Bounded Situation Calculus Action Theories (Extended Abstract)

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

Yves Lespérance
York University
Toronto, Canada
lesperan@cse.yorku.ca

Fabio Patrizi
Sapienza Università di Roma
Roma, Italy
patrizi@dis.uniroma1.it

Abstract

We define a notion of bounded action theory in the situation calculus, where the theory entails that in all situations, the number of ground fluent atoms is bounded by a constant. Such theories can still have an infinite domain and an infinite set of states. We argue that such theories are fairly common in applications, either because facts do not persist indefinitely or because one eventually forgets some facts, as one learns new ones. We discuss various ways of obtaining bounded action theories. The main result of the paper is that verification of an expressive class of first-order μ-calculus temporal properties in such theories is in fact decidable. This paper is an abridged version of (De Giacomo, Lespérance, and Patrizi 2012).

Introduction

The Situation Calculus (McCarthy and Hayes 1969; Reiter 2001) has proved to be an invaluable formal tool for understanding the subtle issues involved in reasoning about action. Its comprehensiveness allows us to place all aspects of dynamic systems in perspective. Basic action theories let us capture change as a result of actions in the system (Reiter 1991), while high level languages such as Golog (Levesque et al. 1997) and ConGolog (De Giacomo, Lespérance, and Levesque 2000) support the representation of processes over the dynamic system. Aspects such as knowledge and sensing (Scherl and Levesque 2003), probabilities and utilities (Boutilier et al. 2000), and preferences (Bienvenu, Fritz, and McIlraith 2006), have all been addressed.

The price of such a generality is that decidability results for reasoning in the situation calculus are rare, e.g., (Ternovskaia 1999) for an argument-less fluents fragment, and (Gu and Soutchanski 2007) for a description logic like 2 variables fragment. Obviously, we have the major feature of being able to rely on regression to reduce reasoning about a given future situation to reasoning about the initial situation (Reiter 2001). Generalizations of this basic result such as just-in-time histories (De Giacomo and Levesque 1999) can also be exploited. However, when we move to temporal properties, virtually all results are based on assuming a finite number of states, although there are exceptions such as (Claßen and Lakemeyer 2008; De Giacomo, Lespérance, and Pearce 2010), who develop incomplete fixpoint approximation-based methods.

Here, we present an important new result on decidability of the situation calculus, showing that verification of *bounded action theories* is *decidable*. *Bounded action theories* are basic action theories (Reiter 2001), where it is entailed that in all situations, the number of ground fluent atoms is bounded. In such theories, the object domain remains nonetheless infinite, as is the domain of situations.

But why should we believe that practical domains conform to this boundedness assumption? While it is often assumed that the law of inertia applies and that ground fluent atoms persist indefinitely in the absence of actions that affect them, we all know that pretty much everything eventually decays and changes. We may not even know how the change may happen, but nevertheless know that it will. Another line of argument for boundedness is epistemic. Agents remember facts that they use and periodically try to confirm them, often by sensing. A fact that never gets used is eventually forgotten. If a fact can never be confirmed, it may be given up as too uncertain. Given this, it seems plausible that an agent's knowledge would always remain bounded.

While these philosophical arguments are interesting and relate to some deep questions about knowledge representation, one may take a more pragmatic stance, and this is what we do here. We identify some interesting classes of bounded action theories and show how they can model typical example domains. We also show how we can transform arbitrary basic action theories into bounded action theories, either by blocking actions that would exceed the bound, or by having persistence (frame axioms) apply only for a bounded number of steps.

The main result of the paper is that verification of an expressive class of first-order (FO) μ-calculus temporal properties in bounded action theories is in fact decidable. This means that we can check whether a system or process specified over such a theory satisfies some specification even if we have an infinite domain and an infinite set of situations or states. In a nutshell, we prove our results by focussing on the *active domain* of situations, i.e., the set of objects for which some atomic fluent hold; we know that the set of such active objects is bounded. We show that essentially we can abstract situations whose active domains are *isomorphic* into a single state, and thus, by suitably abstracting also actions, we can obtain an *abstract finite transition system* that *satisfies exactly the same formulas* of our variant of the μ-calculus.

This work is of interest not only for AI, but also for other areas of CS. In particular, there has recently been some attention paid in the field of business processes and services to including data into the analysis of processes (Hull 2008; Gerede and Su 2007; Dumas, van der Aalst, and ter Hofstede 2005). Interestingly, while we have verification tools that are quite good for dealing with data and processes separately, when we consider them together, we get infinite-state transition systems, which resist classical model checking approaches to verification. Lately, there has been some work on developing verification techniques that can deal with such infinite-state processes (Deutsch et al. 2009; Bagheri Hariri et al. 2011; Belardinelli, Lomuscio, and Patrizi 2011). In particular (Belardinelli, Lomuscio, and Patrizi 2011) brings forth the idea of exploiting state boundeness to get decidability for CTL. In this paper, we build on this idea, making it flourish in the general setting of the situation calculus, extending the verification method to the μ-calculus, allowing for incomplete information, and exploiting the richness of the situation calculus for giving sufficient conditions for boundedness that can easily be used in practice.

This paper is an abridged version of (De Giacomo, Lespérance, and Patrizi 2012).

Preliminaries

The *situation calculus* (McCarthy and Hayes 1969; Reiter 2001) is a sorted predicate logic language for representing and reasoning about dynamically changing worlds. All changes to the world are the result of *actions*, which are terms in the language. We denote action variables by lower case letters a, action types by capital letters A, and action terms by α, possibly with subscripts. A possible world history is represented by a term called a *situation*. The constant S_0 is used to denote the initial situation where no actions have yet been performed. Sequences of actions are built using the function symbol do, where $do(a, s)$ denotes the successor situation resulting from performing action a in situation s. Besides actions and situations, there is also the sort of *objects* for all other entities. Predicates and functions whose value varies from situation to situation are called *fluents*, and are denoted by symbols taking a situation term as their last argument (e.g., $Holding(x, s)$). For simplicity, and w.l.o.g., we assume that there are no functions other than constants and no non-fluent predicates. We denote fluents by F and the finite set of primitive fluents by $\mathcal{F}$. The arguments of fluents (appart the last argument which is of sort situation) are assumed to be of sort object.

Within the language, one can formulate action theories that describe how the world changes as the result of the available actions. Here, we concentrate on *basic action theories* as proposed in (Pirri and Reiter 1999; Reiter 2001). We also assume that there is a *finite number of action types*. Moreover, we assume that the terms of object sort are in fact a countably infinite set $\mathcal{N}$ of standard names for which we have the unique name assumption and domain closure. As a result a basic action theory $\mathcal{D}$ is the union of the following disjoint sets: the foundational, domain independent, (second-order, or SO) axioms of the situation calculus (Σ); (FO) precondition axioms stating when actions can be legally performed ($\mathcal{D}_{poss}$); (FO) successor state axioms describing how fluents change between situations ($\mathcal{D}_{ssa}$); (FO) unique name axioms for actions and (FO) domain closure on action types ($\mathcal{D}_{ca}$); (SO) unique name axioms and domain closure for object constants ($\mathcal{D}_{coa}$); and axioms describing the initial configuration of the world ($\mathcal{D}_0$). A special predicate *Poss*(a, s) is used to state that action a is executable in situation s; precondition axioms in $\mathcal{D}_{poss}$ characterize this predicate. The abbreviation $Executable(s)$ means that every action performed in reaching situation s was possible in the situation in which it occured. In turn, successor state axioms encode the causal laws of the world being modeled; they take the place of the so-called effect axioms and provide a solution to the frame problem.

One of the key features of basic action theories is the existence of a sound and complete *regression mechanism* for answering queries about situations resulting from performing a sequence of actions (Pirri and Reiter 1999; Reiter 2001). In a nutshell, the regression operator $\mathcal{R}^*$ reduces a formula ϕ about some future situation to an equivalent formula $\mathcal{R}^*[\phi]$ about the initial situation S_0, by basically substituting fluent relations with the right-hand side formula of their successor state axioms. Here, we shall use a simple *one-step only* variant $\mathcal{R}$ of the standard regression operator $\mathcal{R}^*$ for basic action theories. Let $\phi(do(\alpha, s))$ be a formula uniform in the situation $do(\alpha, s)$. In essence, a formula $\phi(s)$ is uniform in a situation term s if s is the only situation term it contains; see (Reiter 2001) for a formal definition. Then $\mathcal{R}[\phi(do(\alpha, s))]$ stands for the *one-step* regression of ϕ through the action term α, which is itself a formula uniform in s.

In most of this paper, we assume complete information about S_0, and view $\mathcal{D}_0$ as a finite set of facts (under the closed world assumption (CWA) (Reiter 1982)) that we call the *initial database*. At the end, we relax this assumption and show that our results can be generalized to the incomplete information case.

Bounded Action Theories

Let b be some natural number. We use the notation $|\{\vec{x} \mid \phi(\vec{x})\}| \geq b$ to stand for the FOL formula:

$$\exists \vec{x}_1, \ldots, \vec{x}_b . \phi(\vec{x}_1) \wedge \cdots \wedge \phi(\vec{x}_b) \wedge \bigwedge_{i,j \in \{1,\ldots,b\}, i \neq j} \vec{x}_i \neq \vec{x}_j .$$

We also define $(|\{\vec{x} \mid \phi(\vec{x})\}| < b) \doteq \neg(|\{\vec{x} \mid \phi(\vec{x})\}| \geq b)$.

Using this, we define the notion of a fluent $F(\vec{x}, s)$ in situation s being *bounded* by a natural number b as follows:

$$Bounded_{F,b}(s) \doteq |\{\vec{x} \mid F(\vec{x}, s)\}| < b.$$

The notion of situation s being bounded by a natural number b is defined as follows:

$$Bounded_b(s) \doteq \bigwedge_{F \in \mathcal{F}} Bounded_{F,b}(s).$$

We say that an action theory $\mathcal{D}$ is *bounded* by b if

$$\mathcal{D} \models \forall s. Executable(s) \supset Bounded_b(s).$$

We shall see that for bounded action theories verification of sophisticated temporal properties is decidable.

Obtaining Bounded Action Theories by Blocking

We observe that the formula $Bounded_b(s)$ is an FO formula uniform in s and hence it is regressable for basic action theories. This allows us to introduce a first interesting class of bounded action theories. Indeed, from any basic action theory, we can immediately obtain a bounded action theory by simply blocking the execution of actions whenever the result would exceed the bound.

Let $\mathcal{D}$ be a basic action theory. We define the bounded basic action theory $\mathcal{D}_b$ by replacing each action precondition axiom in $\mathcal{D}$ of the form $Poss(a(\vec{x}), s) \equiv \Phi(\vec{x}, s)$ by a precondition axiom of the form

$$Poss(a(\vec{x}), s) \equiv \Phi(\vec{x}, s) \wedge \mathcal{R}[Bounded_b(do(a(\vec{x}), s))]$$

Theorem 1 *Let $\mathcal{D}$ be a basic action theory with the initial database $\mathcal{D}_0$ such that $\mathcal{D}_0 \models Bounded_b(S_0)$, for some b, and let $\mathcal{D}_b$ be the basic action theory obtained as discussed above. Then, $\mathcal{D}_b$ is bounded by b.*

Proof (sketch). By induction on executable situations. $\square$

Example 1 Suppose that we have a camera on a cell phone or PDA. We could model the storage of photos on the device using a fluent $PhotoStored(p, s)$, meaning that photo p is stored in the device's memory. Such a fluent might have the following successor state axiom:

$$\begin{array}{l} PhotoStored(p, do(a, s)) \equiv a = takePhoto(p) \\ \qquad \vee\ PhotoStored(p, s) \wedge a \neq deletePhoto(p) \end{array}$$

We may also assume that action $takePhoto(p)$ is always executable and that $deletePhoto(p)$ is executable in s if p is stored in s:

$$Poss(takePhoto(p), s) \equiv True$$
$$Poss(deletePhoto(p), s) \equiv PhotoStored(p, s).$$

Now such a device would clearly have limited capacity for storing photos. If we assume for simplicity that photos come in only one resolution and file size, then we can model this by simply applying the transformation discussed above. This yields the following modified precondition axioms:

$$\begin{array}{l} Poss(takePhoto(p), s) \equiv \\ \qquad |\{p' \mid PhotoStored(p', s)\}| < b - 1 \\ Poss(deletePhoto(p), s) \equiv PhotoStored(p, s) \wedge \\ \qquad |\{p' \mid PhotoStored(p', s)\}| < b + 1. \end{array}$$

The resulting theory is bounded by b (assuming the original theory is bounded by b in S_0). $\square$

Note that this way of obtaining a bounded action theory is far from realistic in modeling the actual constraints on the storage of photos. One could develop a more accurate model, taking into account the size of photos, the memory management scheme used, etc. This would also yield a bounded action theory, though one whose boundedness is a consequence of a sophisticated model of memory capacity.

Example 2 Let's extend the previous example by supposing that the device also maintains a contacts directory. We could model this using a fluent $InPhoneDir(name, number, photo, s)$, with the following successor state axiom:

$$\begin{array}{l} InPhoneDir(na, no, p, do(a, s)) \equiv \\ \quad a = add(na, no, p) \vee InPhoneDir(na, no, p, s) \wedge \\ \qquad a \neq deleteName(na) \wedge a \neq deleteNumber(no) \end{array}$$

We could then apply our transformation to this new theory to obtain a bounded action theory, getting precondition axioms such as the following:

$$\begin{array}{l} Poss(add(na, no, p), s) \equiv PhotoStored(p, s) \wedge \\ \quad |\{p' \mid PhotoStored(p', s)\}| < b \wedge \\ \quad |\{\langle na, no, p\rangle \mid InPhoneDir(na, no, p, s)\}| < b - 1 \end{array}$$

The resulting theory blocks actions from being performed whenever the action would result in a number of tuples in some fluent exceeding the bound. $\square$

We observe that this kind of bounded action theories are really modeling a capacity constraint on every fluent,[1] which may block actions from being executed. As a result, an action may be executable in a situation in the original theory, but not executable in the bounded one. Thus an agent may want to "plan" to find a sequence of actions that would make the action executable again. In general, to avoid dead-ends, one should carefully choose the original action theory on which the bound is imposed, in particular there should always be actions that remove tuples from fluents.

Effect Bounded Action Theories

Let's consider another sufficient condition for boundedness. Recall that the general form of successor state axioms is:

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

We say that fluent F is *effect bounded* if:

$$|\{\vec{x} \mid \Phi^+_F(\vec{x}, a, s) \wedge \neg F(\vec{x}, s)\}| \leq |\{\vec{x} \mid \Phi^-_F(\vec{x}, a, s) \wedge F(\vec{x}, s)\}|,$$

i.e., for every action and situation, the number of tuples added to the fluent is less than or equal to that deleted.

We say that a basic action theory is effect bounded if every fluent $F \in \mathcal{F}$ is effect bounded.

Theorem 2 *Let $\mathcal{D}$ be an effect bounded basic action theory with the initial database $\mathcal{D}_0$ such that $\mathcal{D}_0 \models Bounded_b(S_0)$, for some b. Then $\mathcal{D}$ is bounded by b.*

Proof (sketch). By induction on executable situations. $\square$

Note that if the initial database is a finite set of facts, as we are assuming up to now, then it is guaranteed that there exists a b such that $\mathcal{D}_0 \models Bounded_b(S_0)$.

Example 3 Many axiomatizations of the Blocks World are not effect bounded. For instance, suppose that we have fluents $OnTable(x, s)$, i.e., block x is on the table in situation

[1]The bound b applies to each fluent individually, so the total number of tuples in a situation is bounded by $|\mathcal{F}|b$. One could instead impose a global capacity bound on the total number of tuples in a situation, but this would require addition in the language.

s, and $On(x,y,s)$, i.e., block x is on block y in situation s, with the following successor state axioms:

$$\begin{aligned}&OnTable(x, do(a,s)) \equiv a = moveToTable(x)\\ &\qquad \lor OnTable(x,s) \land \neg\exists y.a = move(x,y)\end{aligned}$$

$$\begin{aligned}&On(x,y,do(a,s)) \equiv a = move(x,y) \lor On(x,y,s) \land\\ &\neg\exists z.(z \neq y \land a = move(x,z)) \land a \neq moveToTable(x)\end{aligned}$$

Then, performing the action $moveToTable(B1)$ will result in a net increase in the number of objects that are on the table (assuming that the action is executable and that $B1$ is not already on the table). Thus, fluent $OnTable$ is not effect bounded in this theory.

However, it is easy to develop an alternative axiomatization of the Blocks World that is effect bounded. Suppose that we use only the fluent $On(x,y,s)$ and the single action $move(x,y)$, where y is either a block or the table, which is denoted by the constant $Table$. We can axiomatize the domain dynamics as follows:

$$\begin{aligned}&On(x,y,do(a,s)) \equiv a = move(x,y)\\ &\qquad \lor On(x,y,s) \land \neg\exists z.(z \neq y \land a = move(x,z))\end{aligned}$$

That is, x is on y after action a is performed in situation s iff a is moving x onto y or x is already on y in situation s and a does not involve moving x onto an object other than y. We say that $move(x,y)$ is executable in situation s iff x is not the table in s, x and y are distinct, x is clear and on something other than y in s, and y is clear unless it is the table in s:

$$\begin{aligned}&Poss(move(x,y),s) \equiv x \neq Table \land x \neq y \land\\ &\quad \neg\exists z.On(z,x,s) \land \exists z.(z \neq y \land On(x,z,s)) \land\\ &\quad (y = Table \lor \neg\exists z.On(z,y,s))\end{aligned}$$

Then it is easy to show that any occurence of $move(x,y)$ in a situation s where the action is executable, adds $\langle x,y\rangle$ to $O = \{\langle x',y'\rangle \mid On(x',y',s)\}$ while deleting $\langle x,y''\rangle$ for some y'' s.t. $y'' \neq y$, leaving $|O|$ unchanged. Note that we must require that x be on something in the action precondition axiom to get this. Any action other than $move(x,y)$ leaves O unchanged. Thus On is effect bounded.

The precondition that x be on something for $move(x,y)$ to be executable means that we cannot move a new unknown block onto another or the table. We must of course impose restrictions on "moving new blocks in" if we want to preserve effect boundedness. One way to do this is to add an action $replace(x,y)$, i.e. replacing x by y. We can specify its preconditions as follows:

$$\begin{aligned}&Poss(replace(x,y),s) \equiv x \neq Table \land y \neq Table \land\\ &\quad x \neq y \land \neg\exists z.On(z,x,s) \land \exists z.On(x,z,s) \land\\ &\quad \neg\exists z.On(z,y,s) \land \neg\exists z.On(y,z,s)\end{aligned}$$

$replace(x,y)$ is executable in situation s iff x and y are not the table and are distinct, x is clear and on something in s, and y is clear and not on something in s. We can modify the successor state axiom for On to be:

$$\begin{aligned}&On(x,y,do(a,s)) \equiv a = move(x,y) \lor\\ &\quad \exists z.(a = replace(z,x) \land On(z,y,s))\\ &\quad \lor On(x,y,s) \land \neg\exists z.(z \neq y \land a = move(x,z)) \land\\ &\qquad \neg\exists z.(z \neq y \land a = replace(x,z))\end{aligned}$$

where $On(x,y)$ becomes true if x replaces z and z was on y in s, and $On(x,y)$ becomes false if z replaces x and x was on y in s. It is straightforward to show that this change leaves On effect bounded. $\square$

Example 4 For another simple example (perhaps more practical), let's look at how we could specify the "favorite web sites" menu of an internet application. We can assume that there is a fixed number of favorite web sites positions on the menu, say 1 to k. We can replace what is at position n on the menu by the URL u by performing the action $replace(n,u)$. This can be axiomatized as follows:

$$\begin{aligned}&FavoriteSites(n,u,do(a,s)) \equiv a = replace(n,u) \lor\\ &\quad FavoriteSites(n,u,s) \land\\ &\qquad \neg\exists u'.(u' \neq u \land a = replace(n,u'))\end{aligned}$$

$$\begin{aligned}&Poss(replace(n,u),s) \equiv\\ &\quad n \in [1..k] \land URL(u) \land \exists u'.FavoriteSites(n,u',s)\end{aligned}$$

It is easy to show that in this axiomatization, $FavoriteSites$ is effect bounded. No action, including $replace(n,u)$, causes the number of instances of the fluent to increase. $\square$

The $FavoriteSites$ fluent is typical of many domain properties/relations, such as the passengers in a plane, the students in a class, or the cars parked in a parking lot, where we can think of the relation as having a finite capacity, and where we can reassign the objects that are in it. In some cases, the capacity bound may be difficult to pin down, e.g., the guests at a wedding, altough the capacity is by no means unbounded. As well, there are definitely examples where we need an unbounded theory, e.g., to model a pushdown automata that can recognize a particular context-free language. The situation calculus is a very expressive language that accomodates this, for instance, it has been used to model Turing machines (Lin and Levesque 1998). One might arguably want an unbounded "favorite sites" menu or contacts directory, although this seems hardly practical. Another interesting question is how such capacity constraints might apply to a complex agent such as a robot that is modeling its environment. Clearly, such a robot would have limitations wrt how many environment features/objects/properties it can memorize and track. Finally, note that the condition $|\{\vec{x} \mid \Phi_F^+(\vec{x},a,s)\}| \leq |\{\vec{x} \mid \Phi_F^-(\vec{x},a,s)\}|$ is not an FO formula and it is difficult (in fact, undecidable) in general to determine whether a basic action theory is effect bounded. But as our examples illustrate, there are many instances where it is easy to show that the bounded effects condition holds.

Fading Fluents Action Theories

Fading fluents action theories are based on the idea that information over time loses strength and fades away unless it is reinforced explicitly. A fading fluents action theory with fading length given by a natural number ℓ is an action theory where a fluent $F(\vec{x},s)$ is defined by making use of some auxiliary fluents $F_i(\vec{x},s)$, for $0 \leq i \leq \ell$ where $F(\vec{x},s) \doteq \bigvee_{0\leq i\leq \ell} F_i(\vec{x},s)$ and the auxiliary fluents have successor state axioms of the following special form:

$$F_\ell(\vec{x},do(a,s)) \equiv \Phi_F^+(\vec{x},a,s) \land |\{\vec{x} \mid \exists a.\Phi_F^+(\vec{x},a,s)\}| < b$$

and for $0 \leq i < \ell$ we have:

$$F_i(\vec{x}, do(a,s)) \equiv \neg\Phi_F^+(\vec{x},a,s) \wedge F_{i+1}(\vec{x},s) \wedge \neg\Phi_F^-(\vec{x},a,s).$$

Thus, tuples are initially added to F_ℓ, and progressively lose their strength, moving from F_i to F_{i-1}, each time an action occurs that does not delete or re-add them. Eventually they move out of F_0 and are forgotten.

- Technically, a fading fluents action theory is a basic action theory having as fluents only the auxiliary fluents.
- It is simple to obtain a fading fluent version of any basic action theory.
- It is often convenient to include explicit refresh actions $refresh_F(\vec{x})$, whose effect, when applied to a situation s, is simple to make $F_\ell(\vec{x}, do(refresh_F(\vec{x},s)))$ true, and $F_i(\vec{x}, do(refresh_F(\vec{x},s)))$ false for $0 \leq i < \ell$. Similarly it may be convenient to include forget actions $forget_F(\vec{x})$, whose effect is to make $F_i(\vec{x}, do(forget_F(\vec{x},s)))$ false, for all i.

Theorem 3 *Let $\mathcal{D}$ be a fading fluents action theory with fading length ℓ and initial database $\mathcal{D}_0$ such that $\mathcal{D}_0 \models Bounded_b(S_0)$, for some b. Then, $\mathcal{D}$ is bounded by b.*

Proof (sketch). By induction on executable situations. For the base case, we have that initially for each fluent, we have at most b facts, hence S_0 is bounded by b. For the inductive case, by the inductive hypothesis we have that $Bounded_b(s)$. Now, take an arbitrary action $a(\vec{t})$, and an arbitrary fluent F. Then: *(i)* $Bounded_{F_\ell,b}(do(a(\vec{t}),s))$, since positive effects are bounded by b in its successor state axiom; and *(ii)* for all $0 \leq i < \ell$, since F_i depends on F_{i+1} in the previous situation in its successor state axioms, we have that $Bounded_{F_i,b}(do(a(\vec{t}),s))$ since $Bounded_{F_{i+1},b}(s)$ and in the worst case the whole extension of F_{i+1} in s is carried over to F_i in $do(a(\vec{t}),s)$. □

Example 5 Imagine a sort of "vacuum cleaner world" where a robotic vacuum cleaner may clean a room or region r. If a room/region is used, then it becomes unclean. We could model this using a fluent $IsClean(r,s)$ with the following successor state axiom:

$$\begin{aligned} IsClean(r, do(a,s)) &\equiv a = clean(r) \\ &\vee IsClean(r,s) \wedge \neg a = use(r) \end{aligned}$$

Clearly, cleanliness is a property that fades over time. By applying the proposed transformation to this specification, we obtain the following:

$$IsClean_\ell(r, do(a,s)) \equiv a = clean(r) \wedge 1 < b$$

and for $0 \leq i < \ell$ we have:

$$\begin{aligned} IsClean_i(r, do(a,s)) &\equiv a \neq clean(r) \\ &\wedge IsClean_{i+1}(r,s) \wedge a \neq use(r) \end{aligned}$$

This is a somewhat more realistic model where after ℓ steps, we forget about a room being clean. □

Example 6 Consider a robot that can move objects around. We might model this using a fluent $At(objet, location, s)$ with the following successor state axiom:

$$\begin{aligned} &At(o,l,do(a,s)) \equiv a = moveTo(o,l) \vee \\ &\quad a = observe(o,l) \vee At(o,l,s) \wedge a \neq takeAway(o) \wedge \\ &\quad \neg\exists l'.l' \neq l \wedge (a = moveTo(o,l') \vee a = observe(o,l')) \end{aligned}$$

Here, $moveTo(o,l)$ represents the robot's moving object o to location l. We also have an action $observe(o,l)$ of observing that object o is at location l, a kind of exogenous action that might be produced by the robot's sensors. As well, we have another exogenous action $takeAway(o)$, representing another agent's taking object o to an unknown location l. If the world is dynamic, most objects would not remain where they are indefinitely, even if the robot is unaware of anyone moving them. By applying the proposed transformation to this specification, we obtain a theory where information about the location of objects fades unless it is refreshed by the robot's observations or actions. After ℓ steps, the robot forgets the location of an object that it has not observed or moved (moreover, this happens immediately if the object is taken away by another agent). □

Example 7 As a final example, consider a softbot that keeps track of which hosts are online. We might model this using a fluent $NonFaulty(host,s)$ with the following successor state axiom:

$$\begin{aligned} NonFaulty(h, do(a,s)) &\equiv a = pingS(h) \\ &\vee NonFaulty(h,s) \wedge a \neq pingF(r) \end{aligned}$$

Here the action $pingS(h)$ means that the host h has been pinged successfully, and the action $pingF(h)$ means that the host h has not responded to a pinging within the allocated time. As time passes, we may not want to assume that currently non-faulty hosts will remain non-faulty. If we apply the proposed transformation to this specification, we obtain a theory where information about hosts being non-faulty fades. The agent must ping the host successfully to maintain its knowledge that the host is non-faulty. □

An interesting natural example of such fading representations is the pheromones left by insects. Note that it is also possible to model fading with time as opposed to fading with the number of actions, though we have to bound how many actions can occur between clock ticks.

Expressing Dynamic Properties

To express properties about Situation Calculus action theories, we introduce a specific logic, inspired by the μ-calculus (Emerson 1996; Stirling 2001), one of the most powerful temporal logics, subsuming both linear time logics, such as LTL and PSL, and branching time logics such as CTL and CTL* (Baier, Katoen, and Guldstrand Larsen 2008). In particular, we introduce a variant of the μ-calculus, called $\mu\mathcal{L}$, whose syntax is as follows:

$$\Phi ::= \varphi \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \langle - \rangle\Phi \mid Z \mid \mu Z.\Phi$$

where φ is an arbitrary closed uniform *situation-suppressed* (i.e., with all situation arguments in fluents suppressed) situation calculus FO formula, and Z is an SO (0-ary) predicate variable. We use the following standard abbreviations: $\Phi_1 \vee \Phi_2 = \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, $[-]\Phi = \neg\langle-\rangle\neg\Phi$, and $\nu Z.\Phi = \neg\mu Z.\neg\Phi[Z/\neg Z]$. As usual in the μ-calculus, formulae of the form $\mu Z.\Phi$ (and $\nu Z.\Phi$) must satisfy *syntactic monotonicity* of Φ wrt Z, which states that every occurrence of the variable Z in Φ must be within the scope of an even number of negation symbols.

The *fixpoint formulas* $\mu Z.\Phi$ and $\nu Z.\Phi$ denote respectively the *least* and the *greatest fixpoint* of the formula Φ seen as a predicate transformer $\lambda Z.\Phi$ (their existence is guaranteed by the syntactic monotonicity of Φ). We can express arbitrary temporal/dynamic properties using least and greatest fixpoint constructions. For instance, to say that it is possible to achieve φ, where φ is a closed situation suppressed formula, we use the least fixpoint formula $\mu Z.\varphi \vee \langle-\rangle Z$. Similarly, we can use a greatest fixpoint formula $\nu Z.\varphi \wedge [-]Z$ to express that φ always holds.

Next we turn to semantics. Since $\mu\mathcal{L}$ contains formulae with predicate free variables, given a model $\mathcal{M}$ of an action theory $\mathcal{D}$ with domain $\mathcal{S}$ for sort situation, we introduce a predicate variable valuation $\mathcal{V}$, i.e., a mapping from predicate variables Z to subsets of $\mathcal{S}$. Then we assign semantics to formulae by associating to $\mathcal{M}$ and $\mathcal{V}$ an *extension function* $(\cdot)^{\mathcal{M}}_{\mathcal{V}}$, which maps $\mu\mathcal{L}$ formulae to subsets of $\mathcal{S}$ as inductively defined as follows (we include also key abbreviations for clarity):

$$
\begin{array}{lcl}
(\varphi)^{\mathcal{M}}_{\mathcal{V}} & = & \{s \in \mathcal{S} \mid \mathcal{M} \models \varphi[s]\} \\
(\neg\Phi)^{\mathcal{M}}_{\mathcal{V}} & = & \mathcal{S} - (\Phi)^{\mathcal{M}}_{\mathcal{V}} \\
(\Phi_1 \wedge \Phi_2)^{\mathcal{M}}_{\mathcal{V}} & = & (\Phi_1)^{\mathcal{M}}_{\mathcal{V}} \cap (\Phi_2)^{\mathcal{M}}_{\mathcal{V}} \\
(\langle-\rangle\Phi)^{\mathcal{M}}_{\mathcal{V}} & = & \{s \in \mathcal{S} \mid \exists a.(a,s) \in Poss^{\mathcal{M}} \wedge \\
 & & \qquad do^{\mathcal{M}}(a,s) \in (\Phi)^{\mathcal{M}}_{\mathcal{V}}\} \\
([-]\Phi)^{\mathcal{M}}_{\mathcal{V}} & = & \{s \in \mathcal{S} \mid \forall a.(a,s) \in Poss^{\mathcal{M}} \supset \\
 & & \qquad do^{M}(a,s) \in (\Phi)^{\mathcal{M}}_{\mathcal{V}}\} \\
(Z)^{\mathcal{M}}_{\mathcal{V}} & = & \mathcal{V}(Z) \\
(\mu Z.\Phi)^{\mathcal{M}}_{\mathcal{V}} & = & \bigcap\{\mathcal{E} \subseteq \mathcal{S} \mid (\Phi)^{\mathcal{M}}_{\mathcal{V}[Z/\mathcal{E}]} \subseteq \mathcal{E}\} \\
(\nu Z.\Phi)^{\mathcal{M}}_{\mathcal{V}} & = & \bigcup\{\mathcal{E} \subseteq \mathcal{S} \mid \mathcal{E} \subseteq (\Phi)^{\mathcal{M}}_{\mathcal{V}[Z/\mathcal{E}]}\}
\end{array}
$$

Intuitively, the extension function $(\cdot)^{\mathcal{M}}_{\mathcal{V}}$ assigns to such constructs the following meaning:

- The boolean connectives and quantification over individuals have the expected meaning.
- The extension of $\langle-\rangle\Phi$ consists of the situations s such that for *some* possible successor situation s' , we have that Φ holds in s', while the extension of $[-]\Phi$ consists of the situations s such that for *all* successor situations s', we have that Φ holds in s'.
- The extension of $\mu X.\Phi$ is the *smallest subset* $\mathcal{E}_\mu$ of $\mathcal{S}$ such that, assigning to Z the extension $\mathcal{E}_\mu$, the resulting extension of Φ is contained in $\mathcal{E}_\mu$. That is, the extension of $\mu X.\Phi$ is the *least fixpoint* of the operator $(\Phi)^{\mathcal{M}}_{\mathcal{V}[Z/\mathcal{E}]}$ (here $\mathcal{V}[Z/\mathcal{E}]$ denotes the predicate valuation obtained from $\mathcal{V}$ by forcing the valuation of Z to be $\mathcal{E}$).
- Similarly, the extension of $\nu X.\Phi$ is the *greatest subset* $\mathcal{E}_\nu$ of $\mathcal{S}$ such that, assigning to X the extension $\mathcal{E}_\nu$, the resulting extension of Φ contains $\mathcal{E}_\nu$. That is, the extension of $\nu X.\Phi$ is the *greatest fixpoint* of the operator $(\Phi)^{\mathcal{M}}_{\mathcal{V}[X/\mathcal{E}]}$.

Notice that given a closed uniform situation-suppressed situation calculus formula φ, slightly abusing notation, we denote by $\varphi[s]$ the corresponding formula with situation calculus argument reintroduced and assigned to situation s. Notice also that when a $\mu\mathcal{L}$ formula Φ is closed (wrt predicate variables), its extension $(\Phi)^{\mathcal{M}}_{\mathcal{V}}$ does not depend on the predicate valuation $\mathcal{V}$. The only formulas of interest in verification are those that are closed.

Observe that we do not have actions as parameters of $[-]\cdot$ and $\langle-\rangle\cdot$. However we can easily remember the last action performed, and in fact a finite sequence of previous actions. To do this, for each action $A(\vec{x})$, we introduce a fluent $Last_A(\vec{x}, s)$ with successor state axiom:

$$Last_A(\vec{x}, do(a,s)) \equiv a = A(\vec{x})$$

We can also remember the second last action by introducing fluents $SecondLast_A(\vec{x}, s)$ with successor state axioms:

$$SecondLast_A(\vec{x}, do(a,s)) \equiv Last_A(\vec{x}, s)$$

Similarly for the third last action, etc.

In this way we can store a finite suffix of the history in the current situation and write FO formulas relating the individuals in the parameters of actions occurring in the suffix. E.g., we can write (assuming for simplicity that the mentioned fluents have all the same arity):

$$\mu Z.(\exists\vec{x}.Last_A(\vec{x}) \wedge SecondLast_B(\vec{x})) \vee \langle-\rangle Z,$$

i.e., it is possible to eventually do $B(\vec{x})$ followed by $A(\vec{x})$ for some $\vec{x}$.

Observe also that our $\mu\mathcal{L}$ does not allow for quantification across situations. However the expressiveness of bounded action theories does mitigate this limitation. For instance, we can easily introduce a finite number of "registers", i.e., fluents that store only one tuple, which can be used to store and refer to tuples across situations. We can do this by introducing fluents $Reg_i(\vec{x}, s)$ and two actions $setReg_i(\vec{x})$ and $clearReg_i$ to set and clear the register Reg_i respectively. These are axiomatized as follows:

$$
\begin{array}{c}
Reg_i(\vec{x}, do(a,s)) \equiv a = setReg_i(\vec{x}) \vee \\
\quad Reg_i(\vec{x}, s) \wedge a \neq clearReg_i \\
Poss(setReg_i(\vec{x}), s) \equiv \neg\exists\vec{x}.Reg_i(\vec{x}, s) \\
Poss(clearReg_i, s) \equiv \exists\vec{x}.Reg_i(\vec{x}, s)
\end{array}
$$

For example, we can write (assuming for simplicity that the mentioned fluents have all the same arity):

$$\mu Z.(\exists\vec{x}.Reg_i(\vec{x}) \wedge F(\vec{x}) \wedge \langle-\rangle\exists\vec{y}.Reg_i(\vec{y}) \wedge F'(\vec{y})) \vee \langle-\rangle Z$$

This formula says that there exists a sequence of actions where eventually the tuple referred to by register i has property F and there is an action after which it has property F'.

Verification of Bounded Action Theories

It can be shown that verifying $\mu\mathcal{L}$ temporal properties against bounded action theories is decidable. We first focus on action theories with complete information on the initial situation, described as a (bounded) database. Then, we generalize our results to the cases with incomplete information on the initial situation. Our main result is the following.

Theorem 4 *If $\mathcal{D}$ is a bounded action theory with initial situation described by a (bounded) database, and Φ a closed $\mu\mathcal{L}$ formula, then verifying whether $\mathcal{D} \models \Phi$ is decidable.*

The proof is structured as follows. The first step is to get rid of action terms in formulas, and observe that $\mu\mathcal{L}$ can be equivalently interpreted over a certain kind of transition systems which do not necessarily reflect the tree structure of the situation tree.

In the second step, we introduce the notions of: *active domain* (Abiteboul, Hull, and Vianu 1995), i.e., the domain containing all the objects occurring in the extension of the predicates (fluents at a given situation), *active-domain isomorphism*, i.e., standard isomorphism restricted to active domains, and *active-domain bisimulation*, a variant of standard bisimulation which requires bisimilar states to be active-domain isomorphic. Then we prove that active-domain bisimilar transition systems preserve *domain-independent $\mu\mathcal{L}$ (closed) formulas*, i.e., formulas whose evaluation of FO components depends only on the active domain.

In the third and fundamental step we show how to actually construct an abstract, finite-state transition system that is active-domain bisimilar to the one induced by *the* model of the action theory. We make use of the assumption that situations are bounded, and exploit the specific structure of successor state and action precondition axioms, to devise a bound on the number of distinct objects required to maintain active-domain isomorphisms between states. With this we prove the decidability result for domain-independent $\mu\mathcal{L}$ formulas.

In the final step, we generalize the above result to generic $\mu\mathcal{L}$ formulas, by observing that any FO formula interpreted over standard names admits an equivalent, domain-independent formula –see, e.g., Th. 5.6.3 in (Libkin 2007).

We refer to (De Giacomo, Lespérance, and Patrizi 2012) for the technical details.

Dealing with Incomplete Information

For the case of partial information about the initial situation, assume that $\mathcal{D}_0$ is a set of axioms characterizing a possibly infinite set of bounded initial databases.

Theorem 5 *Consider a b-bounded action theory $\mathcal{D}$ with incomplete information on the initial situation, and let Φ be a $\mu\mathcal{L}$ closed, domain-independent formula. Then, checking whether $\mathcal{D} \models \Phi$ is decidable.*

The result is a consequence of the fact that for a b-bounded action theory, any possible (complete) initial database belongs to one of finitely many distinct isomorphic types, so for each of such types we can arbitrarily select one representative, and then apply Th. 4.

This result, besides stating decidability of the verification problem under incomplete information, provides us with an actual procedure to perform the check.

Conclusion

In this paper, we have defined the notion of bounded action theory in the situation calculus, where the number of ground atomic fluents that are known remains bounded. We have shown that this restriction is sufficient to ensure that verification of an expressive class of temporal properties remains decidable, despite the fact that we have an infinite domain and state space. Our result holds even in the presence of incomplete information. We have also argued that this restriction can be adhered to in practical applications, by identifying interesting classes of bounded action theories and showing that these can be used to model typical example dynamic domains. Decidability is important from a theoretical standpoint, but we stress also that our result is fully constructive being based on a reduction to model checking of an (abstract) finite-state transition system. An interesting future enterprise is to build on such a result to develop an actual situation calculus verification tool.

In future work, we want to do a more systematic investigation of specification patterns for obtaining boundedness. This includes patterns that provide bounded persistence and patterns that model bounded/fading memory. These questions should be examined in light of different approaches that have been proposed for modeling knowledge, sensing, and revision in the situation calculus and related temporal logics (Scherl and Levesque 2003; Demolombe and del Pilar Pozos Parra 2000; Shapiro et al. 2011; van Ditmarsch, van der Hoek, and Kooi 2007).

Acknowledgments. We acknowledge the support of EU Project FP7-ICT ACSI (257593).

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison Wesley.

Bagheri Hariri, B.; Calvanese, D.; De Giacomo, G.; De Masellis, R.; and Felli, P. 2011. Foundations of relational artifacts verification. In *Proc. of BPM'11*, 379–395.

Baier, C.; Katoen, J.-P.; and Guldstrand Larsen, K. 2008. *Principles of Model Checking*. The MIT Press.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2011. Verification of deployed artifact systems via data abstraction. In *Proc. of ICSOC'11*, 142–156.

Bienvenu, M.; Fritz, C.; and McIlraith, S. A. 2006. Planning with qualitative temporal preferences. In *Proc. of KR'06*, 134–144.

Boutilier, C.; Reiter, R.; Soutchanski, M.; and Thrun, S. 2000. Decision-theoretic, high-level agent programming in the situation calculus. In *Proc. of AAAI'00/IAAI'00*, 355–362.

Claßen, J., and Lakemeyer, G. 2008. A logic for non-terminating Golog programs. In *Proc. of KR'08*, 589–599.

De Giacomo, G., and Levesque, H. J. 1999. Projection using regression and sensors. In *Proc. of IJCAI'99*, 160–165.

De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2000. ConGolog, a concurrent programming language based on the situation calculus. *AIJ* 121(1–2):109–169.

De Giacomo, G.; Lespérance, Y.; and Patrizi, F. 2012. Bounded situation calculus action theories and decidable verification. In *Proc. of KR'12*. To appear.

De Giacomo, G.; Lespérance, Y.; and Pearce, A. R. 2010. Situation calculus based programs for representing and reasoning about game structures. In *Proc. of KR'10*, 445–455.

Demolombe, R., and del Pilar Pozos Parra, M. 2000. A simple and tractable extension of situation calculus to epistemic logic. In *Proc. of ISMIS'00*, 515–524.

Deutsch, A.; Hull, R.; Patrizi, F.; and Vianu, V. 2009. Automatic verification of data-centric business processes. In *Proc. of ICDT'09*, 252–267.

Dumas, M.; van der Aalst, W. M. P.; and ter Hofstede, A. H. M. 2005. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons.

Emerson, E. A. 1996. Model checking and the mu-calculus. In *Descriptive Complexity and Finite Models*, 185–214.

Gerede, C. E., and Su, J. 2007. Specification and verification of artifact behaviors in business process models. In *Proc. of ICSOC'07*, 181–192.

Gu, Y., and Soutchanski, M. 2007. Decidable reasoning in a modified situation calculus. In *Proc. of IJCAI'07*, 1891–1897.

Hull, R. 2008. Artifact-centric business process models: Brief survey of research results and challenges. In *OTM 2008 Confederated International Conferences*, volume 5332 of *LNCS*, 1152–1163.

Levesque, H. J.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *JLP* 31:59–84.

Libkin, L. 2007. Embedded finite models and constraint databases. In *Finite Model Theory and Its Applications*. Springer.

Lin, F., and Levesque, H. J. 1998. What robots can do: Robot programs and effective achievability. *Artif. Intell.* 101(1-2):201–226.

McCarthy, J., and Hayes, P. J. 1969. Some Philosophical Problems From the StandPoint of Artificial Intelligence. *Machine Intelligence* 4:463–502.

Pirri, F., and Reiter, R. 1999. Some contributions to the metatheory of the situation calculus. *J. ACM* 46(3):261–325.

Reiter, R. 1982. Towards a logical reconstruction of relational database theory. In *On Conceptual Modeling*, 191–233.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*. Academic Press. 359–380.

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.

Scherl, R. B., and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artif. Intell.* 144(1-2):1–39.

Shapiro, S.; Pagnucco, M.; Lespérance, Y.; and Levesque, H. J. 2011. Iterated belief change in the situation calculus. *Artif. Intell.* 175(1):165–192.

Stirling, C. 2001. *Modal and Temporal Properties of Processes*. Springer.

Ternovskaia, E. 1999. Automata theory for reasoning about actions. In *Proc. of IJCAI'99*, 153–159.

van Ditmarsch, H.; van der Hoek, W.; and Kooi, B. 2007. *Dynamic Epistemic Logic*. Springer.