On the Undecidability of the Situation Calculus Extended with Description Logic Ontologies

Diego Calvanese
Free University of Bozen-Bolzano
Bolzano, Italy
calvanese@inf.unibz.it

Giuseppe De Giacomo
Sapienza Università di Roma
Roma, Italy
degiacomo@dis.uniroma1.it

Mikhail Soutchanski
Ryerson University
Toronto, Canada
mes@cs.ryerson.ca

Abstract

In this paper we investigate situation calculus action theories extended with ontologies, expressed as description logics TBoxes that act as state constraints. We show that this combination, while natural and desirable, is particularly problematic: it leads to undecidability of the simplest form of reasoning, namely satisfiability, even for the simplest kinds of description logics and the simplest kind of situation calculus action theories.

1 Introduction

Combining action formalisms, which describe how systems evolve over time as consequences of actions, and ontological formalisms, which capture knowledge about the domain structure that is invariant over time, is a strong desiderata in AI and CS. Initiatives like OWL-S have been putting forward explicitly this need [Martin and others, 2004]. More recently, the work on data aware processes has come back to this issue by giving a formal account of processes that manipulate unbounded data and devising techniques for their verification [Belardinelli *et al.*, 2012; De Giacomo *et al.*, 2012; Bagheri Hariri *et al.*, 2013a; 2013b].

In this paper we study combining the *situation calculus* [Reiter, 1991; 2001], possibly the best known first-order based formalism for reasoning about actions, with *description logics* [Brachman and Levesque, 1984; Baader *et al.*, 2003], which are currently considered the main formalisms for expressing ontologies. In particular we extend situation calculus action theories with description logics TBoxes, acting as a sophisticated form of state constraints [Lin and Reiter, 1994]. From the technical point of view, we consider the simplest and most natural combination, namely the union of the two sets of axioms. In this way, we have the two ingredients combined into a single logical theory, and we have a single logical language to talk about both static and dynamic aspects of the system.

In fact, this combination is related to combining description logics and temporal logics, which is a well studied subject. Even though recently some decidability results for simple description logics have been obtained [Artale *et al.*, 2013], it is well known that this combination is problematic in general. Indeed, it gives rise to a notorious two-dimensional semantics (where one dimension is for time and the other for the description logic domain), which suffers from a key computational problem: the possibility of specifying that roles, i.e., binary predicates, preserve their extension over time causes undecidability, due to the possibility of easily encoding bidimensional grids, and hence tilings or Turing machine computations [Wolter and Zakharyaschev, 1999; Gabbay *et al.*, 2003]. Notice that if we drop the capability of preserving the extension of binary predicates over time, the usefulness of such a combination becomes disputable: for example we would not be able to represent that a person remains subscribed to some mailing list until s/he unsubscribes.

In light of these results we may already suspect that the combination of the situation calculus and description logics in a single logical theory is going to lead to undecidability of reasoning in general. The aim of this paper is to investigate how deep this problem really is. To do so, we consider a simplified setting in which we partition fluents into two kinds. The first kind includes fluents that constitute what we may call the "*frame*", for which effect axioms are directly specified and inertia is assumed (frame problem solved). The second kind includes fluents for which no direct effects are specified and inertia is not assumed. Hence these fluents evolve over time only due to *ramifications* through the state constraints of the effects on the fluents in the frame, and inertia is not assumed. State constraints predicate on the relationship between fluents of both kinds in a single state. Specifically, our simplified setting is as follows: *(i)* we use only unary and binary fluents (i.e., fluents with one or two object arguments, apart from the situation one); *(ii)* we characterize the frame through a situation calculus basic action theory [Reiter, 1991; 2001]; in particular, we use regular successor state axioms to specify how actions affects fluents in the frame (including the solution to the frame problem). *(iii)* as state constraints, we use a TBox expressed in a given description logic, i.e., a finite set of universal inclusion assertions that must hold in every situation; *(iv)* the initial situation description consists of a finite set of positive facts involving fluents.

Here we show that in the above setting the simplest form of reasoning, namely *satisfiability*, the minimal requirement for any logical theory, is undecidable, even when the state constraints are expressed in the simplest description logics, such as *DL-Lite*$_{core}$, and the situation calculus action theory is of the simplest form, namely *context free* in the sense of [Reiter,

1991] and *local effect* [Vassos *et al.*, 2008]. This contrasts, on the one hand, with the fact that, without state constraints, since the initial situation description consists only of a finite set of positive facts, satisfiability trivially holds [Reiter, 2001]; and, on the other hand, with the fact that, if one considers the initial situation description as an ABox, the decidability of satisfiability of such ABox with any TBox is guaranteed in all description logics [Baader *et al.*, 2003]. Naturally, undecidability of satisfiability implies that all other typical tasks in reasoning about action, such as projection, executability of actions, verification, etc. are also undecidable.

Interestingly, sometimes state constraints can be compiled into the basic action theory itself [Lin and Reiter, 1994; Lin, 1995; Lin and Soutchanski, 2011]. In light of the good computational properties of description logics, one might hope that the compilation approach could be applied also in our setting to get computational effectiveness. But the very result of undecidability of satisfiability implies that this is not the case. Indeed, the basic action theory resulting from the compilation, besides the standard (though possibly complex) successor state axioms, would also contain a new initial situation description. Satisfiability would depend only on the latter, so by our undecidability result such initial situation description could not consist of a description logic knowledge base, since these are always decidable.

The negative results we obtained, even under extreme simplifying restrictions, indicate the need of alternative forms of combinations, strengthening some of the proposals in literature, which, although may look more ad-hoc, guarantee decidability or better computational behavior. We come back to this point in the conclusion.

The rest of the paper is organized as follows. After some preliminaries (Sec. 2), we show by a simple reduction from tiling, that satisfiability of basic action theories extended with $\mathcal{ALC}$ TBoxes is undecidable (Sec. 3). Then we strenghten the result to the simplest description logic of the $\mathcal{AL}$ family (Sec. 4). Finally, by resorting directly to a reduction from the halting problem of Turing machines, we show undecidability also for the simplest logics in the lightweight description logics families *DL-Lite*, and $\mathcal{EL}$, even if the action theory is local effects and context free (Sec. 5). A short discussion concludes the paper (Sec. 6).

2 Preliminaries

Situation Calculus. The situation calculus [Reiter, 1991; 2001] is a multi-sorted predicate logic language for axiomatizing dynamic systems with sort *actions*, sort *situations* and sort *objects*. *Actions* $\alpha_i(\vec{x})$ are first-order (FO) terms built from an action function symbol α_i and a possibly empty tuple $\vec{x}$ of object arguments. *Situations* are first-order terms denoting states resulting from sequences of actions. The constant S_0 denotes the *initial situation*, namely the empty action sequence, and function $do(a, s)$ denotes the next situation resulting from executing action a in situation s. *Objects* are first-order terms other than actions and situations. *Fluents* are relations whose values may vary between situations (we do not consider functional fluents). A fluent is denoted by a predicate symbol whose last argument has the sort *situation*.

A *basic action theory* (BAT) $\mathcal{D}$ is a set of five classes of axioms to model actions and their effects [Reiter, 2001] described below. In axioms, all free variables (they start with lower roman case letters) including object variables $\vec{x}$, situation variable s, and a variable a of sort action are taken to be $\forall$-quantified at the front.

Foundational axioms for situations. These are domain independent second-order axioms that characterize situations [Reiter, 2001].

Unique name axioms (UNA). The actions of the domain and the object constants are pairwise unequal.

Action precondition axioms. For each action $\alpha(\vec{x})$, there is one axiom of the form $Poss(\alpha(\vec{x}), s) \equiv \Pi_\alpha(\vec{x}, s)$, where $Poss(\alpha(\vec{x}), s)$ is a special predicate that denotes that the action $\alpha(\vec{x})$ is executable in situation s, and the condition $\Pi_\alpha(\vec{x}, s)$ is a formula *uniform* in s, i.e., such that s is the only situation term, with the other free variables among $\vec{x}$ at most, and where $Poss$ cannot occur.

Successor state axioms (SSAs). For each fluent $F(\vec{x}, s)$ (in the BAT), there is an axiom of the form

$$F(\vec{x}, do(a, s)) \equiv \gamma^+_F(\vec{x}, a, s) \vee (F(\vec{x}, s) \wedge \neg\gamma^-_F(\vec{x}, a, s))$$

where $\gamma^+_F(\vec{x}, a, s), \gamma^-_F(\vec{x}, a, s)$ are formulas uniform in s. with free variables among $\vec{x}$, a, and s at most. This SSA defines the value of F in the next situation $do(a, s)$ in terms of what holds in the current situation s. Each formula $\gamma^+_F(\vec{x}, a, s)$ (resp., $\gamma^-_F(\vec{x}, a, s)$) is a finite disjunction of

$$\exists\vec{y}\,(a{=}\alpha_i(\vec{t}, \vec{y}) \wedge \psi_i(\vec{x}, \vec{y}, s)) \tag{1}$$

for some action term $\alpha_i(\vec{t}, \vec{y})$ and some situation calculus formula $\psi_i(\vec{x}, \vec{y}, s)$ uniform in s, called a *context condition*, all of whose free object variables are at most among $\vec{x}, \vec{y}$. All object variables $\vec{t}$ are included in $\vec{x}$, and $\vec{y}$ are the new variables not included in $\vec{x}$, if any. If the context condition is situation independent, e.g., $\psi_i(\vec{x}, \vec{y}, s)$ is a boolean combination of equalities between variables from $\vec{x}, \vec{y}$, then an action $\alpha_i(\vec{t}, \vec{y})$ has an unconditional effect on a fluent F. In the SSA for the fluent F, if all actions have unconditional effects on F, then this SSA is called *context-free*. When all SSA are context-free, the BAT is said to be context-free. Here, we consider the original variant of context-free SSAs [Reiter, 1991]:

$$Poss(a, s) \rightarrow \big(F(\vec{x}, do(a, s)) \equiv \bigvee_i \exists\vec{y}(a{=}\alpha_i(\vec{t}, \vec{y})) \vee (F(\vec{x}, s) \wedge \bigwedge_j \neg\exists\vec{y}(a{=}\alpha_j(\vec{t}, \vec{y})))\big)$$

where the SSA specifies the extension of F in the situation resulting from executing action a in s, only if a is indeed executable in s according to its precondition axiom.

When in Formula (1) above, the variables $\vec{t}$ in $\alpha_i(\vec{t}, \vec{y})$ include all of the arguments $\vec{x}$ of the fluent on the left hand side of a generic SSA, we say that this action α_i has a *local effect* on the fluent F because α_i can change the fluent only at the single tuple of object constants, which is used to instantiate arguments of the action, [Vassos *et al.*, 2008]. Notice that, an action with non-local effects can change a fluent at infinitely many object elements, in general. When all actions have only local effect, the BAT is said to be local effect.

Initial situation description. A set of logical formulas uniform in S_0. It specifies the values of all fluents in the initial state. In particular in this paper we will consider $\mathcal{D}_{S_0}$ to be constituted by a finite set of positive facts, i.e., ground atomic formulas (under open-world assumption). Notice that because of the form of basic action theories, if the axioms constituting the initial situation description (together with UNA) are satisfiable (and this is always the case, when such a description amounts to a finite set of positive facts) then the entire action theory is satisfiable [Reiter, 2001].

Description Logics TBoxes. Description logics (DLs) [Baader *et al.*, 2003] are formalisms for representing knowledge on a domain of interest that is invariant over time, in terms of individuals denoting objects, concepts denoting sets of objects, and roles denoting binary relations between objects. In DLs, starting from countably infinite, mutually disjoint sets of *concepts names* (denoted by A) and *roles names* (denoted by P), we construct complex concept and role expressions (or simply, *concepts* C and *roles* R) by inductively applying constructors that depend on the DL in question.

A DL *TBox* is constituted by a finite set of *(general) concept inclusions* of the form $C \sqsubseteq C'$, whose form again depends on the DL. Notice that we allow inclusions to be cyclic, e.g., $\exists parent \sqsubseteq Person$, $\exists parent^- \sqsubseteq Person$, and $Person \sqsubseteq \exists parent$ which type the $parent$ role on both components with $Person$, and express that every person has a parent. This is required in virtually all ontology-based and conceptual modeling applications: e.g., it is needed to capture UML Class Diagrams or ER Schemas [Baader *et al.*, 2003].[1]

In this paper we focus mainly on the DLs $\mathcal{ALC}$, $\mathcal{EL}_\bot$, and *DL-Lite*$_{core}$. In $\mathcal{ALC}$, roles are just names P, and concepts are formed according to the syntax:

$$C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall P.C \mid \exists P.C.$$

IN fact $C \sqcup C'$ is equivalent to $\neg(\neg C \sqcap \neg C')$, and $\forall P.C$ to $\neg\exists P.\neg C$. An $\mathcal{ALC}$ TBox consists of concept inclusions between arbitrary $\mathcal{ALC}$ concepts.

In the *DL-Lite* family [Calvanese *et al.*, 2007; 2013], a role R is either a role name P or an inverse role P^-, while a concept C is either a concept name A, or the projection of a role P on its first, $\exists P.\top$ written $\exists P$, or second, $\exists P^-.\top$ written $\exists P^-$, component. A *DL-Lite*$_{core}$ TBox is a set of concept inclusions of the form $C \sqsubseteq C'$ or $C \sqsubseteq \neg C'$, where C and C' are *DL-Lite* concepts. Notice that in *DL-Lite* negation is used just to express disjointness.

In the $\mathcal{EL}$ family [Baader *et al.*, 2008], roles are names, and concepts obey the syntax $C ::= A \mid C \sqcap C \mid \exists P.C$. An $\mathcal{EL}_\bot$ TBox consists of concept inclusions of the form $C \sqsubseteq C'$, $C \sqsubseteq \bot$, $\mathsf{dom}(P) \sqsubseteq C$, or $\mathsf{ran}(P) \sqsubseteq C$, where C, C' are $\mathcal{EL}$ concepts, $\bot$ denotes the empty concept, $\mathsf{dom}(P)$ denotes the range of role P (i.e., $\exists P$) , and $\mathsf{ran}(P)$ its range (i.e., $\exists P^-$).

We observe that *DL-Lite*$_{core}$ is a fragment of the variant of *DL-Lite* corresponding to the OWL 2 QL profile of OWL 2, and that $\mathcal{EL}_\bot$ is a fragment of $\mathcal{EL}^{++}$, the DL corresponding to the OWL 2 EL profile [Motik and others, 2012].

The semantics of DLs is based on first-order *interpretations*. In fact, one can provide a translation function π_x that translates an atomic DL concept C into an open formula $C(x)$, and TBox inclusions into first-order axioms. The translation π_x is inductively defined as follows:

$$\begin{array}{rcl}
\pi_x(C \sqsubseteq C') & = & \forall x.\pi_x(C) \rightarrow \pi_x(C') \\
\pi_x(\bot) & = & \mathit{false} \\
\pi_x(A) & = & A(x) \\
\pi_x(\neg C) & = & \neg\pi_x(C) \\
\pi_x(C \sqcap C') & = & \pi_x(C) \wedge \pi_x(C') \\
\pi_x(\exists P.C) & = & \exists y.P(x,y) \wedge \pi_y(C) \\
\pi_x(\exists P^-.C) & = & \exists y.P(y,x) \wedge \pi_y(C).
\end{array}$$

BATs extended with TBoxes. We use DL TBoxes as state constraints [Lin and Reiter, 1994]. These are situation calculus sentences having a single situation variable s as the only situation term, which is universally quantified. To use a DL TBoxes as state constraints, it suffices to add a situation argument s to the atomic predicates corresponding to concepts and roles, and quantify over the situation universally in concept inclusions. In particular, we consider fluents partitioned into two sets. On the first set, which we call the *frame*, we have SSAs: so effects of actions are directly specified on them otherwise inertial holds (solution to the frame problem). On the second one we don't and their extension is only constrained by the values of the fluents in the first set and the TBox state constraints only. Inertia it is not assumed. We call the resulting action theories *BATs extended with TBoxes*.

3 BATs with $\mathcal{ALC}$ TBoxes

We start our analysis by showing undecidability of the satisfiability problem for basic action theories extended with an $\mathcal{ALC}$ TBox as state constrains. For now, we pose no restriction on the form of BAT. This is possibly the easiest case to consider, and the proof can be done quite naturally by relying on *tiling*. In the next sections we will gradually strengthen this result, first still relying on tiling, and then relying on a more involved encoding of Turing machines.

An instance of *tiling* [van Emde Boas, 1997] is constituted by a set $\mathbf{D} = \{D_1, \ldots, D_m\}$ of tile types with an initial tile type D_1, a horizontal adjacency relation H, and a vertical adjacency relation V over $\mathbf{D}$. A $\mathbf{D}$-*tiling* is a total function $T : \mathbb{N} \times \mathbb{N} \longrightarrow \{D_1, \ldots, D_m\}$. Given a instance $\mathbf{T} = \langle\{D_1, \ldots, D_m\}, D_1, H, V\rangle$ of tiling, we call a $\mathbf{D}$-tiling T *correct for* $\mathbf{T}$ if $(T(i,j), T(i+1,j)) \in H$ and $(T(i,j), T(i,j+1)) \in V$, where $i, j \in \mathbb{N}$, and $T(0,0) = D_1$. The (unbounded quadrant) *tiling problem* consists in determining, given an instance of tiling, whether a correct tiling exists. This problem is well known to be undecidable.

Given an instance of the tiling problem, we construct an extended action theory Γ formed by a BAT and state constraints in the form of $\mathcal{ALC}$ TBox concept inclusions. We use one atomic concept G to denote the points in the grid (one coordinate being the object and the other coordinate being the situation), the atomic concepts $D_1, \ldots, D_m$ denoting tile types,

[1] When a TBox is acyclic, it can be treated as a set of abbreviations that can be eliminated w.l.o.g. Hence, in our setting, an acyclic TBox does not need to be treated as a set of state constraints [Gu and Soutchanski, 2010].

plus auxiliary ones $PD_1, \ldots, PD_m$, and one role $Right$ to represent one dimension of the grid. There are no SSAs for the fluents $D_k(x,s)$, $k \in \{1,\ldots,m\}$, but their values are implicitly defined via state constraints. For all the other we have SSA (they form the *frame*). We use a 0-ary action Up for the other dimension of the grid. The action Up is possible in every situation ($Poss(Up, s) \equiv true$). Thus, we build the grid by using the structural component for $Right$, and the temporal component for Up.

$$G \sqsubseteq \exists Right \sqcap \forall Right.G \tag{2}$$

$$G(x, do(a,s)) \equiv a = Up \wedge G(x,s) \tag{3}$$

$$Right(x,y,do(a,s)) \equiv a = Up \wedge Right(x,y,s) \tag{4}$$

To deal with the tile types:

$$D_k \sqsubseteq \neg D_h \qquad \text{for each } k,h \in \{1,\ldots,m\},\ k \neq h \tag{5}$$

$$G \sqsubseteq \bigsqcup_{k \in \{1,\ldots,m\}} D_k \tag{6}$$

To encode the horizontal adjacency relation, we use the DL concept inclusions:

$$D_k \sqsubseteq \bigsqcup_{h:(D_k,D_h)\in H} \forall Right.D_h \qquad k \in \{1,\ldots,m\} \tag{7}$$

To encode the vertical adjacency relation, we use a successor state axiom exploiting the vertical relation V and auxiliary fluents $PD_h(x,s)$ that stand for "Tile D_h is possible at point (x,s)", and a TBox concept inclusion stating that if tile D_h is used then it must be "possible":

$$PD_h(x, do(a,s)) \equiv a = Up \wedge \bigvee_{k:(D_k,D_h)\in V} D_k(x,s) \tag{8}$$

$$D_h \sqsubseteq PD_h \qquad h \in \{1,\ldots,m\} \tag{9}$$

Finally for the initial situation, we have only two facts $G(O_0, S_0)$ and $D_1(O_0, S_0)$, where O_0 is an object constant. (The pair (O_0, S_0) represent the origin of the grid.)

Theorem 1. *Satisfiability of BATs extended with an $\mathcal{ALC}$ TBox as state constraints is undecidable.*

Proof. We use the above construction to show that existence of a correct tiling reduces to satisfiability of an extended BAT. Let $\mathbf{T}$ be an instance of the tiling problem and Γ the corresponding extended action theory.

First, we show how to construct from a correct tiling T for $\mathbf{T}$ a model $\mathcal{M}$ of Γ. The model $\mathcal{M}$ has only one action that interprets Up, has $\mathbb{N}$ as both the object and the situation domain, and is defined as follows:

- $O_0^{\mathcal{M}} = 0$, $S_0^{\mathcal{M}} = 0$, and $do(Up, s)^{\mathcal{M}} = 1 + s^{\mathcal{M}}$;
- $Right^{\mathcal{M}} = \{(x, x+1, y) \mid x, y \in \mathbb{N}\}$;
- $G^{\mathcal{M}} = \{(x,y) \mid x, y \in \mathbb{N}\}$;
- $D_k^{\mathcal{M}} = PD_k^{\mathcal{M}} = \{(x,y) \mid T(x,y) = D_k\}$.

By definition, the correct tiling T respects the horizontal and vertical adjacency relations and the condition on the initial tile. It is easy to see that all axioms of Γ are satisfied in $\mathcal{M}$.

Next, we show how to construct from a model $\mathcal{M}$ of Γ a correct tiling T for $\mathbf{T}$. Specifically, to define $T(x,y)$, for $x, y \in \mathbb{N}$, let $\sigma_y = do(Up, \cdots do(Up,\ S_0)\cdots)^{\mathcal{M}}$ be obtained by applying the Up action y times to the initial situation S_0, and let $o_0, o_1, \ldots, o_x$ be objects such that $o_0 = O_0^{\mathcal{M}}$ and $(o_{i-1}, o_i, \sigma_y) \in Right^{\mathcal{M}}$, for $i \in \{1,\ldots,x\}$. Then $T(x,y) = D_k$, where $(o_x, \sigma_y) \in D_k^{\mathcal{M}}$. Notice that *(i)* The specific choice of $o_1, \ldots, o_x$ is irrelevant, since we quantify universally over the $Right$-successor (see axiom (7)); *(ii)* there is a unique predicate D_k that is true for object o_x in situation σ_y, due to the disjointness axioms (5). It is easy to see that tiling T satisfies the horizontal and vertical adjacency relations, due to the axioms that encode such relations in Γ (in particular, axioms (7), (8), and (9)), and the initial tile condition. Hence, tiling T is correct. □

4 BATs with $\mathcal{AL}$ TBoxes

We now turn to the simpler DL $\mathcal{AL}$, which is a fragment of $\mathcal{ALC}$ where disjunctions and qualified existential are not allowed and negation is applied only to atomic concepts. $\mathcal{AL}$ concepts are build according to the following syntax: $C ::= A \mid \neg A \mid C \sqcap C \mid \exists R \mid \forall R.C$. Because of syntactic restrictions in $\mathcal{AL}$, in the encoding developed above for $\mathcal{ALC}$, the DL inclusion axioms (6) and (7) are not in $\mathcal{AL}$. However, they can be reformulated in $\mathcal{AL}$.

For axiom (6), it is immediate to observe that it is logically equivalent to the $\mathcal{AL}$ axiom:

$$\bigsqcap_{k \in \{1,\ldots,m\}} \neg D_k \sqsubseteq \neg G \tag{10}$$

To encode the horizontal adjacency relation in $\mathcal{AL}$, instead of concept inclusions (7), we introduce new auxiliary concepts A_k, one for each D_k, and use the following $2m$ DL concept inclusions, where $k, h \in \{1,\ldots,m\}$:

$$\bigsqcap_{h:(D_k,D_h)\in H} \neg A_h \sqsubseteq \neg D_k \tag{11}$$

$$A_h \sqsubseteq \forall Right.D_h \qquad \forall Right.D_h \sqsubseteq A_h \tag{12}$$

Theorem 2. *Satisfiability of BATs extended with an $\mathcal{AL}$ TBox as state constraints is undecidable.*

Proof. The claim follows immediately from Theorem 1, by observing that the new axioms (10), (11), and (12) are logically equivalent to the axioms (6), (7) that they replace. □

5 Local Effect, Context-Free BATs with Lightweight DLs TBoxes

Next we consider TBoxes expressed in lightweight DLs. We show undecidability of satisfiability when state constraints are formulated in *DL-Lite*$_{core}$, the simplest logic of the *DL-Lite* family [Calvanese *et al.*, 2007]. The result holds also for $\mathcal{EL}_\bot$. Moreover, while until now we have not considered restrictions on the form of the BAT itself, we show that undecidability holds even for BATs that are local effect [Vassos *et al.*, 2008] and context-free [Reiter, 1991].

We exploit a reduction from the halting problem for Turing machines (TMs), relying on the fact that every TM can be transformed into the following variant. The machine starts

with a blank tape that is infinite in both directions. Initially, the machine's head scans some cell on the tape, and we name this cell 0. If the given machine ever halts in some new cell, then it enters the special halting state. The fluents are the following: (1) $Value(c, v, s)$, stating that the alphabet symbol in tape cell c is v in s. (2) $State(q, s)$, stating that the machine's state is q in s. (3) $Scan(c, s)$, stating that the machine's head is scanning tape cell c in s. (4) $Next(l, r, s)$ relating adjacent cells in the tape (hence represents the tape), and stating that cell l is the one immediately to the left of cell r in s. (5) $VT(x, s)$, stating that x is a tape cell belonging to the portion of the tape visited so far, in s. (6) $NT(x, s)$ (disjoint from VT), stating that x is a (non-visited) tape cell (immediately) beyond the visited portion of the tape in s. (7) $Halt(x, s)$, stating that the TM halts at cell x (entering the halting state, encoded by the constant Q_H) in s.

For each particular TM, we assume it is specified by a finite set of situation independent facts $TM(q_1, v_1, q_2, v_2, m)$ encoding the TM transitions. Hence, q_1, q_2 range over the (finitely many) TM states, v_1, v_2 range over the (finitely many) tape alphabet symbols, and m ranges over $\{L, R\}$. [2]

There are three actions, all encoding the TM transitions, but taking into account the position of the head with respect to the visited portion of the tape. Action $Trans(l, c, r, q_1, v_1, q_2, v_2, m)$ means that the TM scanning symbol v_1 in tape cell c when in state q_1, enters state q_2, writes tape symbol v_2 to cell c (possibly replacing symbol v_1 that was there previously) and the head makes transition m to another cell by moving either left or right. Cells l (read as "left of the current cell") and r (read as "right of the current cell") are adjacent to c. Actions $TransL(ll, l, c, r, q_1, v_1, q_2, v_2)$ and $TransR(l, c, r, rr, q_1, v_1, q_2, v_2)$ are similar to $Trans$, but take into account that the head is positioned on the left (resp., right) border of the portion of the tape visited so far and the move is just outside the current border. Moreover, an additional adjacent cell ll (resp., rr) is remembered for setting the new border when $TransL$ (resp., $TransR$) become possible. The encoding as an extended action theory is as follows.

Precondition Axioms: Let the TM have a transition $(q_1, v_1) \xrightarrow{m} (q_2, v_2)$, i.e., $TM(q_1, v_1, q_2, v_2, m)$ holds, where $m \in \{L, R\}$, and let the tuple $\langle l, c, r\rangle$ represent a block of three consecutive cells centered around the cell c under the machine's head. The action $Trans(l, c, r, q_1, v_1, q_2, v_2, m)$ is possible in situation s iff the TM is in state q_1, the head is scanning cell c, v_1 is the alphabet symbol in c, the tape cells l, c, and r are next to each other, and the position to which the head moves has been already visited. Actions $TransL(ll, l, c, r, q_1, v_1, q_2, v_2)$ (resp., $TransR(l, c, r, rr, q_1, v_1, q_2, v_2)$) are analogous, with the proviso that the move is to the left (resp., right) and the cell to which the head moves has never been visited before:

$$\begin{aligned}
&Poss(\,Trans(l, c, r, q_1, v_1, q_2, v_2, m), s) \equiv\\
&\quad TM(q_1, v_1, q_2, v_2, m) \wedge State(q_1, s) \wedge Scan(c, s) \wedge\\
&\quad Value(c, v_1, s) \wedge Next(l, c, s) \wedge Next(c, r, s) \wedge\\
&\quad ((m = L \wedge VT(l, s)) \vee (m = R \wedge VT(r, s)))\\
&Poss(\,TransL(ll, l, c, r, q_1, v_1, q_2, v_2), s) \equiv\\
&\quad TM(q_1, v_1, q_2, v_2, L) \wedge State(q_1, s) \wedge Scan(c, s) \wedge\\
&\quad Value(c, v_1, s) \wedge Next(ll, l, s) \wedge Next(l, c, s) \wedge\\
&\quad Next(c, r, s) \wedge NT(l, s)\\
&Poss(\,TransR(l, c, r, rr, q_1, v_1, q_2, v_2), s) \equiv\\
&\quad TM(q_1, v_1, q_2, v_2, R) \wedge State(q_1, s) \wedge Scan(c, s) \wedge\\
&\quad Value(c, v_1, s) \wedge Next(l, c, s) \wedge Next(c, r, s) \wedge\\
&\quad Next(r, rr, s) \wedge NT(r, s)
\end{aligned}$$

Successor State Axioms (all context-free): The $Next$ fluent (and hence the tape) is preserved across situations:

$$Poss(a, s) \rightarrow \big(Next(l, r, do(a, s)) \equiv Next(l, r, s)\big)$$

Cell x becomes visited only after a $TransL$ or $TransR$ action. Once a cell has been visited, it cannot become non-visited:

$$\begin{aligned}
&Poss(a, s) \rightarrow \big(VT(x, do(a, s)) \equiv\\
&\quad (\exists ll, c, r, q_1, q_2, v_1, v_2.(a = TransL(ll, x, c, r, q_1, v_1, q_2, v_2))) \vee\\
&\quad (\exists l, c, rr, q_1, q_2, v_1, v_2.(a = TransR(l, c, x, rr, q_1, v_1, q_2, v_2))) \vee\\
&\quad VT(x, s)\big)
\end{aligned}$$

Cell x becomes NT after a $TransL$ or $TransR$ transition over the border, and it ceases to be NT if the head visits it:

$$\begin{aligned}
&Poss(a, s) \rightarrow \big(NT(x, do(a, s)) \equiv\\
&\quad (\exists l, c, r, q_1, q_2, v_1, v_2.(a = TransL(x, l, c, r, q_1, v_1, q_2, v_2))) \vee\\
&\quad (\exists l, c, r, q_1, q_2, v_1, v_2.(a = TransR(l, c, r, x, q_1, v_1, q_2, v_2))) \vee\\
&\qquad (NT(x, s) \wedge \neg\exists ll, l, c, r, rr, q_1, q_2, v_1, v_2.(\\
&\qquad\quad a = TransL(ll, x, c, r, q_1, v_1, q_2, v_2) \vee\\
&\qquad\quad a = TransR(l, c, x, rr, q_1, v_1, q_2, v_2))))\big)
\end{aligned}$$

The head can scan only one cell at a time. The head will be scanning the tape cell x in $do(a, s)$ iff it was scanning c in s and moves either left or right to the adjacent cell x:

$$\begin{aligned}
&Poss(a, s) \rightarrow \big(Scan(x, do(a, s)) \equiv\\
&\quad \exists ll, l, c, r, rr, q_1, q_2, v_1, v_2.(\\
&\qquad a = Trans(x, c, r, q_1, v_1, q_2, v_2, L) \vee\\
&\qquad a = Trans(l, c, x, q_1, v_1, q_2, v_2, R) \vee\\
&\qquad a = TransL(ll, x, c, r, q_1, v_1, q_2, v_2) \vee\\
&\qquad a = TransR(l, c, x, rr, q_1, v_1, q_2, v_2)))\big)
\end{aligned}$$

The SSAs for the fluents $State(q, s)$ consider that the TM goes into the state q by making a transition into this state, or it remains in state q if there is no transition out:

$$\begin{aligned}
&Poss(a, s) \rightarrow \big(State(q, do(a, s)) \equiv\\
&\quad \exists ll, l, c, r, rr, q_1, v_1, v_2, m.(\\
&\qquad a = Trans(l, c, r, q_1, v_1, q, v_2, m) \vee\\
&\qquad a = TransL(ll, l, c, r, q_1, v_1, q, v_2) \vee\\
&\qquad a = TransR(l, c, r, rr, q_1, v_1, q, v_2)) \vee\\
&\quad (State(q, s) \wedge \neg\exists ll, l, c, r, rr, q_1, v_1, v_2, m.\,(\\
&\qquad a = Trans(l, c, r, q, v_1, q_1, v_2, m) \vee\\
&\qquad a = TransL(ll, l, c, r, q, v_1, q_1, v_2) \vee\\
&\qquad a = TransR(l, c, r, rr, q, v_1, q_1, v_2))))\big)
\end{aligned}$$

The SSA for $Value(v, c, s)$ is very similar to the one for

[2] Note that the predicate TM can be trivially transformed into a fluent, by asserting the TM transitions as facts in S_0 that no action can change.

$State(q, s)$:

$$\begin{array}{l} Poss(a,s) \rightarrow \big(Value(c,v,do(a,s)) \equiv \\ \quad \exists ll, l, r, rr, q_1, v_1, q_2, m.(\\ \qquad a = Trans(l,c,r,q_1,v_1,q,v,m) \vee \\ \qquad a = TransL(ll,l,c,r,q_1,v_1,q,v) \vee \\ \qquad a = TransR(l,c,r,rr,q_1,v_1,q,v)) \vee \\ \quad (Value(c,v,s) \wedge \neg \exists ll, l, r, rr, q_1, v_1, q_2, m.(\\ \qquad a = Trans(l,c,r,q_1,v_1,q,v,m) \vee \\ \qquad a = TransL(ll,l,c,r,q_1,v_1,q,v) \vee \\ \qquad a = TransR(l,c,r,rr,q_1,v_1,q,v)))\big) \end{array}$$

The next axiom states that the TM halts when it enters state Q_H.

$$\begin{array}{l} Poss(a,s) \rightarrow \big(Halt(x,do(a,s)) \equiv \exists ll, l, c, r, rr, q_1, v_1, v_2.(\\ \quad a = Trans(x,c,r,q_1,v_1,Q_H,v_2,L) \vee \\ \quad a = Trans(l,c,x,q_1,v_1,Q_H,v_2,R) \vee \\ \quad a = TransL(ll,x,c,r,q_1,v_1,Q_H,v_2) \vee \\ \quad a = TransR(l,c,x,rr,q_1,v_1,Q_H,v_2))\big). \end{array}$$

TBox (formulated in *DL-Lite*$_{core}$):

$$\begin{array}{rcl} Scan & \sqsubseteq & VT \\ VT & \sqsubseteq & \neg NT \\ Halt & \sqsubseteq & \neg VT \end{array} \qquad \begin{array}{rcl} \exists Next^- & \sqsubseteq & \exists Next \\ \exists Next & \sqsubseteq & \exists Next^- \end{array}$$

Initial situation description: It includes only the following positive facts[3]:

$$Scan(0,S_0),\ Value(0,\mathtt{B},S_0),\ Next(-1,0,S_0),$$
$$Next(0,1,S_0),\ NT(-1,S_0),\ NT(1,S_0),\ State(Q_0,S_0)$$

The extended action theory above guarantees that in every model, there is a possibly infinite sequence of transitions corresponding to the TM transitions. As these transition occur, they force the existence of an expanding sequence of objects, correctly corresponding to the tape cells, connected by the $Next$ relation[4]. These objects and their properties are preserved by the solution to the frame problem provided by the SSAs. In the case where the TM never halts, we get an unbounded tape. On the other hand, if the TM enters the halting state, we get a contradiction due to the fact that the object representing the tape cell under the head is forced to become simultaneously an instance of $Halt$ and VT, which are disjoint. Hence, we get that the action theory above is satisfiable if and only the TM never halts. Note that *all fluents* are determined by the SSAs, i.e., are in the *frame*. This means that it is undecidable even to determine whether SSAs are consistent with TBox state constraints.

Theorem 3. *Satisfiability of local effect, context free BATs extended with a DL-Lite*$_{core}$ *TBox as state constraints is undecidable.*

The construction can be modified to prove undecidability also for $\mathcal{EL}_\bot$. The main difference is that, due to the lack of inverse roles, we need to work with a TM whose tape is infinite in only one direction, requiring more bookkeeping to deal with the beginning of the tape.

[3]By UNA, all finitely many constants, including those in the situation independent facts encoding $TM(\cdots)$, are pairwise disjoint.

[4]Notice that, besides the part correctly encoding the TM computation, the model may contain also spurious objects and tuples.

Theorem 4. *Satisfiability of local effect, context free BATs extended with an $\mathcal{EL}_\bot$ TBox as state constraints is undecidable.*

We observe that [Reiter, 2001] separates action executability (possible situations) from satisfiability. To do so, it uses a variant of SSAs that are not conditioned on $Poss$. In this case, undecidability of satisfiability cannot be shown along the lines above. However, an analogous construction can be used to show that the executability problem, which focuses only on possible actions, remains undecidable. Obviously, if the theory is inconsistent checking executability makes no sense.

6 Conclusion

In this paper, we have shown that the simplest kind of situation calculus action theories (context free and local effect BATs) extended with state constraints TBoxes expressed in the simplest DLs, even for the simplest form of reasoning (satisfiability) become undecidable. From our constructions it appears that the source of these undecidability results is the very solution to the frame problem, as provided by the SSAs in BATs, combined with the possibility of having in the TBox cyclic assertions. The latter is a crucial feature in description logic ontologies, which however may enforce the existence of infinitely many new objects in the state. It remains to be seen if limiting TBoxes to even less expressive ontology languages would preserve decidability. In particular, we observe that in all our proofs, we use TBoxes with cycles involving assertions of the form $A \sqsubseteq \exists R$. If we disallow these constraints, e.g., by restricting the TBox language to (the DL fragment of) RDFS [Franconi *et al.*, 2013], we break the crux of the above undecidability proofs, and potentially decidability can be regained.[5]

In general, the results in this paper call for alternative approaches in combining actions and ontologies, such as those in [Liu *et al.*, 2006; Eiter *et al.*, 2008; Yehia and Soutchanski, 2012]. One approach that has been proved quite robust in combining temporal logics and description logics is the one based on applying temporal operators only to description logic *axioms*. This is related to the so-called *Levesque's functional view* of KBs [Levesque, 1984], which sees a KB (the ontology) as a system that allows for two kinds of operations: ASK, which returns the (certain) answer to queries, and TELL, which produces a new KB as the result of the application of an atomic action (this is also related to update [Winslett, 1988]). In particular approaches like [Baader *et al.*, 2012; Calvanese *et al.*, 2011] demonstrate that under this view even verification of sophisticated temporal properties (e.g., formulated in first-order based variants of the mu-calculus) over ontologies, ranging from lightweight to very expressive ones, becomes decidable under interesting

[5]Cycles involving assertions of the form $A \sqsubseteq \exists R$ allow for generating infinite chains of objects. Notice, however, that all the DLs we considered here, when taken in isolation, have the *finite model property*: that is even if there exists an infinite model formed by chains, there are other models in which these chains are actually finite loops. We use inertia embedded in the SSAs to indirectly disallow such finite loops.

general conditions. These ideas may be applied to combining ontologies and action theories as well, as shown in, e.g., [Bagheri Hariri *et al.*, 2013b].

References

[Artale *et al.*, 2013] A. Artale, R. Kontchakov, F. Wolter, and M. Zakharyaschev. Temporal description logics for ontology-based data access. In *Proc. of IJCAI*, 2013.

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

[Baader *et al.*, 2008] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED DC*, 2008.

[Baader *et al.*, 2012] F. Baader, S. Ghilardi, and C. Lutz. LTL over description logic axioms. *ACM TOCL*, 13(3), 2012.

[Bagheri Hariri *et al.*, 2013a] B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, and M. Montali. Verification of relational data-centric dynamic systems with external services. In *Proc. of PODS*, 2013.

[Bagheri Hariri *et al.*, 2013b] B. Bagheri Hariri, D. Calvanese, M. Montali, G. De Giacomo, R. De Masellis, and P. Felli. Description logic Knowledge and Action Bases. *JAIR*, 46:651–686, 2013.

[Belardinelli *et al.*, 2012] F. Belardinelli, A. Lomuscio, and F. Patrizi. An abstraction technique for the verification of artifact-centric systems. In *Proc. of KR*, 2012.

[Brachman and Levesque, 1984] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of AAAI*, 1984.

[Calvanese *et al.*, 2007] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2011] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Actions and programs over description logic knowledge bases: A functional approach. In *Knowing, Reasoning, and Acting: Essays in Honour of Hector Levesque*. College Publications, 2011.

[Calvanese *et al.*, 2013] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *AIJ*, 195:335–360, 2013.

[De Giacomo *et al.*, 2012] G. De Giacomo, Y. Lespérance, and F. Patrizi. Bounded Situation Calculus action theories and decidable verification. In *Proc. of KR*, 2012.

[Eiter *et al.*, 2008] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. *AIJ*, 172(12-13):1495–1539, 2008.

[Franconi *et al.*, 2013] E. Franconi, C. Gutierrez, A. Mosca, G. Pirrò, and R. Rosati. The logic of extensional RDFS. In *Proc. of ISWC*, 2013.

[Gabbay *et al.*, 2003] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*, volume 148 of *Studies in Logic*. Elsevier, 2003.

[Gu and Soutchanski, 2010] Y. Gu and M. Soutchanski. A description logic based Situation Calculus. *AMAI*, 58(1–2):3–83, 2010.

[Levesque, 1984] H. J. Levesque. Foundations of a functional approach to knowledge representation. *AIJ*, 23:155–212, 1984.

[Lin and Reiter, 1994] F. Lin and R. Reiter. State constraints revisited. *JLC*, 4(5):655–678, 1994.

[Lin and Soutchanski, 2011] F. Lin and M. Soutchanski. Causal theories of actions revisited. In *Proc. of AAAI*, 2011.

[Lin, 1995] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI*, 1995.

[Liu *et al.*, 2006] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Reasoning about actions using description logics with general TBoxes. In *Proc. of JELIA*, volume 4160 of *LNCS*. Springer, 2006.

[Martin and others, 2004] D. Martin et al. Bringing semantics to web services: The OWL-S approach. In *Proc. of the 1st Int. Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, 2004.

[Motik and others, 2012] B. Motik et al. OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C, December 2012. http://www.w3.org/TR/owl2-profiles/.

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

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Describing & Implementing Dynamical Syst.* MIT Press, 2001.

[van Emde Boas, 1997] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture Notes in Pure and Applied Mathematics*, pages 331–363. Marcel Dekker Inc., 1997.

[Vassos *et al.*, 2008] S. Vassos, G. Lakemeyer, and H. J. Levesque. First-order strong progression for local-effect basic action theories. In *Proc. of KR*, 2008.

[Winslett, 1988] M. Winslett. A model-based approach to updating databases with incomplete information. *ACM TODS*, 13(2):167–196, 1988.

[Wolter and Zakharyaschev, 1999] F. Wolter and M. Zakharyaschev. Temporalizing description logic. In *Frontiers of Combining Systems*, pages 379–402. Wiley, 1999.

[Yehia and Soutchanski, 2012] W. Yehia and M. Soutchanski. Towards an expressive decidable logical action theory. In *Proc. of DL*, volume 846 of *CEUR,* ceur-ws.org, 2012.