DL Actions with GCIs: a Pragmatic Approach

H. Liu[1], C. Lutz[1], M. Miličić[1], F. Wolter[2]

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

[2]Department of CS
Univ. of Liverpool, UK
frank@csc.liv.ac.uk

Abstract

We recently proposed action formalisms based on description logics (DLs) as decidable fragments of well-established action theories such as the Situation Calculus and the Fluent Calculus. One short-coming of our initial proposal is that the considered formalisms admit only acyclic TBoxes, but not GCIs. In this paper, we define DL action formalisms that admits GCIs, propose a pragmatic approach to addressing the ramification problem that is introduced in this way, show that our formalim is decidable and investigate its computational complexity.

1 Introduction

Action theories such as the Situation Calculus (SitCalc) and the Fluent Calculus aim at describing actions in a semantically adequate way [8, 10]. 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 [3] and Flux [11], 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 [1], where we define action formalisms that are based on description logics (DLs). 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 [1]. We also show in [1] that, in this way, we actually get a decidable fragment of SitCalc. However, the DL action formalism defined in [1] has two major limitations: first, we only admit acyclic TBoxes, but no general TBoxes involving GCIs. And 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 a pragmatic approach to overcoming these limitations while retaining decidability of reasoning. In particular, we show how to incorporate general TBoxes into DL action formalisms. Since there is no clear notion of a concept name "being defined" in a general TBox, we also drop the second restriction and admit arbitrary concepts in post-conditions.

The main reason for adopting the mentioned restrictions in [1] was that they disarm the frame and ramification problem, which pose major difficulties in reasoning about actions. When admitting general TBoxes, in particular the ramification problem becomes a serious issue. Attempts to *automatically* solve this problem, e.g. by adopting a Winslett-style PMA semantics [14], lead to semantic and computational problems: we show in [1] 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 ramification problem introduced by general TBoxes, we take a rather pragmatic approach and leave it to the designer of an action description to fine-tune the ramifications of the action. This is similar to what is done 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 [4, 9]. While causality appears to be a satisfactory approach for addressing the ramification problem in the case of propositional state constraints (which correspond to a TBox formulated in propositional logic), it seems not powerful enough for attacking the ramifications introduced by general TBoxes, which may involve complex quantification patterns. We therefore adopt a different strategy for controlling ramifications: when describing an action, the user can specify the predicates that can change by executing the action, as well as those that cannot change. To allow an adequate fine-tuning of ramifications, we admit rather complex statements about the change of predicates such as "the concept name A can change from positive to negative only at the indidivual a, and from negative to positive only where the complex concept C was satisfied before the action was executed".

The family of action formalisms introduced in this paper can be parameterized 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 for several members of the $\mathcal{ALCQIO}$ family of DLs. 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 subsumption in $\mathcal{L}$ extended with nominals. For fine-tuning ramifications, deciding the consistency of actions is of prime importance. We introduce two notions of consistency (weak and strong) and show that one of them is of the same complexity as deciding projection while the other one is undecidable even when the action formalism is instantiated with $\mathcal{ALC}$.

2 Describing Actions

The action formalism proposed in this paper is an extension of the one from [1] and is not restricted to a particular DL. However, for our complexity results we consider the DL $\mathcal{ALCQIO}$ and its fragments. We refrain from introducing the syntax and semantics of $\mathcal{ALCQIO}$ in full detail, referring e.g. to [2], and only give a few central definitions. A *concept literal* is a concept name or the negation thereof, and a *role literal* is a role name or the negation thereof. An *ABox assertion* is of the form $C(a)$ or $r(a, b)$, where a, b are individual names, C is a concept, and r a role literal. An *ABox* $\mathcal{A}$ is a finite set of ABox assertions. A *general concept inclusion axiom (GCI)* is an expression of the form $C \sqsubseteq D$, where C and D are an concepts. A *(general) TBox* $\mathcal{T}$ is a finite set of GCIs.

The main ingredients of our approach to reasoning about actions are action descriptions (as defined below), ABoxes for describing the current knowledge about the state of affairs in the application domain, and TBoxes for describing general knowledge about the application domain similar to state constraints in the SitCalc and Fluent Calculus. On the semantic side, interpretations are used to describe the state of affairs in the application domain. Thus, the knowledge described by an ABox is incomplete: ABoxes may admit more than a single model, and all the corresponding states of affairs are considered possible. Before we go deeper into the semantics, we introduce the syntax of action descriptions. We use $\mathcal{LO}$ to denote the extension of a description logic $\mathcal{L}$ with nominals.

Definition 1 (Action). Let $\mathcal{L}$ be a description logic. An *atomic $\mathcal{L}$-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 $\{\mathsf{occ}_{\varphi_1}, \dots, \mathsf{occ}_{\varphi_n}\}$ indexed by $\mathcal{L}$-ABox assertions $\varphi_1, \dots, \varphi_n$ such that each occ_{φ_i} assigns
  - to every concept literal A an $\mathcal{LO}$-concept $\mathsf{occ}_{\varphi_i}(A)$,
  - to every role literal r a finite set $\mathsf{occ}_{\varphi_i}(r)$ 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.

A *composite $\mathcal{L}$-action* is a finite sequence of atomic $\mathcal{L}$-actions $\alpha_1, \dots, \alpha_n$.

Applying an action changes the state of affairs, and thus transforms an interpretation $\mathcal{I}$ into an interpretation $\mathcal{I}'$. Intuitively, the pre-conditions specify under which conditions the action is applicable. The post-condition φ/ψ says that, if φ is true in the original interpretation $\mathcal{I}$, then ψ is true in the interpretation $\mathcal{I}'$ obtained by applying the action. The purpose of the occlusion pattern is to control the changes that are allowed to occur during the execution of the action. This is necessary because, as will be discussed in more detail later, the presence of TBoxes often requires to allow more changes than those that are directly enforced via the post-conditions. To illustrate how the occlusion pattern works, suppose $\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \dots, \mathsf{occ}_{\varphi_n}\}$ and $\varphi_{i_1}, \dots, \varphi_{i_m}$ are the assertions among the indexes in occ that are true in the original interpretation $\mathcal{I}$. If A is a concept name, then instances of the concept

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

in $\mathcal{I}$ may change from A in $\mathcal{I}$ to $\neg A$ in $\mathcal{I}'$, but non-instances 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 pairs from $C^\mathcal{I} \times D^\mathcal{I}$ that have been connected by r in $\mathcal{I}$ may lose this connection in $\mathcal{I}'$, and similarly for the occlusion of negated role names. Note that the indexing of the mappings occ_φ with ABox assertions makes the occlusions conditional: the occlusion occ_φ is only active if φ is satisfied by the original interpretation $\mathcal{I}$. We will explain later why nominals are always admitted in the occlusion pattern, even if they are not provided by $\mathcal{L}$.

For defining the semantics in a succinct way, it is convenient to introduce the following abbreviation. For an action α with $\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \dots, \mathsf{occ}_{\varphi_n}\}$, an interpretation $\mathcal{I}$, a concept literal A, and a role literal r, we set

$$\begin{aligned}
(\mathsf{occ}(A))^\mathcal{I} &:= \bigcup_{\mathcal{I} \models \varphi_i} (\mathsf{occ}_{\varphi_i}(A))^\mathcal{I} \\
(\mathsf{occ}(r))^\mathcal{I} &:= \bigcup_{(C,D) \in \mathsf{occ}_{\varphi_i}(r), \mathcal{I} \models \varphi_i} (C^\mathcal{I} \times D^\mathcal{I})
\end{aligned}$$

Thus, $\mathsf{occ}(B)^\mathcal{I}$ describes those elements of $\Delta^\mathcal{I}$ that may change from B to $\neg B$ when going to $\mathcal{I}'$.

Definition 2 (Action semantics). Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be an atomic 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

$$\begin{array}{rclcl} A^{\mathcal{I}} \setminus A^{\mathcal{I}'} & \subseteq & (\mathsf{occ}(A))^{\mathcal{I}} & \quad & \neg A^{\mathcal{I}} \setminus \neg A^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg A))^{\mathcal{I}} \\ r^{\mathcal{I}} \setminus r^{\mathcal{I}'} & \subseteq & (\mathsf{occ}(r))^{\mathcal{I}} & \quad & \neg r^{\mathcal{I}} \setminus \neg r^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg r))^{\mathcal{I}} \end{array}$$

The composite action $\alpha_1, \ldots, \alpha_n$ *may transform* $\mathcal{I}$ to $\mathcal{I}'$ w.r.t. $\mathcal{T}$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_n} \mathcal{I}'$) iff there are models $\mathcal{I}_0, \ldots, \mathcal{I}_n$ of $\mathcal{T}$ with $\mathcal{I} = \mathcal{I}_0$, $\mathcal{I}' = \mathcal{I}_n$, and $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq n$.

We now give a simple example for DL actions that illustrates the ramification problem introduced by general TBoxes and how occlusion patterns can be used to control this problem. The TBox $\mathcal{T}$ contains the following GCI, which says that everybody registered for a course has access to the university library:

$$\exists \mathsf{registered\_for}.\mathsf{Course} \quad \sqsubseteq \quad \exists \mathsf{access\_to}.\mathsf{Library}$$

The ABox $\mathcal{A}$, which describes the current state of the world, is defined as:

$$\mathsf{Course}(\mathsf{cs}), \quad \neg\exists\mathsf{registered\_for}.\mathsf{Course}(\mathsf{peter}), \quad \neg\exists\mathsf{access\_to}.\mathsf{Library}(\mathsf{peter}).$$

Obviously, $\mathcal{A}$ states that computer science is a course and that Peter is neither registered for a course nor has access to a library. Now, the action

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

describes the registration of Peter for the computer science course. For simplicity, the set of pre-conditions is empty and taut is some valid ABox assertion (say $\top(\mathsf{cs})$), i.e., the post-condition is unconditional.

Following the law of inertia, it may seem reasonable to specify occ such that only the fact stated explicitly in the post-condition can change when executing the action: occ consists of just one (unconditional) mapping $\mathsf{occ}_{\mathsf{taut}}$ which maps all concept and role literals with the exception of $\neg\mathsf{registered\_for}$ to $\bot$ and $\{(\bot, \bot)\}$, respectively. By setting

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

where $\{\mathsf{peter}\}$ and $\{\mathsf{cs}\}$ are nominals, we achieve the desired effect that only the pair $(\mathsf{peter}, \mathsf{cs})$ can be added to "registered_for", and nothing else can be changed. This shows why nominals are indispensible in the occlusion pattern: without them, we are not able to occlude only the change that is enforced by simple post-conditions such as the one in the example.

However, our choice of occlusion pattern is too strict. Due to the presence of the TBox, the action is inconsistent in the sense that there is no model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$ for some model $\mathcal{I}'$ of $\mathcal{T}$. The reason is that, due to $\mathcal{T}$, Peter should have access to a library after execution of the action. Since he does not have access before the action and the occlusion pattern does not allow us to change $\mathsf{access\_to}$, there is no way to achieve this. This shows that admitting general TBoxes induces a ramification problem: there may be indirect effects of an action that are not directly stated in the post-conditions. Such ramifications, which led us to introducing occlusion patterns in the current paper, cannot occur in the more basic DL action formalism introduced in [1] where we admit only acyclic TBoxes and primitive concepts in post-conditions. An obvious way to regain consistency of the action α is to allow via the occlusion pattern that $\mathsf{access\_to}$ can change in the required way, i.e., setting

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

and thus allow Peter to have access to a library after the action.

In the example above, for simplicity we did not use conditional post-conditions, i.e., conditions φ/ψ where φ is not taut. For this reason, it was sufficient that occ consists of a single (unconditional) mapping as well. Obviously, controlling the ramifications of conditional post-conditions requires occlusion patterns that are conditional as well.

The above example suggests that deciding consistency of an action is an important task because failure to specify the occlusion pattern in a proper way can result in inconsistent actions. In the following, we propose two notions of consistency.

Definition 3 (Consistency). Let $\alpha = \alpha_1, \ldots, \alpha_n$ be a composite action, let $\mathcal{T}$ be a TBox, and $\mathcal{A}$ an ABox. We say that

- α is *weakly consistent with $\mathcal{T}$ and $\mathcal{A}$* iff there is a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$, and a model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T}} \mathcal{I}'$.
- α is *strongly consistent with $\mathcal{T}$ and $\mathcal{A}$* iff for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$, there is a model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T}} \mathcal{I}'$.

Ideally, the designer of an action $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ should establish strong consistency of α w.r.t. the relevant TBox and the ABox pre. Note that strong consistency of an action α w.r.t. $\mathcal{T}$ and pre implies strong consistency of α w.r.t. $\mathcal{T}$ and $\mathcal{A}$ for all ABoxes $\mathcal{A}$ that satisfy the preconditions pre. Unfortunately, we shall show later that strong consistency is undecidable even for $\mathcal{ALC}$ actions. In contrast, weak consistency will turn out to be decidable. As demonstrated by our example, in which the first attempt to specify occ yields an action that is not even weakly consistent, checking for weak consistency is helpful to detect severe ramfication problems. In the case of weak consistency, consistency w.r.t. pre does not imply consistency w.r.t. all ABoxes satisfying the pre-conditions. Hence, the designer of an action should not only consider the ABox pre when checking weak consistency, but also additional ABoxes that he considers typical for the application domain.

To check whether an action can be applied in a given situation, the user wants to know whether it is (strongly consistent and) executable, where executable means that 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. This problem is usually called projection [8, 1].

Definition 4 (Executability and projection). Let $\alpha_1, \ldots, \alpha_n$ be a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ for $i = 1, \ldots, n$, let $\mathcal{T}$ be a TBox, and $\mathcal{A}$ an ABox.

- Executablity: $\alpha_1, \ldots, \alpha_n$ is *executable in* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$ iff the following conditions are true for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$:
  - $\mathcal{I} \models \mathsf{pre}_1$
  - for all i with $1 \leq i < n$ and all interpretations $\mathcal{I}'$ with $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_i} \mathcal{I}'$, we have $\mathcal{I}' \models \mathsf{pre}_{i+1}$.
- Projection: The assertion φ is a *consequence of applying* $\alpha_1, \ldots, \alpha_n$ *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 $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_n} \mathcal{I}'$, we have $\mathcal{I}' \models \varphi$.

It is not too difficult to see that the action formalism just introduced is a generalization of the one introduced in [1], for details see [5]. As in [1], projection and executability are mutually reducible in polynomial time. Moreover, (i) an action α is weakly consistent with a TBox $\mathcal{T}$ and ABox $\mathcal{A}$ iff $\bot(a)$ is not a consequence of applying α in $\mathcal{A}$ w.r.t. $\mathcal{T}$; and (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 $(\mathsf{pre}, \mathsf{occ}, \mathsf{post} \cup \{\top(a)/\neg\varphi\})$ is not weakly consistent with $\mathcal{T}$ and $\mathcal{A}$. Thus, since both executability and weak consistency can be reduced to (non-)projection and vice versa, we will concentrate on the latter throughout this paper.

3 Deciding Projection in $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$

We consider the prominent DLs $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$ and show that projection is co-NExpTime-complete. It is shown in [5] that Lemma 8 of [1] implies the following.

Theorem 5. *Projection and executability (weak consistency) in $\mathcal{ALCQI}$ are co-*NExpTime*-hard (*NExpTime*-hard) even if occlusions for roles are restricted to $\{(\bot, \bot)\}$ and only nominals are allowed in the occlusions of concept names.*

Note that projection in $\mathcal{ALCQI}$ is thus harder than subsumption in the same logic, which is ExpTime-complete [13]. In the case of $\mathcal{ALCQIO}$, the complexities of subsumption and projection coincide. Intuitively (and as shown by the proof of Theorem 5), projection in a logic $\mathcal{L}$ should be expected to be of the same complexity as subsumption in $\mathcal{L}$ extended with nominals.

In the following, we establish a matching 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 $\mathcal{ALCQIO}^{\neg,\cap,\cup}$, i.e. the extension of $\mathcal{ALCQIO}$ with the Boolean role constructors.

Let $\alpha_1, \ldots, \alpha_n$ be a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ for $i = 1, \ldots, n$, and let $\mathcal{T}$ be 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 $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. In what follows, we call $\alpha_1, \ldots, \alpha_n$, $\mathcal{T}$, $\mathcal{A}_0$ and φ_0 the *input*. W.l.o.g., we make the following assumptions:

- φ_0 is of the form $\varphi_0 = C_0(a_0)$, where C_0 is a (possibly complex) concept.

  This assumption can be made because an assertion $r(a, b)$ can be replaced with $(\exists r.\{b\})(a)$, and $\neg r(a, b)$ with $(\neg\exists r.\{b\})(a)$.

- Each occlusion pattern occ_i contains exactly one occlusion pattern that is unconditional (i.e., indexed by taut) and formulated in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$.

  An occlusion pattern $\{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ can be converted into an occlusion pattern $\{\mathsf{occ}_{\mathsf{taut}}\}$ formulated in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ as follows. First, we may assume w.l.o.g. that φ_i is of the form $C_i(a_i)$ for $1 \leq i \leq n$ (see previous point). For $1 \leq i \leq n$, let P_i denote the concept $\forall U.(\{a_i\} \to C_i)$, where U denotes the universal role, i.e. $r \cup \neg r$ for some $r \in \mathsf{N_R}$. Then, define for each concept literal A

  $$\mathsf{occ}_{\mathsf{taut}}(A) := \bigsqcup_{1 \leq i \leq n} (P_i \sqcap \mathsf{occ}_{\varphi_i}(A))$$

  Likewise, for each role literal r, define

  $$\mathsf{occ}_{\mathsf{taut}}(r) := \{(P_i \sqcap C, P_i \sqcap D) \mid (C, D) \in \mathsf{occ}_{\varphi_i}\}.$$

  Having the occlusion pattern formulated in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ is unproblematic since our reduction is to $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ anyway. In the following, we slightly abuse notation and confuse the singleton set occ_i with the (unconditional) occlusion mapping contained in it.

The idea of the reduction is to define an ABox $\mathcal{A}_{\mathsf{red}}$ and a TBox $\mathcal{T}_{\mathsf{red}}$ such that a single model of them encodes a sequence of interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}_0, \mathcal{T}$ and $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $i = 1, \ldots, n$. In the following, we use Sub to denote the set of subconcepts of the concepts which occur in the input. In the reduction, we introduce concept names $A^{(i)}$ and role names $r^{(i)}$ for every concept name A and every role name r used in the input, and every $i \leq n$. Intuitively, $A^{(i)}$ and $r^{(i)}$ represent the extensions of A and r in the i-th interpretation. For a complex concept $C \in \mathsf{Sub}$, we use $C^{(i)}$, for $i \leq n$, 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 start by assembling the reduction ABox $\mathcal{A}_{\mathsf{red}}$. First, define a "copy" $\mathcal{A}_{\mathsf{ini}}$ of the input ABox $\mathcal{A}_0$ as:

$$\begin{array}{rl}\mathcal{A}_{\mathsf{ini}} := & \{C^{(0)}(a) \mid C(a) \in \mathcal{A}_0\} \cup \\ & \{r^{(0)}(a,b) \mid r(a,b) \in \mathcal{A}_0\} \cup \{\neg r^{(0)}(a,b) \mid \neg r(a,b) \in \mathcal{A}_0\}\end{array}$$

Then, introduce abbreviations, for $i \leq n$:

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

Now we can define the components of $\mathcal{A}_{\mathsf{red}}$ that take care of post-condition satisfaction. For $1 \leq i \leq n$, we define:

$$\mathcal{A}_{\mathsf{post}}^{(i)} := \{\big(\mathsf{p}_{i-1}(\varphi) \to \mathsf{p}_i(\psi)\big)(a_0) \mid \varphi/\psi \in \mathsf{post}_i\}$$

We assemble $\mathcal{A}_{\mathsf{red}}$ as

$$\mathcal{A}_{\mathsf{red}} := \mathcal{A}_{\mathsf{ini}} \cup \bigcup_{1 \leq i \leq n} \mathcal{A}_{\mathsf{post}}^{(i)}.$$

Next, we define the components of the TBox $\mathcal{T}_{\mathsf{red}}$. Since all interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ have to be models of the input TBox $\mathcal{T}$, we define for each $i \leq n$, a copy $\mathcal{T}^{(i)}$ of $\mathcal{T}$ in the obvious way:

$$\mathcal{T}^{(i)} = \{C^{(i)} \sqsubseteq D^{(i)} \mid C \sqsubseteq D \in \mathcal{T}\}.$$

To deal with occlusions, we introduce auxiliary role names $r^{(i)}_{\mathsf{Dom}(C)}$ and $r^{(i)}_{\mathsf{Ran}(D)}$ for $0 \leq i < n$ and all concepts C, D such that $(C, D) \in \mathsf{occ}_i(s)$ for some role literal s. The following TBox $\mathcal{T}_{\mathsf{aux}}^{(i)}$ ensures that $r^{(i)}_{\mathsf{Dom}(C)}$ and $r^{(i)}_{\mathsf{Ran}(D)}$ are interpreted as $C^{(i)} \times \top$ and $\top \times D^{(i)}$, respectively. It contains the following axioms, for all concepts C, D as above:

$$\begin{array}{rclcrcl}C^{(i)} & \sqsubseteq & \forall \neg r^{(i)}_{\mathsf{Dom}(C)}.\bot & \qquad & \top & \sqsubseteq & \forall r^{(i)}_{\mathsf{Ran}(D)}.D^{(i)} \\ \neg C^{(i)} & \sqsubseteq & \forall r^{(i)}_{\mathsf{Dom}(C)}.\bot & & \top & \sqsubseteq & \forall \neg r^{(i)}_{\mathsf{Ran}(D)}.\neg D^{(i)}\end{array}$$

The following TBox $\mathcal{T}_{\mathsf{fix}}^{(i)}$ ensures that concept and role names do not change unless this is allowed by the occlusion pattern:

- for every concept name A in the input,

$$\begin{aligned} A^{(i)} \sqcap \neg A^{(i+1)} &\sqsubseteq (\mathsf{occ}_{i+1}(A))^{(i)} \\ \neg A^{(i)} \sqcap A^{(i+1)} &\sqsubseteq (\mathsf{occ}_{i+1}(\neg A))^{(i)} \end{aligned}$$

- for every role name r in the input,

$$\begin{aligned} \top &\sqsubseteq \forall\neg\Big(\bigcup_{(C,D)\in\mathsf{occ}_{i+1}(r)} (r^{(i)}_{\mathsf{Dom}(C)} \cap r^{(i)}_{\mathsf{Ran}(D)})\Big) \cap (r^{(i)} \cap \neg r^{(i+1)}).\bot \\ \top &\sqsubseteq \forall\neg\Big(\bigcup_{(C,D)\in\mathsf{occ}_{i+1}(\neg r)} (r^{(i)}_{\mathsf{Dom}(C)} \cap r^{(i)}_{\mathsf{Ran}(D)})\Big) \cap (\neg r^{(i)} \cap r^{(i+1)}).\bot \end{aligned}$$

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

$$\mathcal{T}_{\mathsf{red}} := \bigcup_{0\le i\le n} \mathcal{T}^{(i)} \cup \bigcup_{0\le i<n} \mathcal{T}^{(i)}_{\mathsf{aux}} \cup \bigcup_{0\le i<n} \mathcal{T}^{(i)}_{\mathsf{fix}}.$$

The following is shown in [5]:

Lemma 6. *$C_0(a_0)$ is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}} \cup \{\neg C_0^{(n)}(a_0)\}$ is inconsistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

Since $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ is a fragment of $\mathcal{C}^2$ (the 2-variable fragment of first-order logic with counting), ABox inconsistency in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ is in co-NExpTime even if numbers are coded in binary [7]. Since $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ are polynomial in the size of the input, Lemma 6 gives us a co-NExpTime upper bound for projection in $\mathcal{ALCQIO}$ and $\mathcal{ALCQI}$.

Theorem 7. *Projection and executability are co-*NExpTime*-complete, while weak consistency is* NExpTime*-complete in $\mathcal{ALCQIO}$ and $\mathcal{ALCQI}$.*

4 Undecidability of Strong Consistency

Here we show that strong consistency is undecidable. 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 concepts and interpretations with only one role name r, which we call $\mathcal{ALC}_r$-concepts. Accordingly, we also assume that interpretations interprete 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 [12], 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 ABox $\mathcal{A}_E := \{\neg E(a)\}$ and the atomic action $\alpha_D = (\emptyset, \{\mathsf{occ_{taut}}\}, \mathsf{post})$ with $\mathsf{post} := \{\top(a)/(\exists u.\neg D)(a)\}$ where u is an arbitrary role name and $\mathsf{occ_{taut}}$ maps r and $\neg r$ to $\{(\bot, \bot)\}$, all other role literals to $\{(\top, \top)\}$, and all concept literals to $\top$. Then the action α_D is strongly consistent with the empty TBox and the ABox $\mathcal{A}_E$ iff E is a semantic consequence of D [5]. As an immediate consequence, we obtain the following theorem.

Theorem 9. *Strong consistency of $\mathcal{ALC}$-actions is undecidable, even with the empty TBox.*

5 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. We believe that undecidability of strong consistency is no serious obstacle for the feasibility of our approach in practice. Although deciding strong consistency would provide valuable support for the designer of an action, it 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.

In [6], we investigate the complexity of reasoning with DL actions for other fragments of $\mathcal{ALCQIO}$, most notably $\mathcal{ALC}$, $\mathcal{ALCI}$, and $\mathcal{ALCIO}$. It turns out that, in these logics, deciding projection (as well as executability and weak consistency) is ExpTime-complete, which is in accordance with the observation that, usually, reasoning with $\mathcal{L}$-actions has the same complexity as subsumption in $\mathcal{L}$ extended with nominals. In [5], we also show how reasoning with a restricted version of $\mathcal{ALCQIO}$ actions can be reduced to reasoning in $\mathcal{ALCQIO}$ instead of $\mathcal{ALCQIO}^{\neg,\cap,\cup}$. This is relevant since reasoners for the former (but not for the latter) are readily available.

Acknowledgements. We would like to thank Giuseppe De Giacomo for ideas and discussions. The third author is supported by the DFG Graduiertenkolleg 334. The fourth author is partially supported by UK EPSRC grant no. GR/S63182/01.

References

[1] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proc. of (AAAI-05)*, Pittsburgh, PA, USA, 2005.

[2] F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.

[3] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *J. Log. Program.*, 31(1-3):59–83, 1997.

[4] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI-95*, Montreal, Canada, 1995. Morgan Kaufmann.

[5] 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.

[6] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Description logic actions with general TBoxes: a pragmatic approach. Submitted to JELIA'06, 2006.

[7] I. Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Inf.*, 14(3):369–395, 2005.

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

[9] M. Thielscher. Ramification and causality. *Artificial Intelligence Journal*, 89(1–2):317–364, 1997.

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

[11] M. Thielscher. FLUX: A logic programming method for reasoning agents. *TPLP*, 5(4-5):533–565, 2005.

[12] S. K. Thomason. The logical consequence relation of propositional tense logic. *Z. Math. Logik Grundl. Math.*, 21:29–40, 1975.

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

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