ASP for reasoning about actions with an $\mathcal{EL}^{\bot}$ knowledge base

Laura Giordano[1], Alberto Martelli[2], Matteo Spiotta[2], and Daniele Theseider Dupré[1]

[1] DISIT - Università del Piemonte Orientale, Alessandria, Italy
laura.giordano@uniupo.it, dtd@di.unipmn.it
[2] Dip. di Informatica - Università di Torino, Italy
mrt@di.unito.it , matteo.spiotta@gmail.com

Abstract. In this paper we propose an approach based on Answer Set Programming (ASP) for reasoning about actions in a domain description including knowledge expressed in the low complexity description logic $\mathcal{EL}^{\bot}$. We consider an action theory in which the state is a set of positive and negative assertions, that we represent through explicit negation. The action language allows for non-deterministic actions, and causal rules are introduced to deal with ramifications. We provide sufficient conditions under which action consistency can be guaranteed and we define a polynomial encoding of the action theory in ASP.

1 Introduction

The integration of description logics and action formalisms has recently gained a lot of interest [5, 4, 11, 1]. In this paper we explore the combination of an action language based on Answer Set Programming (ASP) [16] with the low complexity description logic $\mathcal{EL}^{\bot}$ [2, 3]. The temporal extension of ASP proposed in [20] is used as the basis for integrating an $\mathcal{EL}^{\bot}$ knowledge base over the action theory. The aim is reasoning about action execution with an $\mathcal{EL}^{\bot}$ knowledge base.

As usual in the formalisms integrating description logics and action languages [5, 6, 11, 1], we regard inclusions in the KB as state constraints of the action theory, which we expect to be satisfied in the state resulting after an action. In the literature of reasoning about actions it is well known that causal laws and their interplay with domain constraints are crucial for solving the ramification problem [27, 25, 28, 12, 18, 21].

In case knowledge on a domain is expressed in a description logic, the issue has been considered, e.g., in [4] where causal laws are used to ensure the consistency with the TBox (i.e., the set of terminological axioms) of a state resulting from an action. For instance, given a TBox containing $\exists Teaches.Course \sqsubseteq Teacher$, and an ABox (i.e., a set of assertions on individuals) containing the assertion $Course(math)$, an action which adds the assertion $Teaches(john, math)$, without also adding $Teacher(john)$, will not give rise to a next state consistent with the knowledge base. The addition of the causal law **caused** $Teacher(john)$ **if** $Teaches(john, math) \wedge Course(math)$ would allow for instance the above TBox inclusion to be satisfied in the resulting state.

In this paper we define an action theory for reasoning about actions with an $\mathcal{EL}^{\bot}$ knowledge base. The semantics aims at extending to the treatment of non-deterministic actions (as well as to the treatment of frame/non-frame fluents) the approach proposed by Baader et al. [4] which uses causal relationships to deal with the ramification problem in an action formalism based on description logics. In [4] a semantics of actions

and causal laws is defined in the style of Winslett's [29] and McCain and Turner's [27] fixpoint semantics. To deal with non-deterministic effects of actions (not allowed in [4]), as well as with ramifications, through static and dynamic causal laws [21, 19, 12, 13, 20], we provide a semantics based on Answer Sets [16], which appears to be well suited to provide a simple definition of temporal projection for $\mathcal{EL}^{\bot}$ action theories, along the lines of previous work on reasoning about actions in ASP [17, 9, 13, 7].

In particular, as in $\mathcal{EL}^{\bot}$ negated concepts are not included, we allow ASP-like *explicit negation* to occur in assertions within states as well as in direct or indirect effects of actions so to allow for the addition and deletion of assertions by action execution. In this paper, we consider reasoning about states which correspond to DL interpretations; a state transition transforms an interpretation into a new one, in agreement with the semantics of other DL-based approaches to reasoning about actions [5, 4, 11, 1]. We also provide sufficient conditions under which an action specification can be guaranteed to be consistent with the TBox.

The projection problem is reduced to the problem of computing the answer sets of the action theory, given an underlying $\mathcal{EL}^{\bot}$ knowledge base. In this respect, two approaches are feasible: either to encode the action theory into a formalism which already combines ASP and DLs, as in [14], or to provide a direct encoding of the action language in ASP. As inference in $\mathcal{EL}^{\bot}$ can be polynomially encoded into ASP by the materialization calculus proposed by Krötzsch [23], the second approach appears to be a natural choice, which also allows to exploit well known encodings of reasoning about actions into ASP [17, 9, 13, 7]. We define an encoding in ASP of the action theory so that temporal projection and other reasoning problems can be reduced to computing the answer sets of a program, whose size is polynomial in the size of the action theory and of the $\mathcal{EL}^{\bot}$ knowledge base. The complexity of the temporal projection problem is in co-NP so that, as expected, the presence of an $\mathcal{EL}^{\bot}$ TBox does not increase the worst case complexity for the temporal projection problem.

Our approach also relates to the approach to action reasoning proposed in [20], and can be extended by exploiting the approach therein for the verification of LTL temporal properties of an action theory, through bounded model checking.

2 The description logic $\mathcal{EL}^{\bot}$

We consider a fragment of the logic $\mathcal{EL}^{++}$ [2] that, for simplicity of presentation, does not include role inclusions and concrete domains. The fragment, let us call it $\mathcal{EL}^{\bot}$, includes the concept $\bot$ as well as nominals.

We let N_C be a set of concept names, N_R a set of role names and N_I a set of individual names. A concept in $\mathcal{EL}^{\bot}$ is defined as follows:

$$C := A \mid \top \mid \bot \mid C \sqcap C \mid \exists r.C \mid \{a\}$$

where $A \in N_C$ and $r \in N_R$. Observe that complement, disjunction and universal restriction are not allowed in $\mathcal{EL}^{\bot}$.

A KB is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox containing a finite set of concept inclusions $C_1 \sqsubseteq C_2$ and $\mathcal{A}$ is an ABox containing assertions of the form $C(a)$ and $r(a, b)$, with C, C_1, C_2 concepts, $r \in N_R$ and $a, b \in N_I$.

We will assume that the TBox is in normal form [3]. Let BC_{KB} be the smallest set of concepts containing $\top$, all the concept names occurring in KB and all nominals $\{a\}$, for any individual name a occurring in KB. An inclusion is in *normal form* if it has one of the following forms: $C_1 \sqsubseteq D$, $C_1 \sqcap C_2 \sqsubseteq D$, $C_1 \sqsubseteq \exists r.C_2$, $\exists r.C_2 \sqsubseteq D$, where $C_1, C_2 \in BC_{KB}$, and $D \in BC_{KB} \cup \{\bot\}$. In [3] it is shown that any TBox can be normalized in linear time, by introducing new concept and role names.

The following is the usual definition for interpretations and models.

Definition 1 (Interpretations and models). *An interpretation in $\mathcal{EL}^{\bot}$ is any structure $(\Delta^I, \cdot^I)$ where: Δ^I is a domain; $\cdot^I$ is an interpretation function that maps each concept name A to set $A^I \subseteq \Delta^I$, each role name r to a binary relation $r^I \subseteq \Delta^I \times \Delta^I$, and each individual name a to an element $a^I \in \Delta^I$. Furthermore:*

- $\top^I = \Delta^I$, $\bot^I = \emptyset$;
- $\{a\}^I = \{a^I\}$;
- $(C \sqcap D)^I = C^I \cap D^I$;
- $(\exists r.C)^I = \{x \in \Delta \mid \exists y \in C^I : (x, y) \in r^I\}$.

An interpretation $(\Delta^I, \cdot^I)$ satisfies an inclusion $C \sqsubseteq D$ if $C^I \subseteq D^I$; it satisfies an assertion $C(a)$ if $a^I \in C^I$; it satisfies an assertion $r(a, b)$ if $(a^I, b^I) \in r^I$.

Given a $KB = (\mathcal{T}, \mathcal{A})$, an interpretation $(\Delta^I, \cdot^I)$ is a model *of $\mathcal{T}$ if $(\Delta^I, \cdot^I)$ satisfies all inclusions in $\mathcal{T}$; $(\Delta^I, \cdot^I)$ is a* model *of KB if $(\Delta^I, \cdot^I)$ satisfies all inclusions in $\mathcal{T}$ and all assertions in $\mathcal{A}$. $\mathcal{A}$ is* consistent with *$\mathcal{T}$ if there is a model of $\mathcal{T}$ satisfying all the assertions in $\mathcal{A}$.*

In the following we will denote with $N_{C,KB}$, $N_{R,KB}$ and $N_{I,KB}$ the (finite) sets of concept names, role names and individual names occurring in KB.

3 The action theory

Reasoning about actions in a description logic, given a knowledge base $KB = (\mathcal{T}, \mathcal{A})$, consists in reasoning about the evolution of the world, starting from an initial state compatible with the ABox $\mathcal{A}$ and satisfying the TBox $\mathcal{T}$, and updating the state according to actions specifications. In our proposal, as well as in most other proposals for reasoning about actions in DLs [5, 6, 11, 4, 1] the TBox is regarded as a set of state constraints, i.e. conditions that must be satisfied by any state of the world.

This section defines the language of the action theory, and the notion of state. Let $Pred$ be a set of *predicate symbols* and $\mathcal{C}$ be a finite set of *constants*. We define a set of *fluents* $\mathcal{F}$ as a set of ground atomic propositions $p(a_1, \ldots, a_k)$ where $p \in Pred$ is a n-ary predicate symbol and $a_1, \ldots, a_n \in \mathcal{C}$. A *fluent literal* l is a fluent f or its explicit negation $-f$. We denote by Lit the set of fluent literals. We assume that distinguished 0-ary predicates $\bot$ and $\top$ (representing inconsistency and truth) belong to Lit.

We want to regard DL assertions as fluents that occur in our action laws as well as in the states of the action theory. Given a (normalized) $\mathcal{EL}^{\bot}$ knowledge base $KB = (\mathcal{T}, \mathcal{A})$, we require $Pred$ to include:

- a unary predicate C, for each (possibly complex) concept C occurring in KB (we let $Pred_C$ be the set of such predicates);

– a binary predicate r, for each role name $r \in N_{R,KB}$ (we let $Pred_R$ be the set of such predicates);

Also, we require $\mathcal{C}$ to include the individual names occurring in the KB, i.e. $N_{I,KB} \subseteq \mathcal{C}$.

Observe that if a complex concept such as $\exists r.C$ occurs in the KB, there is a predicate name $\exists r.C$ in $Pred$ and for each $a \in N_{I,KB}$, the fluent literals $(\exists r.C)(a)$ and $-(\exists r.C)(a)$ belong to the set Lit (sometimes we will still call such literals assertions). Although classical negation is not allowed in $\mathcal{EL}^{\bot}$ (as well as in ASP), we use *explicit negation* [16] to allow negative literals of the form $-C(a)$ in the action language.

A literal in Lit is said to be a *simple literal* (or a *simple assertion*) if it has the form $B(a)$ or $r(a,b)$ or $-B(a)$ or $-r(a,b)$, where $B \in BC_{KB}$ is a base concept in KB, $r \in N_{R,KB}$ and $a, b \in N_{I,KB}$. Observe that $\{a\}(c)$ and $-\{a\}(c)$ are simple literals, while $(\exists r.C)(a)$ and $-(\exists r.C)(a)$ are non-simple litterals.

In order to deal with existential restrictions, in addition to the individual names $N_{I,KB}$ occurring in the KB we introduce a finite set Aux of auxiliary individual names, as proposed in [23] to encode $\mathcal{EL}^{\bot}$ inference in Datalog, where Aux contains a new individual name $aux^{A \sqsubseteq \exists r.B}$, for each inclusion $A \sqsubseteq \exists r.B$ occurring in the KB. We require that the names in Aux are contained in $\mathcal{C}$. Let $\Delta = N_{I,KB} \cup Aux$ be the enlarged set of named individuals.

A *state* S is a set of literals in Lit. A state S is *consistent* if it is not the case that both a literal and its complement belong to S, nor that $\bot \in S$. A state S is *complete* if the following conditions hold: for all assertions $C(a) \in Lit$, either $C(a) \in S$ or $-C(a) \in S$; and for all assertions $r(a,b) \in Lit$, either $r(a,b) \in S$ or $-r(a,b) \in S$.

Definition 2 (Action theory). *Given a set of actions Σ, a set $Pred$ of predicate symbols and an $\mathcal{EL}^{\bot}$ knowledge base $KB = (\mathcal{T}, \mathcal{A})$ an action theory is a tuple $(KB, \Pi, Frame)$, where:*

- *$KB = (\mathcal{T}, \mathcal{A})$ is an $\mathcal{EL}^{\bot}$ knowledge base;*
- *Π is a set of laws: action, causal, executability and initial state laws (see below);*
- *$Frame$ is a set of positive literals, the set of fluents to which inertia applies*[1].

The notion of *frame* fluents was first introduced in [24].

We introduce action theory laws adopting a notation similar to those used in [9, 22, 21, 13]. *Action laws* describe the direct effects of actions. They have the form:

$$\alpha \textbf{ causes } \phi \textbf{ if } \psi_1 \textbf{ after } \psi_2$$

meaning that the execution of action α in a state in which ψ_2 holds causes ϕ to hold in the new state as a direct effect, if ψ_1 holds in the new state as well, where: α is an action name[2], ϕ is a literal in Lit and $\psi_i = L_1 \wedge \ldots \wedge L_m, not\ L_{m+1} \wedge \ldots \wedge not\ L_n$ is a conjunction of literals $L_i \in Lit$ or their default negations.

[1] For simplicity of exposition, here we consider a single set of frame fluents, but different sets of frame fluents, for distinct actions, could be introduced as done in [26] for occluded fluents.

[2] In the $\mathcal{C}+$ action description language [21] this kind of action laws would be written as **caused** ϕ **if** ψ_1 **after** $\alpha \wedge \psi_2$. In practice, an action name can have parameters also occurring in ϕ, ψ_1 and ψ_2, and a parametric action law is a shorthand for all the instances with individual names.

Non-deterministic effects of actions can be defined using default negation in the body of action laws. For instance, after flipping a coin, it may be head or not:

$$flip \textbf{ causes } head \textbf{ if } not - head$$
$$flip \textbf{ causes } -head \textbf{ if } not\ head$$

Causal laws describe indirect effects of actions. They have the form:

$$\textbf{caused } \phi \textbf{ if } \psi_1 \textbf{ after } \psi_2$$

meaning that ψ_1 causes ϕ to hold whenever ψ_2 holds in the previous state; ϕ is a literal in Lit and ψ_1 and ψ_2 are conjunctions of literals or default negations of literals, as above. Both static and dynamic causal laws are allowed. In particular, a causal law is said *static*, when the condition ψ_2 is $\top$. In such a case, we will write the causal law as **caused** ϕ **if** ψ_1. Observe that, differently from [5, 4], we do not restrict direct and indirect effects of actions to be simple literals.

Precondition laws describe the executability conditions of actions. They have the form: α **executable if** ψ, meaning that the execution of action α is possible in a state where the precondition ψ holds; α is an action name and ψ is a conjunction of literals or default negations of literals.

The *constraints* define conditions that must be satisfied by all states. They have the form: $\bot$ **if** ψ, meaning that any state in which ψ holds is inconsistent.

Initial state laws are needed to introduce conditions that have to hold in the initial state. They have the form: **Init** ϕ **if** ψ. When $\phi = \bot$, we get the a constraint on the initial state **Init** $\bot$ **if** ψ.

Persistency of frame fluents from a state to the next one can be captured by introducing in Π a set of causal laws, said *persistency laws* for all fluents $p \in Frame$:

$$\textbf{caused } p \textbf{ if } not - p \textbf{ after } p$$
$$\textbf{caused } -p \textbf{ if } not\ p \textbf{ after } -p$$

meaning that, if p holds in a state, then p will hold in the next state, unless its negation $-p$ is caused to hold (and similarly for $-p$). Persistency of a fluent is blocked by the execution of an action which causes the value of the fluent to change, or by a nondeterministic action which may cause it to change. The persistency laws above play the role of *inertia rules* in $\mathcal{C}$ [22], $\mathcal{C}+$ [21] and $\mathcal{K}$ [13].

If p is a non-frame fluent, p is not expected to persist and may change its value when an action α is executed. We can introduce *non-frame laws* to guarantee, for some fluent p, that in the new state obtained by executing an action, either p or $-p$ holds:

$$\textbf{caused } p \textbf{ if } not - p$$
$$\textbf{caused } -p \textbf{ if } not\ p$$

In the following we assume that persistency laws and non-frame laws can be applied to simple literals but not to non-simple ones, whose value in a state is determined from the value of simple fluents. For simple literals, one has to choose whether the corresponding concept is frame or non-frame (so that either persistency laws or non-frame laws are introduced). In particular, we assume that all the nominals are always (implicitly) in $Frame$: if $\{a\}(b)$ (respectively $-\{a\}(b)$) belongs to a state, it will persist to the next state unless it is cancelled by the direct or indirect effects of an action.

ABox assertions may incompletely specify the initial state. As we want to reason about states corresponding to $\mathcal{EL}^{\bot}$ interpretations, we assume that the laws:

$$\textbf{Init } p \textbf{ if } \ not - p$$
$$\textbf{Init } -p \textbf{ if } \ not\ p$$

for completing the initial state are introduced in Π for all simple literals p (including assertions with nominals). As shown in [20], the assumption of complete initial states, together with suitable conditions on the laws in Π, gives rise to semantic interpretations (extensions) of the domain description in which all states are complete. In particular, to guarantee that each state is complete for simple literals we assume that, either a simple literal is frame, and persistency laws are introduced for it, or it is non-frame, and non-frame laws are introduced. The other fluents are not subject to this requirement but, as we will see below, the value of existential literals in a state will be determined from the value of simple ones.

Any action theory $(KB, \Pi, Frame)$, where $KB = (\mathcal{T}, \mathcal{A})$, has to contain a set of domain constraints and causal laws, to guarantee that any consistent state S of the action theory respects the semantics of DL concepts in the KB. Observe that, if a state is consistent, then $C(x)$ and $-C(x)$ (nor $r(x, y)$ and $-r(x, y)$) cannot both occur in it.

Let Π_{KB} be the following set of laws:

(1) $\bot$ **if** $\ \bot(a)$
(2) **caused** $\top(x)$ **if** $\top$
(3) **caused** $\{a\}(a)$ **if** $\top$
(4) **caused** $exists\_r\_B(x)$ **if** $r(x, y) \wedge B(y)$
(5) **caused** $(\exists r.B)(x)$ **if** $exists\_r\_B(x)$
(6) **caused** $-(\exists r.B)(x)$ **if** $not\ exists\_r\_B(x)$
(7) $\bot$ **if** $\ \{a\}(x) \wedge B(x) \wedge not\ B(a)$, $\ $ for $x \neq a$
(8) $\bot$ **if** $\ \{a\}(x) \wedge B(a) \wedge not\ B(x)$, $\ $ for $x \neq a$
(9) $\bot$ **if** $\ \{a\}(x) \wedge r(z, x) \wedge not\ r(z, a)$, $\ $ for $x \neq a$

for all $x, y \in \Delta$, $a \in N_{I,KB}$, $B \in BC_{KB}$ (the base concepts occurring in KB) and $r \in N_{R,KB}$ (the roles occurring in KB). Observe that the first constraint has the effect that a state S, in which the concept $\bot$ has an instance, is made inconsistent. Law (4) makes $exists\_r\_B(x)$ hold (where $exists\_r\_B$ is an additional auxiliary predicate for any $B \in BC_{KB}$) in any state in which there is a domain element y such that $r(x, y)$ and $B(y)$ hold. Then, laws (5) and (6) guarantee that, for all $x \in \Delta$, either $(\exists r.B)(x)$ or $-(\exists r.B)(x)$ is contained in the state. State constraints (7-9) are needed for the treatment of nominals and are related to the materialization calculus rules (27-29) [23]. As we will see in Section 4, the laws in Π_{KB} guarantee that any state corresponds to an $\mathcal{EL}^{\bot}$ interpretation.

In general, we are interested in the states which are consistent with the TBox $\mathcal{T}$. We say that a state S *is consistent with the TBox* $\mathcal{T}$ if there is an interpretation $(\Delta^I, \cdot^I)$ such that $(\Delta^I, \cdot^I)$ is a model of $\mathcal{T}$ and $(\Delta^I, \cdot^I)$ satisfies the state S.

Consistency of a state S with the TBox $\mathcal{T}$ can be checked in $\mathcal{EL}^{\bot}$ by defining a knowledge base $(\mathcal{T}', \mathcal{A}')$ such that the ABox $\mathcal{A}'$ contains all the assertions in S^+, and $\mathcal{T}'$ contains all the inclusions in $\mathcal{T}$ as well as, for each assertion $C(a) \in S^-$, an inclusion $\{a\} \sqcap C \sqsubseteq \bot$ and, for each assertion $r(a, b) \in S^-$, an inclusion $\{a\} \sqcap \exists r.\{b\} \sqsubseteq \bot$.

Hence, verifying that a state S *satisfies the TBox* $\mathcal{T}$ amounts to verifying the consistency of the $\mathcal{EL}^{\bot}$ knowledge base $(\mathcal{A}', \mathcal{T}')$, which can be polynomially reduced to a subsumption problem and decided in polynomial time [2].

Let us consider the example from the introduction.

Example 1. Let $KB = (\mathcal{T}, \mathcal{A})$ be a knowledge base such that $\mathcal{T}$= $\{\exists Teaches.Course \sqsubseteq Teacher\}$ and $\mathcal{A}$= $\{\ Person(john), Course(cs1)\}$. We assume that all simple assertions are frame, i.e., *Frame*= $\{Person(x),\ Teacher(x),\ Course(x), Teaches(x, y)$: for all $x, y \in \Delta\}$,

Let us consider a state S_0 where John does not teach any course. If an action $Assign(cs1, john)$ is executed in S_0 and Π contains:

$$\textbf{caused}\ Teaches(john, cs1)\ \textbf{if}\ Assign(cs1, john)$$

the resulting state would contain *Teaches(john,cs1)* and $\exists$*Teaches. Course(john)*, but not *Teacher(john)*, thus violating the inclusion in $\mathcal{T}$. If the causal law:

$$\textbf{caused}\ Teacher(x)\ \textbf{if}\ Teaches(x, y) \wedge Course(y)$$

were present in Π, *Teacher(john)* would be caused to hold in the resulting state (let us call it S_1) which would then be consistent with the TBox. In S_1 *Person(john), Course(cs1), Teaches(john,cs1)* and *Teacher(john)* hold.

Suppose now that the action *Change_to_seminar(cs1)* is executed in S_1 with the effect that *cs1* becomes a seminar and it is not a course any more, i.e., in state S_2, $-Course(cs1)$ holds. By law (6) in Π_{KB}, $-(\exists Teaches.Course)(john)$ also holds. Should we conclude that John is not a teacher any more? As we have included the concept *Teacher* among frame fluents, once John has been recognized to be a teacher, he would remain such until some action (e.g., retiring) causes him not to be a theacher any more. Indeed, *Teacher(john)* still holds in S_2 by persistence.

Consider, now, the case when action *retire(john)* is executed in S_1, and suppose that the action law: **caused** $-Teacher(john)$ **if** $retire(john)$ is in Π. Then, $-Teacher(john)$ will belong to the new state (let us call it S_2'), but S_2' will still contain the literals: *Course(cs1), Teaches(john,cs1)*. Hence, S_2' would violate the TBox. To avoid this Π should contain some causal law which allows to avoid inconsistency, for instance,

$$\textbf{caused} -Teaches(x, y)\ \textbf{if} -Teacher(x) \wedge Course(y)\ \textbf{after}\ Teaches(x, y)$$

By this causal law, when John retires he stops teaching all the courses he was teaching before. In particular, he stops teaching $cs1$.

As we will see in Section 5, the causal laws that are needed to restore consistency when an action is executed can in essence be obtained from the inclusions in the TBox and from their contrapositives, even though not all the contrapositives are always wanted.

4 Semantics of actions

In the previous section, we have defined a *state* of an action theory as a set of literals. Two options can be considered: either the state provides a complete description of the world and is intended to represent an interpretation of the knowledge base; or the state

represents an incomplete specification of the world, which essentially is intended to describe what is known to hold in the knowledge base.

As already mentioned, in the paper we consider the first option, in agreement with the approaches in [5, 6, 11, 4, 1] where, in essence, actions have the effect of updating interpretations. In particular, in this section, we define a semantics of the action theory introduced in the previous section based on the answer set semantics.

Given an action theory $(KB, \Pi, Frame)$, we want to determine the sequences of actions executable according to the action theory by determining: (1) the initial states and (2) the transition relation among states. Concerning the first point, we have the following definition:

Definition 3. *A state S_0 is an initial state of an action theory $(KB, \Pi, Frame)$ if*

(1)S_0 is an answer set of the program P_{init} defined as follows:

$P_{init} = \{\phi \leftarrow \psi \ : \textbf{Init}\ \phi\ \textbf{if}\ \psi \in \Pi\} \cup \{\phi \leftarrow \psi \ : \textbf{caused}\ \phi\ \textbf{if}\ \psi \in \Pi\} \cup \{\bot \leftarrow \psi \ : \bot\ \textbf{if}\ \psi \in \Pi\}$

(2) S_0 satisfies the TBox $\mathcal{T}$

Observe that P_{init} is defined as a set of rules, an ASP program obtained from the initial state laws, as well as from the static causal laws and the state constraints (which hold for all the states). Given an ABox $\mathcal{A}$, initial state laws are introduced to constrain the initial state of the action theory to satisfy ABox assertions. Let us call this set of laws $\Pi_{\mathcal{A}}$. In particular, for each assertion $C(a)$ (resp., $r(a, b)$) in the ABox $\mathcal{A}$, we introduce in $\Pi_{\mathcal{A}}$ a law **Init** $C(a)$ **if** $\top$ (resp., **Init** $r(a, b)$ **if** $\top$). Remember that, to make the initial state complete for simple literals, for all the fluents f of the form $B(a)$ or $r(a, b)$, with $B \in BC_{KB}$, $r \in Pred_R$ and $a, b \in \Delta$, the laws **Init** f **if** $not - f$ and **Init** $-f$ **if** $not\ f$ are assumed to be in Π. We also assume that $\Pi_{\mathcal{A}} \subseteq \Pi$.

Similarly to the initial state, any state of the action theory is required to satisfy all state constraints and static causal laws in Π. And we define a state S to be *admissible for Π* if it is an answer set of the program: $P_{admissible} = \{\phi \leftarrow \psi \ : \textbf{caused}\ \phi\ \textbf{if}\ \psi \in \Pi\} \cup \{\bot \leftarrow \psi \ : \bot\ \textbf{if}\ \psi \in \Pi\}$

The following proposition shows that, for any consistent state S satisfying the laws in Π_{KB}, there is an $\mathcal{EL}^{\bot}$ interpretation satisfying the positive assertions in S and falsifying the negative ones. Given a consistent state S, we let S^+ be the set of $\mathcal{EL}^{\bot}$ assertions $C(a)$ (resp., $r(a, b)$), such that $C(a) \in S$ (resp., $r(a, b) \in S$), and S^- be the set of $\mathcal{EL}^{\bot}$ assertions $C(a)$ (resp., $r(a, b)$), such that $-C(a) \in S$ (resp., $-r(a, b) \in S$).

Proposition 1. *Let $\Pi_{KB} \subseteq \Pi$ and let S be a consistent state admissible for Π. Then there is an interpretation $(\Delta^I, \cdot^I)$ that satisfies all the assertions in S^+ and does not safisfy any of the assertions in S^- (and we say that $(\Delta^I, \cdot^I)$* satisfies *the state S).*

Proof. (Sketch) Let $\sim$ be a relation on Δ defined as follows: $a \sim b$ *if and only if* $\{b\}(a) \in S$. It can be shown by laws (3), (7), (8) that $\sim$ is an equivalence relation. We define the interpretation $(\Delta^I, \cdot^I)$ as follows: $\Delta^I = \Delta/\sim$ is the quotient set of Δ by $\sim$; for all $a \in \Delta$, we let $a^I = [a]_{\sim}$; for all concept names $B \in BC_{KB}$ and $a \in \Delta$, we let $a^I \in B^I$ iff $B(a) \in S$; for all role names $r \in N_{R,KB}$ and $a, b \in \Delta$, we let $(a^I, b^I) \in r^I$ iff $r(a, b) \in S$. It is easy to prove that the interpretation $(\Delta^I, \cdot^I)$ is well defined and that it satisfies S. □

Observe that, in general, if a state S is admissible for Π and $\Pi_{KB} \subseteq \Pi$, S is not guaranteed to be complete for all assertions, i.e. that for all assertions $C(a)$ (or $r(a,b)$) either $C(a) \in S$ or $-C(a) \in S$. However, the initial state is guaranteed to be complete for all assertions, due to the presence of initial state laws and to laws (4-6) in Π_{KB}.

Given a state S, which is consistent and complete for assertions, we want to define a next-state relation to determine the set of possible resulting states when a given action is executed in S. Given Π, we define the set of *direct effects* of action α, when executed in the state S, as follows:

$$\textit{Dir\_Eff}(\alpha, S, \Pi) = \{\phi \leftarrow \psi_1 \ : \alpha \textbf{ causes } \phi \textbf{ if } \psi_1 \textbf{ after } \psi_2 \in \Pi \text{ and } S \models \psi_2\}$$

where the satisfiability of a condition ψ in a state S is defined as follows: $S \models L_1 \wedge \ldots \wedge L_m, not\ L_{m+1} \wedge \ldots \wedge not\ L_n$ iff $S \models L_i$, for all $i = 1, \ldots, m$, and $S \not\models L_j$, for all $j = m+1, \ldots, n$, where, for a literal L, $S \models L$ iff $L \in S$.

Observe that this is the usual definition of satisfiability of a clause body in ASP, as the precondition ψ of an action law is a conjunction of literals or their default negation.

Similarly, we define the set of *indirect effects rules* in a state S, i.e., the set of causal rules that are available in the resulting state, when an action is executed in state S:

$$\textit{Indir\_Eff}(S, \Pi) = \{\phi \leftarrow \psi_1 \ : \textbf{caused } \phi \textbf{ if } \psi_1 \textbf{ after } \psi_2 \in \Pi \text{ and } S \models \psi_2\}$$

Note that both $\textit{Dir\_Eff}(\alpha, S, \Pi)$ and $\textit{Indir\_Eff}(S, \Pi)$ are sets of ASP rules.

Let us now consider the execution of an action α in a state S. We define the conditions that a state S' has to satisfy to be a resulting state of the execution of α in S. Let $P^{\alpha}_{S,\Pi}$ be the set containing the direct effects of action α in S and the indirect effects rules introduced from the causal laws whose *after* precondition is satisfied in S:

$$P^{\alpha}_{S,\Pi} = \textit{Dir\_Eff}(\alpha, S, \Pi) \cup \textit{Indir\_Eff}(S, \Pi)$$

$P^{\alpha}_{S,\Pi}$ is a set of ASP facts and rules, i.e., an ASP program. Remember that Π contains the persistency laws for all frame fluents which, in particular, allow to deal with persistency of the positive and negative assertions from the state S to the new state S'.

In the following definition, we introduce a transition relation from states to states. A first requirement for the new state S' is that it is an answer set of $P^{\alpha}_{S,\Pi}$. A further requirement is that the action α is executable in the state S. An action α is *executable* in the state S if there is an executability law "α **executable if** ψ" in Π and $S \models \psi$.

Definition 4 (Transition relation). *Let S be a state consistent and complete for assertions. S'* is a possible result of the execution of action α in S wrt Π, *if S' is an answer set of $P^{\alpha}_{S,\Pi}$ and α is* executable *in S. The set of possible results of the execution of α in S wrt Π is $Res(\alpha, S, \Pi)$.*

It is possible to show that any state $S' \in Res(\alpha, S, \Pi)$ is consistent and complete for assertions (if S is so) given our assumptions that frame/non-frame laws are introduced for all simple literals. However, as we have seen from the examples, according to the actions specification, S' might not satisfy the TBox $\mathcal{T}$. The requirement that S' satisfies the TBox $\mathcal{T}$ can nevertheless be incorporated in the transition relation. In fact, for a state S which is complete for assertions, the verification that S satisfies a TBox $\mathcal{T}$ (in normal form) can be reduced to the verification that S satisfies a set of state constraints. We associate with $\mathcal{T}$ a set $\Pi_{\mathcal{T}}$ containing the state constraints:

- $\bot$ **if** $A(x) \wedge not\ D(x)$, for each $A \sqsubseteq D$ in $\mathcal{T}$;
- $\bot$ **if** $A(x) \wedge B(x) \wedge not\ D(x)$, for each $A \sqcap B \sqsubseteq D$ in $\mathcal{T}$;
- $\bot$ **if** $A(x) \wedge not\ (\exists r.B)(x)$, for each $A \sqsubseteq \exists r.B$ in $\mathcal{T}$;
- $\bot$ **if** $(\exists r.B)(x) \wedge not\ D(x)$, for each $\exists r.B \sqsubseteq D$ in $\mathcal{T}$;

where $A, B \in BC_{KB}$, $D \in BC_{KB} \cup \{\bot\}$ and $x \in \Delta$. For $D = \bot$, the condition $not\ D(x)$ is omitted. It is easy to see that:

Proposition 2. *Given a state S which is consistent and complete for assertions, S satisfies $\mathcal{T}$ if and only if S satisfies the state constraints in $\Pi_{\mathcal{T}}$.*

This result motivates the following definition of the transition relation.

Definition 5 (Transition relation satisfying $\mathcal{T}$). *Let S be a state consistent and complete for assertions satisfying $\mathcal{T}$. S'* is a possible result of the execution of action α in S satisfying $\mathcal{T}$, *written $S \Rightarrow_{\mathcal{T}}^{\alpha} S'$, if $S' \in Res(\alpha, S, \Pi \cup \Pi_{\mathcal{T}})$.*

Definition 6. *Let S be a state consistent and complete for assertions satisfying $\mathcal{T}$. A state S' is reachable from S through the action sequence $\alpha_1, \ldots, \alpha_k$, written $S \Rightarrow_{\mathcal{T}}^{\alpha_1,\ldots,\alpha_k} S'$, if there is a sequence of states $S = S_0, S_1, \ldots, S_k = S'$ such that $S_{i-1} \Rightarrow_{\mathcal{T}}^{\alpha_i} S_i$, for $i = 1, \ldots, k$. In this case, we say that $\alpha_1, \ldots, \alpha_k$ is* executable *in S wrt. $\mathcal{T}$.*

Besides determining for a given initial state S_0 of an action theory $(KB, \Pi, Frame)$ whether an action sequence $\alpha_1, \ldots, \alpha_k$ is executable in S_0 wrt. $\mathcal{T}$ (executability problem), we want to determine, for a given assertion ϕ, if ϕ holds in all the states resulting from the execution of $\alpha_1, \ldots, \alpha_k$ in any initial state (temporal projection problem).

Definition 7 (Projection). *Given an action theory $(KB, \Pi, Frame)$, an assertion ϕ, and a sequence $\alpha_1, \ldots, \alpha_k$ of actions,* ϕ is a consequence of applying $\alpha_1, \ldots, \alpha_k$ in $\mathcal{A}$ wrt. $\mathcal{T}$ *if, for all states S and S' such that S is an initial state of $(KB, \Pi, Frame)$ and $S \Rightarrow_{\mathcal{T}}^{\alpha_1,\ldots,\alpha_k} S'$, it holds that $S' \models \phi$.*

5 TBox axioms and causal laws to repair inconsistencies

A further problem is that of determining, for an action theory, whether an action specification is consistent with a TBox [4]. Given an action theory $(KB, \Pi, Frame)$, the **action specification** $(\Pi, Frame)$ *is* **consistent** *with the TBox $\mathcal{T}$* if for all pairs of states S, S' and all actions α, if S is consistent, complete for assertions, and it satisfies $\mathcal{T}$, α is executable and $S' \in Res(\alpha, S, \Pi)$, then S' satisfies $\mathcal{T}$.

Given S and α, it may be the case that even though there is some $S' \in Res(\alpha, S, \Pi)$, there is no resulting state that satisfies the TBox (i.e., $Res(\alpha, S, \Pi \cup \Pi_{\mathcal{T}}) = \emptyset$). As observed in [4], when this happens, the action specification can be regarded to be underspecified as it is not able to capture the dependencies among fluents which are specified in the TBox. To guarantee that the TBox is satisfied in the new state, causal laws are needed which allow the state to be repaired.

In general, while defining a domain description, one has to choose which causal relationships are intended by the inclusions and which are not. The choice depends on

the domain and should not be done automatically, although sufficient conditions that guarantee the consistency of action specifications and TBoxs can be defined.

The case of a (normalized) inclusion $A \sqsubseteq B$, with $A, B \in N_C$, is relatively simple; the execution of an action α with effect $A(c)$ (but not $B(c)$), in a state in which none of $A(c)$ and $B(c)$ holds, would lead to a state which violates the constraints in the KB. Similarly for an action β with effect $-B(c)$. Deleting $B(c)$ should cause $A(c)$ to be deleted as well, if we want the inclusion $A \sqsubseteq B$ to be satisfied. Hence, to guarantee that the TBox is satisfied in the new state, for each inclusion $A \sqsubseteq B$, two causal laws are needed:

caused $B(x)$ **if** $A(x)$ and **caused** $-A(x)$ **if** $-B(x)$.

For an axiom $A \sqcap B \sqsubseteq \bot$, consider the concrete case $pending \sqcap approved \sqsubseteq \bot$, representing mutually exclusive states of a claim in a process of dealing with an insurance claim, we expect the following causal laws to be included:

caused $-pending(x)$ **if** $approved(x)$ **after** $-approved(x)$

caused $-approved(x)$ **if** $pending(x)$ **after** $-pending(x)$

even though the second one is only useful if a claim can become pending again after having become (temporarily) approved.

In the same domain, an axiom $\exists approved\_by.clerk \sqsubseteq approved$ could have the associated causal law:

caused $approved(x)$ **if** $(\exists approved\_by.clerk)(x)$

If we admit that the claim, after being approved by a clerk, can be made $-approved$ by a supervisor, we could also add:

caused $-approved\_by(x, y)$ **if** $-approved(x) \wedge clerk(y)$ **after** $approved\_by(x, y)$

while we do not expect $clerk$ to possibly change as a side effect.

Proposition 3. *Given an action theory $(KB, \Pi, Frame)$, a sufficient condition to guarantee the consistency of an action specification Π with a TBox $\mathcal{T}$, is that, for each inclusion in normal form occurring in $\mathcal{T}$, Π contains a set of causal laws, as follows:*

- *For $A \sqsubseteq B$ in $\mathcal{T}$, the laws: 1)* **caused** $B(x)$ **if** $A(x)$*; 2)* **caused** $-A(x)$ **if** $-B(x)$.
- *For $A \sqcap B \sqsubseteq D$, the laws: (3)* **caused** $D(x)$ **if** $A(x) \wedge B(x)$*; and at least one of (4)* **caused** $-A(x)$ **if** $-D(x) \wedge B(x)$ *and (5)* **caused** $-B(x)$ **if** $-D(x) \wedge A(x)$
- *For $A \sqsubseteq \exists r.B$, the laws: (6)* **caused** $r(x, aux^{A \sqsubseteq \exists r.B})$ **if** $A(x)$*;*
  (7) **caused** $B(aux^{A \sqsubseteq \exists r.B})$ **if** $A(x)$ *; (8)* **caused** $-A(x)$ **if** $-(\exists r.B)(x)$*;*
- *For $\exists r.B \sqsubseteq A$, the laws: (9)* **caused** $A(x)$ **if** $(\exists r.B)(x)$*;*
  (10) **caused** $-(\exists r.B)(x)$ **if** $-A(x)$*;*
  at least one of: (11) **caused** $-r(x, y)$ **if** $-A(x) \wedge B(y)$
  (12) **caused** $-B(y)$ **if** $-A(x) \wedge r(x, y)$

Observe that for $A \sqsubseteq \exists r.B$, causal laws (6-7) here, together with (4-5) in Π_{KB}, are such that $(\exists r.B)(x)$ is also caused.

6 Encoding the action theory in ASP

In this section, we show how to translate a domain description to standard ASP.

States are represented in ASP as integers, starting with the initial state 0. We will use the predicates $occurs(Action, State)$, $next(State, State)$ and $holds(Fluent, State)$. Predicate $next(S, SN)$ means that SN is the next state of S:

$next(S, SN) \leftarrow state(S), SN = S + 1.$

Occurrence of exactly one action in each state must be encoded:

$-occurs(A, S) \leftarrow occurs(A1, S), action(A), action(A1), A \neq A1, state(S).$

$occurs(A, S) \leftarrow not\ -occurs(A, S), action(A), state(S).$

To represent the fact that a literal holds in a state, we use different predicates for the fluents that correspond to $\mathcal{EL}^{\bot}$ assertions and for other fluents. In particular, we introduce the predicates $holds\_inst(Concept, Name, State)$ and $holds\_triple(Role, Name, Name, State)$ to represent the fact that an assertions of the form $C(a)$ (resp., $r(a, b)$) hods in a state. Instead, we use the predicate $holds(Fluent, State)$ for the fluents $p(a_1, \ldots, a_n)$ which are not assertions.

The laws Π can be translated as follows: An *action law* α **causes** L_0 **if** ψ_1 **after** ψ_2, where $\psi_1 = L_1 \wedge \ldots \wedge L_m, not\ L_{m+1} \wedge \ldots \wedge not\ L_n$ and $\psi_2 = L'_1 \wedge \ldots \wedge L'_m, not\ L'_{m+1} \wedge \ldots \wedge not\ L'_n$ is translated to:

$h_0 \leftarrow state(S), S' = S + 1, occurs(a, S), h_1 \ldots h_m, not\ h_{m+1} \ldots not\ h_n,$
$h'_1 \ldots h'_m, not\ h'_{m+1} \ldots not\ h'_n$

where $h_0 = (-)holds\_inst(C_0, a_0, S')$ if $L_0 = (-)C_0(a_0)$, $h_0 = (-)holds\_triple(r_0, a_0, b_0, S')$, if $L_0 = (-)r_0(a_0, b_0)$, and $h_0 = (-)holds(p(a_1, .., a_n), S')$ if $L_0 = (-)p(a_1, .., a_n)$ and similarly for the h_i's and h'_j's, using S' for the h_i's and S for the h'_j's (where C_0 in $holds\_inst$ stands for the ground term representing C_0, and similarly $p(a_1, .., a_n)$ within $holds$ stands for the corresponding ground term).

A *dynamic causal law* **caused** L_0 **if** ψ_1 **after** ψ_2, where $\psi_1 = L_1 \wedge \ldots \wedge L_m, not\ L_{m+1} \wedge \ldots \wedge not\ L_n$ and $\psi_2 = L'_1 \wedge \ldots \wedge L'_m, not\ L'_{m+1} \wedge \ldots \wedge not\ L'_n$ is translated to:

$h_0 \leftarrow state(S), S' = S + 1, h_1 \ldots h_m, not\ h_{m+1} \ldots not\ h_n,$
$h'_1 \ldots h'_m, not\ h'_{m+1} \ldots not\ h'_n$

where the h_i's and h'_j's are defined as before, using S' for the h_i's and S for the h'_j's.

A *precondition law* α **executable if** $L_1 \wedge \ldots \wedge L_m, not\ L_{m+1} \wedge \ldots \wedge not\ L_n$ is translated to the following two ASP rules, where the h_i's are defined as before using S and the predicate $executable(Action, State)$:

$\leftarrow state(S), occurs(A, S), not\ executable(A, S)$
$executable(A, S) \leftarrow h_1 \ldots h_m, not\ h_{m+1} \ldots not\ h_n$

A *constraint* $\bot$ **if** $L_1 \wedge \ldots \wedge L_m, not\ L_{m+1} \wedge \ldots \wedge not\ L_n$ is translated to the following ASP constraint, where the h_i's are defined as before using S:

$$\leftarrow state(S), occurs(a, S), h_1 \ldots h_m, not\ h_{m+1} \ldots not\ h_n$$

Initial state laws are translated in a similar way but they are evaluated in state 0.

Given a domain description $(KB, \Pi, Frame)$, we denote by $tr(\Pi)$ the set of rules containing the translation of each law in Π, including those in Π_{KB}, $\Pi_{\mathcal{A}}$ and in $\Pi_{\mathcal{T}}$, as well as the definitions of $occurs$ and $next$ introduced above.

Proposition 4. *Given an action theory $(KB, \Pi, Frame)$, let $tr(\Pi)$ be the encoding of Π as defined above. The following holds:*

- *To check for the* executability *of an action sequence* $act1, act2, \ldots, actn$*, the facts:* $occurs(act1, 0), occurs(act2, 1), \ldots, occurs(actn, n-1)$ *should be included in* $tr(\Pi)$*. The sequence is executable when* $tr(\Pi)$ *has an answer set.*
- *To check whether a given literal* $L = (-)C(a)$ *(or* $(-)r(a, b)$*) holds in some of the resulting states after executing the action sequence* $act1, act2, \ldots, actn$ *from an initial state, one has to look for an answer set of* $tr(\Pi)$ *enriched with the following:*
  - $occurs(act1, 0)$*,* $occurs(act2, 1), \ldots, occurs(actn, n-1)$*,*
  - *the constraint* $\leftarrow$ *not* $(-)holds\_inst(C, a, n)$ *if* $L = (-)C(a)$
  - *the constraint* $\leftarrow$ *not* $(-)holds\_triple(R, a, b, n)$ *) if* $L = (-)r(a, b)$
- *For the temporal projection problem, checking whether a literal* $L = (-)C(a)$ *(or (-)*$r(a, b)$*) holds in all the possible resulting states after executing the action sequence* $act1, act2, \ldots, actn$ *from an initial state, amounts to checking the non-existence of answer sets when the following rules are added to* $tr(\Pi)$*:*
  - $occurs(act1, 0)$*,* $occurs(act2, 1), \ldots, occurs(actn, n-1)$
  - *the constraint* $\leftarrow$ $(-)holds\_inst(C, a, n)$ *if* $L = (-)C(a)$
  - *the constraint* $\leftarrow$ $(-)holds\_triple(R, a, b, n)$ *) if* $L = (-)r(a, b)$

Notice that size of the ASP encoding is polynomial in the size of the action theory.

7 Conclusions and Related Work

In this paper we have proposed an approach for reasoning about actions in an action language that includes a knowledge base in $\mathcal{EL}^{\bot}$ and allows for causal laws and non-deterministic actions. We have provided a semantics for the action language based on Answer Sets as well as a polynomial ASP encoding of a domain description. It follows that the temporal projection problem in our action logic is a co-NP problem and that it can be solved by using standard ASP solvers.

Many of the proposals in the literature for combining DLs with action theories focus on expressive DLs. In their seminal work [5], Baader et al. study the integration of action formalisms with expressive DLs, from $\mathcal{ALC}$ to $\mathcal{ALCOIQ}$, under Winslett's *possible models approach* (PMA) [29], based on the assumption that TBox is acyclic and on the distinction between defined and primitive concepts (i.e., concept names that are not defined in the TBox), where only primitive concepts are allowed in action effects. They determine the complexity of the executability and projection problems and show that they get decidable fragments of the situation calculus. Our semantics departs from PMA as causal laws are considered. As [26] and [4] we do not require acyclic TBoxes and primitive concepts in postconditions.

The requirement of acyclic TBoxes is lifted in the paper by Liu et al. [26], where an approach to the ramification problem is proposed which does not use causal relationships, but exploits *occlusion* to provide a specification of the predicates that can change through the execution of actions. The idea is to leave to the designer of an action description the control of the ramification of the actions.

Similar considerations are at the basis of the approach by Baader et al. [4] that, instead, exploit causal relationships for modeling ramifications. They define an action language for $\mathcal{ALCO}$, which deals with the ramification problem using causal relationships and provide a semantics for it in the style of McCain and Turner fixpoint semantics.

They show that temporal projection is decidable and EXPTIME-complete. Their action theory does not deal with non-deterministic effects of actions. In this paper, following [4], we exploit causal laws for modeling ramifications in the context of an action language for $\mathcal{EL}^{\bot}$. We allow for non-deterministic effects of actions and for the distinction between frame and non-frame fluents [24] (which is strongly related to occlusion used in [26]) based on the answer set semantics. We also provide sufficient conditions for an action specification to be consistent with a normalized $\mathcal{EL}^{\bot}$ KB.

Ahmetai et al. [1] study the evolution of Graph Structured Data as a result of updates expressed in an action language. They provide decidability and complexity results for expressive DLs such as $\mathcal{ALCHOIQ}br$ (under finite satisfiability) as well as for variants of DL-*lite*. Complex actions including action sequence and conditional actions are considered. Complex actions are considered as well in [11], where an action formalism is introduced for a family DLs, from $\mathcal{ALCO}$ to $\mathcal{ALCHOIQ}$, exploiting PDL program constructors to define complex actions. As in [5], the TBox is assumed to be acyclic.

In [8] Description Logic and Action Bases are introduced, where an initial Abox evolves over time due to actions which have conditional effects. In the basic approach, if the resulting state it inconsistent, is not considered; in [10] the approach is extended to allow for different notions of *repairing* the resulting state: either a maximal subset that is consistent with the Tbox, or the intersection of all such subsets. In this paper, we rely on causal laws for repairing states; selecting the appropriate causal laws means acquiring more knowledge, and it allows for a finer control of the resulting state.

In [14] a language, dl-programs, is proposed which combines logic programming, under the answer set semantics, and the expressive description logics $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$. In this paper, instead, we do not propose a general language paradigm, but we provide a semantics of $\mathcal{EL}^{\bot}$ action theories based on answer sets, that can be easily encoded in ASP borrowing some ideas from the materialization calculus in [23]. The system DReW [15] provides an implementation of dl-programs for two DLs, including $\mathcal{SROEL}(\sqcap, \times)$, also using the materialization calculus in [23]. However, as in our approach states are complete for $\mathcal{EL}^{\bot}$ assertions, and correspond to $\mathcal{EL}^{\bot}$ interpretations, we do not need to perform inference in $\mathcal{EL}^{\bot}$, but just model checking.

Our semantics for actions requires, as in many proposals in the literature, that the state provides a complete description of the world and is intended to represent an interpretation of the knowledge base. Alternatively, as in [8], a state could represent an incomplete specification of the world, which describes what is known about the world. In this way, one can avoid, for a given ABox, to consider as initial states all the complete states consistent with it and with the TBox. Rather, one can consider a single initial state as an epistemic state representing what is known and what is unknown. We leave the study of this alternative approach for future work.

Our proposal is related to the approach to action reasoning in ASP proposed in [20], where verification of LTL temporal properties of an action theory is encoded in ASP through bounded model checking. As future work, we aim at incorporating verification of temporal properties also in the approach proposed in this paper.

Acknowledgements. This research is partially supported by INDAM - GNCS Project 2016 *Ragionamento Defeasible nelle Logiche Descrittive*.

References

1. Ahmetaj, S., Calvanese, D., Ortiz, M., Simkus, M.: Managing change in graph-structured data using description logics. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. pp. 966–973 (2014)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. IJCAI 2005. pp. 364–369. Edinburgh, Scotland, UK (August 2005)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: LTCS-Report LTCS-05-01. Inst. for Theoretical Computer Science, TU Dresden (2005)
4. Baader, F., Lippmann, M., Liu, H.: Using causal relationships to deal with the ramification problem in action formalisms based on description logics. In: LPAR-17. pp. 82–96 (2010)
5. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Proc. AAAI 2005. pp. 572–577 (2005)
6. Baader, F., Liu, H., ul Mehdi, A.: Verifying properties of infinite sequences of description logic actions. In: ECAI. pp. 53–58 (2010)
7. Babb, J., Lee, J.: Cplus2asp: Computing action language $\mathcal{C}$+ in Answer Set Programming. In: Proc. Logic Programming and Nonmonotonic Reasoning, LPNMR 2013. pp. 122–134 (2013)
8. Bagheri Hariri, B., Calvanese, D., Montali, M., De Giacomo, G., De Masellis, R., Felli, P.: Description logic knowledge and action bases. J. Artif. Intell. Res. 46, 651–686 (2013)
9. Baral, C., Gelfond, M.: Reasoning agents in dynamic domains. In: Logic-Based Artificial Intelligence, pp. 257–279 (2000)
10. Calvanese, D., Kharlamov, E., Montali, M., Santoso, A., Zheleznyakov, D.: Verification of inconsistency-aware knowledge and action bases. In: Proc. IJCAI 2013
11. Chang, L., Shi, Z., Gu, T., Zhao, L.: A family of dynamic description logics for representing and reasoning about actions. J. Autom. Reasoning 49(1), 1–52 (2012)
12. Denecker, M., Theseider Dupré, D., Van Belleghem, K.: An inductive definitions approach to ramifications. Electronic Transactions on Artificial Intelligence 2, 25–97 (1998)
13. Eiter, T., Faber, W., Leone, N., Pfeifer, G., Polleres, A.: A logic programming approach to knowledge-state planning: Semantics and complexity. ACM Transactions on Computational Logic 5(2), 206–263 (2004)
14. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artificial Intellgence 172(12-13) (2008)
15. Eiter, T., Krennwallner, T., Schneider, P., Xiao, G.: Uniform evaluation of nonmonotonic dl-programs. In: Foundations of Information and Knowledge Systems - 7th International Symposium, FoIKS 2012, Kiel, Germany, March 5-9, 2012. Proceedings. pp. 1–22 (2012)
16. Gelfond, M.: Handbook of Knowledge Representation, chapter 7, Answer Sets. Elsevier (2007)
17. Gelfond, M., Lifschitz, V.: Action languages. Electron. Trans. Artif. Intell. 2, 193–210 (1998)
18. Giordano, L., Martelli, A., Schwind, C.: Ramification and causality in a modal action logic. J. Log. Comput. 10(5), 625–662 (2000)
19. Giordano, L., Martelli, A., Schwind, C.: Reasoning about actions in dynamic linear time temporal logic. The Logic Journal of the IGPL 9(2), 289–303 (2001)
20. Giordano, L., Martelli, A., Theseider Dupré, D.: Reasoning about actions with temporal answer sets. Theory and Practice of Logic Programming 13, 201–225 (2013)
21. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., , Turner, H.: Nonmonotonic causal theories. Artificial Intelligence 153(1-2), 49–104 (2004)
22. Giunchiglia, E., Lifschitz, V.: An action language based on causal explanation: Preliminary report. In: Proc. AAAI/IAAI 1998. pp. 623–630 (1998)

23. Krötzsch, M.: Efficient inferencing for OWL EL. In: Proc. JELIA 2010. pp. 234–246 (2010)
24. Lifschitz, V.: Frames in the space of situations. Artificial Intelligence 46, 365–376 (1990)
25. Lin, F.: Embracing causality in specifying the indirect effects of actions. In: IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes. pp. 1985–1993 (1995)
26. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Reasoning about actions using description logics with general tboxes. In: Proc. JELIA 2006, Liverpool, UK. pp. 266–279 (2006)
27. McCain, N., Turner, H.: A causal theory of ramifications and qualifications. In: Proc. IJCAI 95. pp. 1978–1984 (1995)
28. Thielscher, M.: Ramification and causality. Artif. Intell. 89(1-2), 317–364 (1997)
29. Winslett, M.: Reasoning about action using a possible models approach. In: Proc. AAAI, St. Paul, MN, August 21-26, 1988. pp. 89–93 (1988)