied before the action was executed".

Name	Syntax	Semantics
inverse role	r^-	$(r^\mathcal{I})^{-1}$
nominal	$\{a\}$	$\{a^\mathcal{I}\}$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
at-least restriction	$(\geqslant n\ r\ C)$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in C^\mathcal{I} \mid (x,y) \in r^\mathcal{I}\} \geq n\}$
at-most restriction	$(\leqslant n\ r\ C)$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in C^\mathcal{I} \mid (x,y) \in r^\mathcal{I}\} \leq n\}$

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

The family of action formalisms introduced in this paper can be parameterised with any description logic. We show that, for many standard DLs, the reasoning problems *executability* and *projection* in the corresponding action formalism are decidable. We also pinpoint the exact computational complexity of these reasoning problems. As a rule of thumb, our results show that reasoning in the action formalism instantiated with a description logic $\mathcal{L}$ is of the same complexity as standard reasoning in $\mathcal{L}$ extended with nominals (which correspond to first-order constants [1]). For fine-tuning the ramifications, consistency of actions is an important property. We introduce two notions of consistency (weak and strong) and show that weak consistency is of the same complexity as deciding projection while strong consistency is undecidable even when the action formalism is instantiated with the basic DL $\mathcal{ALC}$. Details regarding the technical results can be found in the report [7].

2 Description Logics

In DLs, *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 (possibly) a set $\mathsf{N_I}$ of *individual names*. In this section, we introduce the DL $\mathcal{ALCQIO}$, whose concepts are formed using the constructors shown in Figure 1. There, the inverse constructor is the only role constructor, whereas the remaining six constructors are concept constructors. In Figure 1 and throughout this paper, we use $\#S$ to denote the cardinality of a set S, a and b to denote individual names, r and s to denote roles (i.e., role names and inverses thereof), A, B to denote concept names, and C, D to denote (possibly complex) concepts. As usual, we use $\top$ as abbreviation for an arbitrary (but fixed) propositional tautology, $\bot$ for $\neg\top$, $\rightarrow$ and $\leftrightarrow$ for the usual Boolean abbreviations, $\exists r.C$ (*existential restriction*) for $(\geqslant 1\ r\ C)$, and $\forall r.C$ (*universal restriction*) for $(\leqslant 0\ r\ \neg C)$.

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 its sublanguages in a simple way.

The semantics of $\mathcal{ALCQIO}$-concepts is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ 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 $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, and each individual name $a \in \mathsf{N_I}$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to inverse roles and arbitrary concepts is inductively defined as shown in the third column of Figure 1.

A *general concept inclusion axiom (GCI)* is an expression of the form $C \sqsubseteq D$, where C and D are concepts. A *(general) TBox* $\mathcal{T}$ is a finite set of GCIs. An ABox is a finite set of *concept assertions* $C(a)$ and *role assertions* $r(a, b)$ and $\neg r(a, b)$ (where r may be an inverse role). An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, a concept assertion $C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$, a role assertion $r(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, and a role assertion $\neg r(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$. We denote satisfaction of a GCI $C \sqsubseteq D$ by an interpretation $\mathcal{I}$ with $\mathcal{I} \models C \sqsubseteq D$, and similar for ABox assertions. An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ (written $\mathcal{I} \models \mathcal{T}$) iff it satisfies all GCIs in $\mathcal{T}$. It is a *model* of an ABox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) iff it satisfies all assertions in $\mathcal{A}$.

A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ iff $C^{\mathcal{I}} \neq \emptyset$ for some model $\mathcal{I}$ of $\mathcal{T}$. An ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ iff $\mathcal{A}$ and $\mathcal{T}$ have a common model.

3 Describing Actions

The action formalism proposed in this paper is not restricted to a particular DL. However, for our complexity results we consider the DL $\mathcal{ALCQIO}$ and its sublogics. In the following, we use $\mathcal{LO}$ to denote the result of extending the DL $\mathcal{L}$ with nominals. A *concept literal* is a concept name or the negation thereof, and a *role literal* is defined analogously.

Definition 1 (Action). *Let $\mathcal{L}$ be a description logic. An $\mathcal{L}$-*action *$\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ consists of*

- *a finite set* pre *of $\mathcal{L}$ ABox assertions, the* pre-conditions*;*
- *the* occlusion pattern occ *which is a set of mappings $\{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ indexed by $\mathcal{L}$ ABox assertions $\varphi_1, \ldots, \varphi_n$ such that each occ_{φ_i} assigns*
  - *to every concept literal B an $\mathcal{LO}$-concept $\mathsf{occ}_{\varphi_i}(B)$,*
  - *to every role literal s a finite set $\mathsf{occ}_{\varphi_i}(s)$ of pairs of $\mathcal{LO}$-concepts.*
- *a finite set* post *of* conditional post-conditions *of the form φ/ψ, where φ and ψ are $\mathcal{L}$ ABox assertions.*

Intuitively, the pre-conditions specify under which conditions the action is applicable. A post-condition φ/ψ says that, if φ is true before executing the action, then ψ should be true afterwards. The purpose of the occlusion patterns is to control ramifications: they provide a description of where concept and role names may change during the execution of an action. More precisely, suppose

$\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ and $\varphi_{i_1}, \ldots, \varphi_{i_m}$ are the assertions which are true before the action was executed. If A is a concept name, then instances of the concept

$$\mathsf{occ}_{\varphi_{i_1}}(A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(A)$$

may change from A to $\neg A$ during the execution of the action provided, but instances of $\neg(\mathsf{occ}_{\varphi_{i_1}}(A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(A))$ may not. Likewise, instances of

$$\mathsf{occ}_{\varphi_{i_1}}(\neg A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(\neg A)$$

may change from $\neg A$ to A. For role names, $(C, D) \in \mathsf{occ}_{\varphi_{i_k}}(r)$ means that if an instance of C was connected to an instance of D by the role r before the action, then this connection may vanish through the execution of the action, and similarly for the occlusion of negated role names.

Before giving more details on how occlusions relate to ramifications, we introduce the semantics of actions. In our formalism, interpretations correspond to the state of the world. Thus, actions transform interpretations into new interpretations. We first introduce a convenient abbreviation. For an action α with $\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$, an interpretation $\mathcal{I}$, a concept literal B, and a role literal s, we set

$$\mathsf{occ}(B)^{\mathcal{I}} := \bigcup_{\mathcal{I} \models \varphi_i} (\mathsf{occ}_{\varphi_i}(B))^{\mathcal{I}} \qquad \mathsf{occ}(s)^{\mathcal{I}} := \bigcup_{(C,D) \in \mathsf{occ}_{\varphi_i}(s), \mathcal{I} \models \varphi_i} (C^{\mathcal{I}} \times D^{\mathcal{I}}).$$

Now, the semantics of actions is defined as follows.

Definition 2 (Action semantics). *Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be an action and $\mathcal{I}, \mathcal{I}'$ interpretations sharing the same domain and interpretation of all individual names. We say that α* may transform $\mathcal{I}$ to $\mathcal{I}'$ *w.r.t. a TBox $\mathcal{T}$ ($\mathcal{I} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}'$) iff the following holds:*

- *$\mathcal{I}, \mathcal{I}'$ are models of $\mathcal{T}$;*
- *for all $\varphi/\psi \in \mathsf{post}$: $\mathcal{I} \models \varphi$ implies $\mathcal{I}' \models \psi$ (written $\mathcal{I}, \mathcal{I}' \models \mathsf{post}$);*
- *for each $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$, we have*

$$A^{\mathcal{I}} \setminus A^{\mathcal{I}'} \subseteq (\mathsf{occ}(A))^{\mathcal{I}} \qquad \neg A^{\mathcal{I}} \setminus \neg A^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg A))^{\mathcal{I}}$$
$$r^{\mathcal{I}} \setminus r^{\mathcal{I}'} \subseteq (\mathsf{occ}(r))^{\mathcal{I}} \qquad \neg r^{\mathcal{I}} \setminus \neg r^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg r))^{\mathcal{I}}$$

Let us reconsider the example from the introduction to explain how occlusions provide a way to control the ramifications induced by general TBoxes. The TBox $\mathcal{T}$ contains the following GCIs which say that everybody registered for a course has access to a university library, and that every university has a library:

$$\exists \mathsf{registered\_for}.\mathsf{Course} \sqsubseteq \exists \mathsf{access\_to}.\mathsf{Library}$$
$$\mathsf{University} \sqsubseteq \exists \mathsf{has\_facility}.\mathsf{Library}$$

The upper GCI cannot be expressed in terms of an acyclic TBox and is thus outside the scope of the formalism in [2]. The ABox $\mathcal{A}$ which describes the current state of the world (in an incomplete way) says that computer science is a course held at TU Dresden, SLUB is the library of TU Dresden, and Dirk is neither registered for a course nor has access to a library:

$$\begin{array}{lll} \mathsf{Course}(\mathsf{cs}) & \mathsf{held\_at}(\mathsf{cs},\mathsf{tud}) & \neg\exists\mathsf{registered\_for}.\mathsf{Course}(\mathsf{dirk}) \\ \mathsf{University}(\mathsf{tud}) & \mathsf{has\_facility}(\mathsf{tud},\mathsf{slub}) & \neg\exists\mathsf{access\_to}.\mathsf{Library}(\mathsf{dirk}) \\ \mathsf{Library}(\mathsf{slub}) & & \end{array}$$

The action

$$\alpha := (\emptyset, \mathsf{occ}, \{\mathsf{taut}/\mathsf{registered\_for}(\mathsf{dirk},\mathsf{cs})\})$$

describes the registration of Dirk for the computer science course. For simplicity, the set of pre-conditions is empty and taut is some ABox assertion that is trivially satisfied, say $\top(\mathsf{cs})$. To obtain occ, we may start by strictly following the law of inertia, i.e., requiring that the only changes are those that are explicitly stated in the post-condition. Thus, occ consists of just one mapping $\mathsf{occ}_{\mathsf{taut}}$ such that

$$\mathsf{occ}_{\mathsf{taut}}(\neg\mathsf{registered\_for}) := \{(\{\mathsf{dirk}\},\{\mathsf{cs}\})\}$$

and all concept and role literals except $\neg\mathsf{registered\_for}$ are mapped to $\bot$ and $\{(\bot,\bot)\}$, respectively. This achieves the desired effect that only the pair $(\mathsf{dirk},\mathsf{cs})$ can be added to "$\mathsf{registered\_for}$" and nothing else can be changed.

It is not hard to see that this attempt to specify occlusions for α is too strict. Intuitively, not allowing any changes is appropriate for Course, $\mathsf{Library}$, $\mathsf{University}$, $\mathsf{held\_at}$, $\mathsf{has\_facility}$ and their negations since the action should have no impact on these predicates. However, not letting $\neg\mathsf{access\_to}$ change leads to a problem with the ramifications induced by the TBox: as Dirk has no access to a library before the action and $\neg\mathsf{access\_to}$ is not allowed to change, he cannot have access to a library after execution of the action as required by the TBox. Thus, the action is inconsistent in the following sense: there is no model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ and model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}'$. To take care of the TBox ramifications and regain consistency, we can modify occ. One option is to set

$$\mathsf{occ}_{\mathsf{taut}}(\neg\mathsf{access\_to}) := \{(\{\mathsf{dirk}\},\mathsf{Library})\}$$

and thus allow Dirk to have access to a library after the action. Another option is to set

$$\mathsf{occ}_{\mathsf{taut}}(\neg\mathsf{access\_to}) := \{(\{\mathsf{dirk}\},\mathsf{slub})\}$$

which allows Dirk to have access to SLUB after the action, but not to any other library.

Two remarks regarding this example are in order. First, the occlusion occ consists only of a single mapping $\mathsf{occ}_{\mathsf{taut}}$. The reason for this is that there is only a single post-condition in the action. If we have different post-conditions φ/ψ and φ'/ψ such that φ and φ' are not equivalent, there will usually be different occlusion mappings (indexed with φ and φ') to deal with the ramifications that the TBox induces for these post-conditions. Second, the example explains the need for extending $\mathcal{L}$ to $\mathcal{LO}$ when describing occlusions (c.f. Definition 1): without nominals, we would not have been able to properly formulate the occlusions although all other parts of the example are formulated without using nominals (as a concept-forming operator).

As illustrated by the example, it is important for the action designer to decide consistency of actions to detect ramification problems that are not properly addressed by the occlusions. In the following, we propose two notions of consistency.

Definition 3 (Consistency). *Let* $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ *be an action and* $\mathcal{T}$ *a TBox. We say that*

– α *is* weakly consistent with $\mathcal{T}$ *iff there are models* $\mathcal{I}, \mathcal{I}'$ *of* $\mathcal{T}$ *such that* $\mathcal{I} \models \mathsf{pre}$ *and* $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$.
– α *is* strongly consistent with $\mathcal{T}$ *iff for all models* $\mathcal{I}$ *of* $\mathcal{T}$ *and* pre*, there is a model* $\mathcal{I}'$ *of* $\mathcal{T}$ *such that* $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$.

Intuitively, strong consistency is the most desirable form of consistency: if the preconditions of an action are satisfied by an interpretation $\mathcal{I}$, then the action can transform $\mathcal{I}$ into a new interpretation $\mathcal{I}'$. Unfortunately, strong consistency will turn out to be undecidable. For this reason we also introduce weak consistency, which is still sufficient to detect serious ramification problems. In the example above, the first attempt to define the occlusions results in an action that is not even weakly consistent. After each of the two possible modifictions, the action is strongly consistent. We will see later that weak consistency is decidable while strong consistency is not.

To check whether an action can be applied in a given situation, the user wants to know whether it is executable, i.e., whether all pre-conditions are satisfied in the states of the world considered possible. If the action is executable, he wants to know whether applying it achieves the desired effect, i.e., whether an assertion that he wants to make true really holds after executing the action. These two problems are called executability and projection [10, 2].

Definition 4 (Executability and projection). *Let* $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ *be an action,* $\mathcal{T}$ *a TBox, and* $\mathcal{A}$ *an ABox.*

– *Executability:* α *is* executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$ *iff* $\mathcal{I} \models \mathsf{pre}$ *for all models* $\mathcal{I}$ *of* $\mathcal{A}$ *and* $\mathcal{T}$*;*
– *Projection: The assertion* φ *is a* consequence of applying α in $\mathcal{A}$ w.r.t. $\mathcal{T}$ *iff for all models* $\mathcal{I}$ *of* $\mathcal{A}$ *and* $\mathcal{T}$ *and for all* $\mathcal{I}'$ *with* $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$*, we have* $\mathcal{I}' \models \varphi$.

To make sure that an action α can be successfully executed, α has to be both strongly consistent and executable: without strong consistency, it could be that although the action α is executable w.r.t. the ABox $\mathcal{A}$ describing the knowledge about the current state of the world, the actual state of the world $\mathcal{I}$ is such that there is no interpretation $\mathcal{I}'$ with $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$.

It is not difficult to see that the action formalism just introduced is a generalisation of the one introduced in [2] when composite actions are disallowed, for details see [7]. Clearly, executability can be polynomially reduced to *ABox consequence* which is defined as follows: given an ABox $\mathcal{A}$ and an assertion φ, decide whether $\mathcal{I}$ satisfies φ in all models $\mathcal{I}$ of $\mathcal{A}$. The complexity of this problem is extensively discussed in [2]. For example, it is NExpTime-complete for $\mathcal{ALCQIO}$ and ExpTime-complete for $\mathcal{ALC}$ extended with at most two of $\mathcal{Q}$, $\mathcal{I}$, and $\mathcal{O}$.

It can also be seen that (i) an action $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ is weakly consistent with a TBox $\mathcal{T}$ iff $\bot(a)$ is not a consequence of applying α in pre w.r.t. $\mathcal{T}$; (ii) φ is a consequence of applying $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff the action $(\mathcal{A} \cup \mathsf{pre}, \mathsf{occ}, \mathsf{post} \cup \{\top(a)/\neg\varphi\})$ is not weakly consistent with $\mathcal{T}$. Thus, weak consistency can be reduced to (non-)projection and vice versa and complexity results carry over from one to the other. In this paper, we will concentrate on projection.

4 Projection in ExpTime

We show that projection and weak consistency are ExpTime-complete for DL actions formulated in $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCI}$, $\mathcal{ALCIO}$. Thus, in these DLs reasoning about actions is not more difficult than the standard DL reasoning problems such as concept satisfiability and subsumption w.r.t. TBoxes. The complexity results established in this section are obtained by proving that projection in $\mathcal{ALCIO}$ is in ExpTime. We use a Pratt-style type elimination technique as first proposed in [8].

In the following, we assume that the set occ of occlusions of an action consists of only one mapping $\mathsf{occ}_{\mathsf{taut}}$, where taut is $\top(a)$. We will identify occ with the mapping $\mathsf{occ}_{\mathsf{taut}}$ and write $\mathsf{occ}(X)$ instead of $\mathsf{occ}_{\mathsf{taut}}(X)$. Proofs are easily extended to actions containing general occlusions, see [7].

Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be an action, $\mathcal{T}$ a TBox, $\mathcal{A}_0$ an ABox and φ_0 an assertion. We want to decide whether φ_0 is a consequence of applying α in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. In what follows, we call α, $\mathcal{T}$, $\mathcal{A}_0$ and φ_0 the *input*. W.l.o.g., we make the following assumptions:

- concepts used in the input are built only from the constructors $\{a\}$, $\neg$, $\sqcap$, and $\exists r.C$;
- φ_0 is of the form $\varphi_0 = C_0(a_0)$, where C_0 is a (complex) concept;
- $\mathcal{A}_0$ and α contain only concept assertions.

The last two assumptions can be made because every assertion $r(a, b)$ can be replaced with $(\exists r.\{b\})(a)$, and every $\neg r(a, b)$ with $(\neg\exists r.\{b\})(a)$.

Before we can describe the algorithm, we introduce a series of notions and abbreviations. With Sub, we denote the set of subconcepts of the concepts which occur in the input. With Ind, we denote the set of individual names used in the input, and set $\mathsf{Nom} := \{\{a\} \mid a \in \mathsf{Ind}\}$.

The algorithm for deciding projection checks for the existence of a counter-model witnessing that φ_0 is *not* a consequence of applying α in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$.

Such a countermodel consists of interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}_0$, $\mathcal{I} \Rightarrow^{\mathcal{T}} \mathcal{I}'$, and $\mathcal{I}' \not\models \varphi_0$. To distinguish the extension of concept and role names in $\mathcal{I}$ and $\mathcal{I}'$, we introduce concept names $A^{(0)}, A^{(1)}$ and role names $r^{(0)}, r^{(1)}$, for every concept name A and role name r used in the input. Here, the superscript $\cdot^{(0)}$ identifies $\mathcal{I}$ and the superscript $\cdot^{(1)}$ identifies $\mathcal{I}'$. For a concept $C \in \mathsf{Sub}$ that is not a concept name and $i \in \{0,1\}$, we use $C^{(i)}$ to denote the concept obtained by replacing all concept names A and role names r occurring in C by $A^{(i)}$ and $r^{(i)}$, respectively. We define the set of concepts Cl as:

$$\mathsf{Cl} = \{C^{(0)}, \neg C^{(0)}, C^{(1)}, \neg C^{(1)} \mid C \in \mathsf{Sub} \cup \mathsf{Nom}\}.$$

The notion of a type plays a central role in the projection algorithm to be devised.

Definition 5. *A set of concepts $t \subseteq \mathsf{Cl}$ is a* type *for Cl iff it satisfies the following conditions:*

- *for all $\neg D \in \mathsf{Cl}$: $\neg D \in t$ iff $D \notin t$;*
- *for all $D \sqcap E \in \mathsf{Cl}$: $D \sqcap E \in t$ iff $\{D, E\} \subseteq t$;*
- *for all $C \sqsubseteq D \in \mathcal{T}$, $C^{(0)} \in t$ implies $D^{(0)} \in t$ and $C^{(1)} \in t$ implies $D^{(1)} \in t$;*
- *for all concept names A, $\{A, \neg A'\} \subseteq t$ implies that $\mathsf{occ}(A) \in t$ and $\{\neg A, A'\} \subseteq t$ implies that $\mathsf{occ}(\neg A) \in t$.*

A type is anonymous *if it does not contain a nominal. Let $\mathfrak{T}_{\mathsf{ano}}$ be the set of all anonymous types.*

Intuitively, a type describes the concept memberships of a domain element in the interpretations $\mathcal{I}$ and $\mathcal{I}'$. Our algorithm starts with a set containing (almost) all types, then repeatedly eliminates those types that cannot be realized in a countermodel witnessing that φ_0 is not a consequence of applying α in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$, and finally checks whether the surviving types give rise to such a countermodel. The picture is slightly complicated by the presence of ABoxes and nominals. These are treated via core type sets to be introduced next.

Definition 6. *$\mathfrak{T}_S$ is a* core type set *iff $\mathfrak{T}_S$ is a minimal set of types such that, for all $a \in \mathsf{Ind}$, there is a $t \in \mathfrak{T}_S$ with $\{a\} \in \mathfrak{T}_S$.*
A core type set $\mathfrak{T}_S$ is called proper *if the following conditions are satisfied:*

1. *for all $C(a) \in \mathcal{A}_0$, $\{a\} \in t \in \mathfrak{T}_S$ implies $C^{(0)} \in t$;*
2. *for all $C(a)/D(b) \in \mathsf{post}$: if there is a $t \in \mathfrak{T}_S$ with $\{\{a\}, C^{(0)}\} \subseteq t$ then there is a $t' \in \mathfrak{T}_S$ with $\{\{b\}, D^{(1)}\} \subseteq t'$.*

Intuitively, a core type set carries information about the "named" part of the interpretations $\mathcal{I}_0$ and $\mathcal{I}_1$, where the named part of an interpretation consists of those domain elements that are identified by nominals. Let m be the size of the input. It is not difficult to check that the number of core type sets is exponential in m. Also, checking whether a core type set is proper can be done in linear time.

The following definition specifies the conditions under which a type is eliminated. For a role name r, we set $\mathsf{occ}(r^-) := \{(Y, X) \mid (X, Y) \in \mathsf{occ}(r)\}$, and analogously for $\mathsf{occ}(\neg r^-)$. For role names r, we set $\mathsf{Inv}(r) := r^-$ and $\mathsf{Inv}(r^-) := r$.

```
$\mathcal{ALCIO}$-elim$(\mathcal{A}_0, \mathcal{T}, \alpha, \varphi_0)$
    for all proper core type sets $\mathfrak{T}_S$ do
        $i := 0$;
        $\mathfrak{T}_0 := \mathfrak{T}_S \cup \mathfrak{T}_{\mathsf{ano}}$
        repeat
            $\mathfrak{T}_{i+1} := \{t \in \mathfrak{T}_i \mid t$ is good in $\mathfrak{T}_i\}$;
            $i := i + 1$;
        until $\mathfrak{T}_i = \mathfrak{T}_{i-1}$;
        if $\mathfrak{T}_S \subseteq \mathfrak{T}_i$ and there is a $t \in \mathfrak{T}_i$ with $\{\{a_0\}, \neg C_0^{(1)}\} \subseteq t$ then
            return false
        endif
    endfor
    return true
```

Fig. 2. The type elimination algorithm.

Definition 7. *Let $\mathfrak{T}$ be a set of types for* Cl. *Then a type $t \in \mathfrak{T}$ is* good in $\mathfrak{T}$ *if for all $(\exists r.C)^{(i)} \in t$, there exists a type $t' \in \mathfrak{T}$ a set $\rho \subseteq \{0, 1\}$ such that $i \in \rho_\ell$ and the following are satisfied, for $i \in \{0, 1\}$:*

- *if $(\neg\exists r.C)^{(i)} \in t$ and $i \in \rho_j$, then $(\neg C)^{(i)} \in t'$;*
- *if $\neg\exists(\mathsf{Inv}(r).C)^{(i)} \in t'$ and $i \in \rho_j$, then $(\neg C)^{(i)} \in t$;*
- *if $0 \in \rho_j$ and $1 \notin \rho_j$ then there exists a pair $(X, Y) \in \mathsf{occ}(r)$ such that $X^{(0)} \in t$ and $Y^{(0)} \in t'$,*
- *if $0 \notin \rho_j$ and $1 \in \rho_j$ then there exists a pair $(X, Y) \in \mathsf{occ}(\neg r)$ such that $X^{(0)} \in t$ and $Y^{(0)} \in t'$.*

Intuitively, the above definition checks whether there can be any instances of t in an interpretation in which all domain elements have a type in $\mathfrak{T}$. More precisely, t is the type of a domain element that is needed to satisfy the existential restriction $(\exists r.C)^{(i)}$. The set ρ determines the extension of the role r: if $0 \in \rho$, then the instance of t' is reachable via the role r from the instance of t in $\mathcal{I}$, and similarly for $1 \in \rho_j$ and $\mathcal{I}'$.

The type elimination algorithm is given in a pseudo-code notation in Figure 2, where C_0 is the concept from the ABox assertion $\varphi_0 = C_0(a_0)$. A proof of the following lemma can be found in [7].

Lemma 1. $\mathcal{ALCIO}$-elim$(\mathcal{A}_0, \mathcal{T}, \alpha, \varphi_0)$ *returns true iff φ_0 is a consequence of applying α in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$.*

The algorithm runs in exponential time: first, we have already argued that there are only exponentially many core type sets. Second, the number of elimination rounds is bounded by the number of types, of which there are only exponentially many. And third, it is easily seen that it can be checked in exponential time whether a type is good in a given type set. Since concept satisfiability w.r.t. TBoxes is ExpTime-hard in $\mathcal{ALC}$ [3] and concept satisfiability can be reduced to (non-)projection [2], we obtain the following result.

Theorem 1. *Projection and weak consistency are* ExpTime*-complete in $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCI}$, and $\mathcal{ALCIO}$.*

It is not too difficult to adapt the algorithm given in this section to the DL $\mathcal{ALCQO}$. Therefore, we conjecture that the reasoning problems from Theorem 1 are also ExpTime-complete for $\mathcal{ALCQ}$ and $\mathcal{ALCQO}$.

5 $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$: Beyond ExpTime

In the previous section, we have identified a number of DLs for which both reasoning about actions and standard DL reasoning are ExpTime-complete. Another candidate for a DL with such a behaviour is $\mathcal{ALCQI}$, in which satisfiability and subsumption are ExpTime-complete as well [15]. However, it follows from results in [2] that projection in $\mathcal{ALCQI}$ is co-NExpTime-hard. In the following, we show that it is in fact co-NExpTime-complete, and that the same holds for the DL $\mathcal{ALCQIO}$. Note that, for the latter DL, also concept subsumption is co-NExpTime-complete.

It is shown in [7] that Lemma 8 of [2] implies the following.

Theorem 2. *Projection (weak consistency) in $\mathcal{ALCQI}$ is co-*NExpTime*-hard (*NExpTime*-hard) even if occlusions for role literals are restricted to $(\bot, \bot)$ and occlusions of concept literals are restricted to $\bot$ and nominals.*

In the following, we establish a co-NExpTime upper bound for projection in $\mathcal{ALCQIO}$ (and thus also $\mathcal{ALCQI}$). The proof proceeds by reducing projection in $\mathcal{ALCQIO}$ to ABox (in)consistency in $\mathcal{ALCQIO}^{\neg,\cup,\cap}$, the extension of $\mathcal{ALCQIO}$ with the Boolean role constructors complement, union, and intersection.

Let α be an action, $\mathcal{T}$ a TBox, $\mathcal{A}_0$ an ABox and φ_0 an assertion. We are interested in deciding whether φ_0 is a consequence of applying α in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. We use the same notions and abbreviations as in Section 4. As in that section, we also assume that φ_0 is of the form $C_0(a_0)$ and that occlusions are of a restricted form.

The idea for the following reduction is to define an ABox $\mathcal{A}_{\mathsf{red}}$ and a TBox $\mathcal{T}_{\mathsf{red}}$ such that each model of $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ encodes interpretations $\mathcal{I}$ and $\mathcal{I}'$ with $\mathcal{I} \models \mathcal{A}_0$ and $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$, and $\mathcal{I}' \not\models \varphi_0$. The encoding of the two interpretations $\mathcal{I}$ and $\mathcal{I}'$ into a single model of $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ is similar to what was done in the previous section: we introduce superscripted version of all concept and role names to distinguish the extension in $\mathcal{I}$ from that in $\mathcal{I}'$. We start by assembling the reduction ABox $\mathcal{A}_{\mathsf{red}}$. First, introduce abbreviations:

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

where U denotes the universal role, i.e. $r \cup \neg r$ for some $r \in \mathsf{N_R}$. Now we can define the components of $\mathcal{A}_{\mathsf{red}}$ that take care of post-condition satisfaction. We define:

$$\mathcal{A}_{\mathsf{post}} := \{\big(\mathsf{p}_0(\varphi) \rightarrow \mathsf{p}_1(\psi)\big)(a_0) \mid \varphi/\psi \in \mathsf{post}\},$$

We assemble $\mathcal{A}_{\mathsf{red}}$ as $\mathcal{A}_{\mathsf{red}} := \mathcal{A}_0 \cup \mathcal{A}_{\mathsf{post}}$ and continue by defining the components of the TBox $\mathcal{T}_{\mathsf{red}}$. The first component ensures that auxiliary role names $r_{\mathsf{Dom}(C)}$ and $r_{\mathsf{Ran}(D)}$ are interpreted as $C \times \top$ and $\top \times D$, respectively. For every $(C, D) \in \mathsf{occ}(s)$ for some role literal s from the input, the TBox $\mathcal{T}_{\mathsf{aux}}$ contains the following GCIs :

$$\begin{array}{rclrcl} C^{(0)} & \sqsubseteq & \forall \neg r_{\mathsf{Dom}(C)}.\bot & \qquad \top & \sqsubseteq & \forall r_{\mathsf{Ran}(D)}.D^{(0)} \\ \neg C^{(0)} & \sqsubseteq & \forall r_{\mathsf{Dom}(C)}.\bot & \qquad \top & \sqsubseteq & \forall \neg r_{\mathsf{Ran}(D)}.\neg D^{(0)} \end{array}$$

The following component describes the behaviour of concept names and role names in parts of the domain where they are *not* allowed to vary. The TBox $\mathcal{T}_{\mathsf{fix}}$ contains for every concept name A in the input,

$$\neg\mathsf{occ}(A)^{(0)} \sqcap A^{(0)} \sqsubseteq A^{(1)}$$
$$\neg\mathsf{occ}(\neg A)^{(0)} \sqcap \neg A^{(0)} \sqsubseteq \neg A^{(1)}$$

and for every role name r in the input,

$$\top \sqsubseteq \forall \neg\Big(\bigcup_{(C,D) \in \mathsf{occ}(r)} (r_{\mathsf{Dom}(C)} \cap r_{\mathsf{Ran}(D)}) \Big) \cap (r^{(0)} \cap \neg r^{(1)}).\bot$$
$$\top \sqsubseteq \forall \neg\Big(\bigcup_{(C,D) \in \mathsf{occ}(\neg r)} (r_{\mathsf{Dom}(C)} \cap r_{\mathsf{Ran}(D)}) \Big) \cap (\neg r^{(0)} \cap r^{(1)}).\bot$$

Finally, we can construct $\mathcal{T}_{\mathsf{red}}$ as

$$\mathcal{T}_{\mathsf{red}} := \mathcal{T}_{\mathsf{aux}} \cup \mathcal{T}_{\mathsf{fix}} \cup \{C^{(i)} \sqsubseteq D^{(i)} \mid i \in \{0,1\} \land C \sqsubseteq D \in \mathcal{T}\}.$$

The last two components of $\mathcal{T}_{\mathsf{red}}$ ensure that $\mathcal{I}$ and $\mathcal{I}'$ are models of the input TBox $\mathcal{T}$. It is not difficult to show that the following holds:

Lemma 2. *$C_0(a_0)$ is a consequence of applying α in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}} \cup \{\neg C_0^{(1)}(a_0)\}$ is inconsistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

Since $\mathcal{ALCQIO}^{\cup,\cap,\neg}$ is a fragment of $\mathcal{C}^2$ (the 2-variable fragment of first-order logic with counting), we have that ABox inconsistency in $\mathcal{ALCQIO}^{\cup,\cap,\neg}$ is in co-NExpTime, even if numbers are coded in binary [9]. Since $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ are polynomial in the size of the input ABox $\mathcal{A}_0$, TBox $\mathcal{T}$, and action α, Lemma 2 gives us the same upper complexity bound for projection in $\mathcal{ALCQIO}$ and $\mathcal{ALCQI}$. Theorem 2 implies that this is a tight complexity bound:

Theorem 3. *In $\mathcal{ALCQIO}$, projection is co-*NExpTime*-complete and weak consistency is* NExpTime*-complete.*

6 Undecidability of Strong Consistency

We show that strong consistency is undecidable already in $\mathcal{ALC}$. The proof consists of a reduction of the undecidable *semantic consequence problem* from modal logic. Before formulating the DL version of this problem, we need some preliminaries. We use $\mathcal{ALC}$ concepts with only one fixed role name r, which we call $\mathcal{ALC}_r$-concepts. Accordingly, we also assume that interpretations interpret only concept names and the role name r. A *frame* is a structure $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ where $\Delta^{\mathcal{F}}$ is a non-empty set and $r^{\mathcal{F}} \subseteq \Delta^{\mathcal{F}} \times \Delta^{\mathcal{F}}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is *based* on a frame $\mathcal{F}$ iff $\Delta^{\mathcal{I}} = \Delta^{\mathcal{F}}$ and $r^{\mathcal{I}} = r^{\mathcal{F}}$. We say that a concept C is *valid* on $\mathcal{F}$ (written $\mathcal{F} \models C$) iff $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ based on $\mathcal{F}$.

Definition 8 (Semantic consequence problem). *Let D and E be $\mathcal{ALC}_r$-concepts. We say that E is a* semantic consequence *of D iff for every frame $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ such that $\mathcal{F} \models D$, it holds that $\mathcal{F} \models E$.*

In [14], it is proved that for $\mathcal{ALC}_r$-concepts D and E, the problem "Is E a semantic consequence of D?" is undecidable. We now show that the semantic consequence problem can be reduced to strong consistency. For $\mathcal{ALC}_r$-concepts D and E, we define the action $\alpha_D = (\mathsf{pre}, \{\mathsf{occ}_{\mathsf{taut}}\}, \mathsf{post})$ where $\mathsf{pre} := \{\neg E(a)\}$, $\mathsf{post} := \{\top(a)/(\exists u.\neg D)(a)\}$ (u a role name), and $\mathsf{occ}_{\mathsf{taut}}$ maps r and $\neg r$ to $\{(\bot, \bot)\}$, all other role literals to $\{(\top, \top)\}$, and all concept literals to $\top$. Then the following holds.

Lemma 3. *The action α_D is strongly consistent with the empty TBox iff E is a semantic consequence of D.*

Proof. "$\Rightarrow$" We show the contraposition. Assume that E is not a semantic consequence of D. Then there exists a frame $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ such that $\mathcal{F} \models D$ and there is an interpretation $\mathcal{I}$ based on $\mathcal{F}$ such that $E^{\mathcal{I}} \neq \Delta^{\mathcal{I}}$. We take $\mathcal{I}$ based on $\mathcal{F}$ such that $a^{\mathcal{I}} \notin E^{\mathcal{I}}$, thus $\mathcal{I} \models \mathsf{pre}$. But every $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow^{\emptyset}_{\alpha_D} \mathcal{I}'$ must be based on $\mathcal{F}$ (since $r^{\mathcal{I}'} = r^{\mathcal{I}} = r^{\mathcal{F}}$) and must satisfy $D^{\mathcal{I}'} \neq \Delta^{\mathcal{I}'}$ (by the post-condition of α). Since $\mathcal{F} \models D$, there is no such $\mathcal{I}'$. Thus, α_D is not strongly consistent with the empty TBox.

"$\Leftarrow$" Assume that E is a semantic consequence of D. Let $\mathcal{I} \models \mathsf{pre}$. By definition of pre, we have that $a^{\mathcal{I}} \notin E^{\mathcal{I}}$, and thus $\mathcal{I}$ is not based on a frame $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ validating E. Since E is a semantic consequence of D, $\mathcal{F}$ is not validating D either, and there is an interpretation $\mathcal{I}'$ based on $\mathcal{F}$ such that $D^{\mathcal{I}'} \neq \Delta^{\mathcal{I}'}$. Take $y \in \Delta^{\mathcal{I}'}$ such that $y \notin D^{\mathcal{I}'}$. Since D is an $\mathcal{ALC}_r$- concept, we may assume that $u^{\mathcal{I}'} = \{(a^{\mathcal{I}'}, y)\}$. Obviously, we have that $\mathcal{I} \Rightarrow^{\emptyset}_{\alpha_D} \mathcal{I}'$, and, consequently, α_D is strongly consistent with the empty TBox.

As an immediate consequence, we obtain the following theorem.

Theorem 4. *Strong consistency of $\mathcal{ALC}$-actions is undecidable, even with the empty TBox.*

7 Discussion

We have introduced an action formalism based on description logics that admits general TBoxes and complex post-conditions. To deal with ramifications induced by general TBoxes, the formalism includes powerful occlusion patterns that can be used to fine-tune the ramifications. Most important reasoning tasks in our formalism turn out to be decidable.

Our only negative result concerns the undecidability of strong consistency. To discuss the impact of this result, let us briefly review the relevance of strong consistency for the action designer and for the user of the action (the person who applies the action).

For the action *designer*, an algorithm for checking strong consistency would be useful for fine-tuning the ramifications of his action. However, it is worth noting that deciding strong consistency could not replace manual inspection of the ramifications. For example, occluding all concept names with $\top$ and all role names with $\{(\top, \top)\}$ usually ensures strong consistency but does not lead to an intuitive behaviour of the action. With weak consistency, we offer at least some automatic support to the action designer for detecting ramification problems.

For the *user* of the action, strong consistency is required to ensure that the execution of an action whose preconditions are satisfied will not fail. If the action is such that failure cannot be tolerated (because executing the action is expensive, dangerous, etc), strong consistency is thus indispensible and should already be guaranteed by the action designer. However, for other actions it is conceivable that an execution failure does not have any negative effects. If this is the case, the action user only needs to check that the action is executable, and strong consistency is not strictly required.

Future work will include developing practical decision procedures. A first step is carried out in [7], where we show that in the following special (but natural) case, projection can be reduced to standard reasoning problems in DLs that are implemented in DL reasoners such as RACER and FaCT++: (i) role occlusions in actions are given by $\mathsf{occ}_{\mathsf{taut}}$; (ii) $\mathsf{occ}_{\mathsf{taut}}(r) = \mathsf{occ}_{\mathsf{taut}}(\neg r)$; and (iii) concepts used in $\mathsf{occ}_{\mathsf{taut}}(r)$ are nominals, $\top$, or $\bot$.

Acknowledgements. We would like to thank Giuseppe De Giacomo for ideas and discussions. The first author was supported by the DFG Project BA1122/10-2. The second author was supported by the EU funded IST-2005-7603 FET Project Thinking Ontologies (TONES). The third author was supported by the DFG Graduiertenkolleg 334. The fourth author was partially supported by UK EPSRC grant no. GR/S63182/01.

References

1. C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. *Proc. of CSL-99*, number 1683 in LNCS, pages 307–321. Springer, 1999.
2. F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proc. of AAAI-05*, AAAI Press, 2005.
3. F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications.* Cambridge University Press, 2003.

4. G. de Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the update of description logic ontologies at the instance level. *Proc. of AAAI-06*, AAAI Press, 2006.
5. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.
6. F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI-95*, pages 1985–1991, Morgan Kaufmann, 1995.
7. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Description logic actions with general TBoxes: a pragmatic approach. LTCS-Report 06-03, TU Dresden, Germany, 2006. See http://lat.inf.tu-dresden.de/research/reports.html.
8. V. R. Pratt. Models of program logics. In *Proc. of the Twentieth FoCS*, San Juan, Puerto Rico, 1979.
9. I. Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.
10. R. Reiter. *Knowledge in Action*. MIT Press, 2001.
11. M. Thielscher. Ramification and causality. *Artificial Intelligence Journal*, 89(1–2):317–364, 1997.
12. M. Thielscher. Introduction to the Fluent Calculus. *Electronic Transactions on Artificial Intelligence*, 2(3–4):179–192, 1998.
13. M. Thielscher. FLUX: A logic programming method for reasoning agents. *TPLP*, 5(4-5):533–565, 2005.
14. S. K. Thomason. The logical consequence relation of propositional tense logic. *Z. Math. Logik Grundl. Math.*, 21:29–40, 1975.
15. S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *JAIR*, 12:199–217, 2000.
16. M. Winslett. Reasoning about action using a possible models approach. In *AAAI-88*, pages 89–93, 1988.
17. A list of DL reasoners: http://www.cs.man.ac.uk/∼sattler/reasoners.html