ECAI 2014
T. Schaub et al. (Eds.)
© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-419-0-369

LTL Verification of Online Executions with Sensing in Bounded Situation Calculus

Giuseppe De Giacomo[1] and **Yves Lespérance**[2] and **Fabio Patrizi**[3] and **Stavros Vassos**[4]

Abstract. We look at agents reasoning about actions from a first-person perspective. The agent has a representation of world as situation calculus action theory. It can perform sensing actions to acquire information. The agent acts "online", i.e., it performs an action only if it is certain that the action can be executed, and collects sensing results from the actual world. When the agent reasons about its future actions, it indeed considers that it is acting online; however only *possible* sensing values are available. The kind of reasoning about actions we consider for the agent is verifying a first-order (FO) variant (without quantification across situations) of linear time temporal logic (LTL) . We mainly focus on bounded action theories, where the number of facts that are true in any situation is bounded. The main results of this paper are: *(i)* possible sensing values can be based on consistency if the initial situation description is FO; *(ii)* for bounded action theories, progression over histories that include sensing results is always FO; *(iii)* for bounded theories, verifying our FO LTL against online executions with sensing is decidable.

1 Introduction

In this paper, we look at agents reasoning about actions from a first-person perspective. In particular, the agent has a computationally grounded model of the world [26] as a situation calculus action theory [16]. This includes a first-order (FO) description of the initial situation and of actions' preconditions and effects (as successor state axioms). Moreover, we have *sensing actions* that do not affect the world state but update the logical theory with sensing results. The agent acts *online* [7, 19], i.e., it performs an action only if it is certain that the action can be executed, and collects sensing results from the actual world. The kind of reasoning about actions we consider for the agent is verifying a FO variant (without quantification across situations) of linear time temporal logic (LTL) [15, 24]. (Notice that FO LTL, though typically over a finite horizon, has been considered often in situation calculus e.g., in [9].)

We focus on *bounded action theories* [5] for which it is certain that the number of tuples that are in the extension of any fluent in any situation is bounded. For bounded action theories, verification of FO mu-calculus properties over *offline executions* without sensing is decidable. In [6], decidability was shown also for verifying such properties over *online executions without sensing*. Here, we consider a linear time setting [24] and show that verification of FO LTL properties against *online executions with sensing* is decidable as well.

The extension to sensing is nontrivial. Indeed, when the agent reasons about its future actions it considers that it acts online, but obviously it cannot know ahead of time the actual results that will come from sensing, so it has to consider *possible* sensing results. Specifically, the agent must consider each possible world it could be in, i.e., each possible model of its action theory, and look at the sensing results that that world would produce. But in general doing this is impractical. A simpler approach is for the agent to assume possible every sensing result that is *consistent* with the action theory and what has been sensed so far. However, as shown in [17, 18], this simplification produces incorrect results in general, as it does not ensure that all information available to the agent is fully taken into account. For example, suppose the agent knows that if it keeps chopping at a tree, it will eventually come down. After any number of chops, it is still consistent that sensing will say that the tree is up. So the agent ends up considering possible a run where it keeps chopping and the tree never falls. The first contribution of the paper is to show that the simplification is in fact *correct* along (infinite) runs, if we start from an initial situation description in first-order logic (FOL), because of compactness of FOL. This result enables the effective verification of FO LTL formulas. Notice that our result is based on runs and hence does not apply to branching-time logics, including mu-calculus.

Then we define *progression* [12] over histories that include sensing results. We show that for bounded action theories such progression over histories is always FO representable. So the belief state of the agent after a sequence of actions with sensing can be represented as a FO theory. Finally, with these two results at hand, we can show that for bounded action theories with sensing, we can faithfully abstract the infinite set of possible runs of the agent into a finite transition system, which gives us an effective way of verifying FO LTL properties, based on automata techniques on infinite objects [23, 24]. Thus for bounded theories, verifying our FO LTL against online executions with sensing is decidable.

Our account is related, but quite different from that in [4], which takes a third-person point of view based on a version of the situation calculus with a knowledge modality and focusses on offline executions. There a more restrictive notion of bounded epistemic action theory is adopted, where the number of tuples that the agent thinks may belong to any given fluent is bounded. Here, we only require that it be entailed that the number of distinct tuples in any fluent is bounded, and the agent need not know anything about which.

2 Situation Calculus

The *situation calculus* [13, 16] 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 and terms by lower case letters a, action types by capital letters A, possibly with subscripts. A possible world history is represented by a term called a *situation*.

[1] DIAG - Sapienza Univ. Rome, Italy, email: degiacomo@dis.uniroma1.it
[2] EECS - York Univ., Toronto, ON, Canada, email: lesperan@cse.yorku.ca
[3] DIAG - Sapienza Univ. Rome, Italy, email: patrizi@dis.uniroma1.it
[4] DIAG - Sapienza Univ. Rome, Italy, email: vassos@dis.uniroma1.it

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 fluents by $\mathcal{F}$. The arguments of fluents (apart 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* (BATs) as proposed in [14, 16]. We also assume that there is a *finite number of action types* $\mathcal{A}$. Moreover, we assume that the terms of object sort are in fact a countably infinite set $\mathcal{N}$ of names for which we have the unique name assumption (which is expressible in FOL).[5] 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 ($\mathcal{D}_{ca}$); (FO) unique name axioms for object constants ($\mathcal{D}_{uo}$); and (FO) 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. Obviously, $\mathcal{D}$ must be consistent for being of interest.

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

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

We can also define $(|\{\vec{x} \mid \phi(\vec{x})\}| < b) \doteq \neg(|\{\vec{x} \mid \phi(\vec{x})\}| \geq b)$. Using this, [5] defines the notion of a fluent $F(\vec{x}, s)$ in situation s being *bounded* by a natural number b as $Bounded_{F,b}(s) \doteq |\{\vec{x} \mid F(\vec{x}, s)\}| < b$ and the notion of situation s being bounded by b: $Bounded_b(s) \doteq \bigwedge_{F \in \mathcal{F}} Bounded_{F,b}(s)$. An action theory $\mathcal{D}$ then is *bounded* by b if $\mathcal{D} \models \forall s. Executable(s) \supset Bounded_b(s)$. [5] shows that for bounded theories, verification of sophisticated temporal properties is decidable. It also identifies interesting classes of such theories. Also, [6] shows that if a theory's initial state description is bounded, then checking whether the action theory is bounded (in all executable situations) is decidable.

We close by stressing that in this view, boundedness is a property of the world not of what the agent knows about the world. See [4] for a different view of boundedness related agent's knowledge.

3 Sensing and Online Execution

Sensing Actions. Assume now that some actions may be used to sense aspects of the agents' environment. These are called *sensing actions* and can be assumed to return a boolean result. Following [10], we assume that the information provided by such sensing actions is specified using a special predicate $SF(a, s)$, which holds if action a returns the binary sensing result 1 in situation s. Using this, we can write *sense fluent axioms* of the form $SF(senseOpen(door), s) \equiv Open(door, s)$, i.e. the action $senseOpen(door)$ returns the result 1 in situation s iff $door$ is open in s. (Non-binary sensing actions can be handled as shown in [21]). From now on, let us assume that our action theories include a set $\mathcal{D}_{sf}$ of such (FO) sense fluent axioms of the form $SF(A(\vec{x}), s) \equiv \phi_A(\vec{x}, s)$, one for each action type A, which characterize SF (for any ordinary action type A that does not involve sensing, we use $SF(A(\vec{x}), s) \equiv \mathtt{true}$). For simplicity, as in [21] we assume that sensing actions only affect what the agent knows, and do not do change the state of the world; any "world changing sensing action" can be modeled as a sensing action followed by an ordinary action.

Histories. To describe a run that includes both actions and their sensing results, we use the notion of *history*. By a history we mean a sequence of pairs (a, v) where a is a ground action term and v is 1 or 0, a sensing result. Intuitively, the history $(a_1, v_1) \cdot \ldots \cdot (a_n, v_n)$ is one where actions $a_1, \ldots, a_n$ happen starting in some initial situation, and each action a_i returns sensing value v_i. The assumption is that if a_i is an ordinary action with no sensing, then $v_i = 1$. Notice that the empty sequence ϵ is a history.

Histories are not terms of the situation calculus. It is convenient, however, to use $end[h]$ as an abbreviation for the situation term called the *end situation* of history h on the initial situation S_0, and defined inductively by: $end[\epsilon] = S_0$ and $end[h \cdot (a, x)] = do(a, end[h])$. We also use $Sensed[h]$ as an abbreviation for a formula of the situation calculus, the *sensing results* of a history, defined inductively by: $Sensed[\epsilon] = \mathtt{true}$, $Sensed[h \cdot (a, 1)] = Sensed[h] \land SF(a, end[h])$, and $Sensed[h \cdot (a, 0)] = Sensed[h] \land \neg SF(a, end[h])$. This formula uses SF to tell us what must be true for the sensing to come out as specified by h starting in the initial situation S_0. Note that if no sensing action is performed along a history h, then $Sensed[h]$ becomes equivalent to $\mathtt{true}$.

We will model what the agent knows after the actions and observations in h have occured using $\mathcal{D} \cup \{Sensed[h]\}$. Note that we are modeling knowledge meta-theoretically and are taking a first-person view of the action theory augmented by sensing results, as representing what the agent knows about the world (see [21] for an alternative third-person view where knowledge is modeled in the language).

Online Execution and Sensing. An *on-line execution* of an agent is a sequence of (ground) actions that are known to be executable. If an action is not executable its effects are unpredictable, and we assume the agent is only interested in performing actions that are known to be executable and have predictable effects. First, observe that sensing actions are just like ordinary actions except for the new information they provide, which is specified by the axioms involving SF. However, whether the agent knows that an action is executable may now depend on the values sensed so far. That is, if h is the history of actions and sensing values obtained so far starting from S_0, then action a can be legally executed in $end[h])$ whenever

$$\mathcal{D} \cup \{Sensed[h]\} \models Poss(a, end[h]).$$

In other words, now we are looking for actions whose preconditions are logically implied by $\mathcal{D}$ together with *the values sensed so far*.

However, one problem is where the new sensing results v in h come from. In the real execution they come from the agent's environment. But when the agent reason about what to do next, it will

[5] Such names are akin to standard names [11], but we do not enforce domain closure which requires second-order logic.

want to check dynamic or temporal properties over its *possible on-line executions*, since these are the only executions that the agent could actually perform. So the question is where the sensing values come from in such possible on-line executions. To capture this we introduce two notions of possible executable histories, one that uses a structure/model of the action theory to represent the agent's environment, and one that uses consistency with the action theory to determine possible sensing results.

First, we look at *executable histories wrt a model $\mathcal{M}$ of an action theory $\mathcal{D}$* . We define the set $Exec_{\mathcal{M}}$ of executable histories h wrt $\mathcal{M}$, inductively as follows:

- $\epsilon \in Exec_{\mathcal{M}}$;
- if $h \in Exec_{\mathcal{M}}$ and $\mathcal{D} \cup \{Sensed(h)\} \models Poss(a, end[h])$ and $\mathcal{M} \models \mathcal{D} \cup \{Sensed(h \cdot (a, v))\}$, then $h \cdot (a, v) \in Exec_{\mathcal{M}}$.

Notice that $Exec_{\mathcal{M}}$ contains all the histories built by ensuring that at every step it is known that the agent is able to execute action a, i.e., $\mathcal{D} \cup \{Sensed(h)\} \models Poss(a, end[h])$, and where sensing results are obtained from the actual model $\mathcal{M}$ corresponding to the real environment, i.e., $\mathcal{M} \models \mathcal{D} \cup \{Sensed(h \cdot (a, v))\}$, which amounts to having $h \cdot (a, 1) \in Exec_{\mathcal{M}}$ only if $\mathcal{M} \models SF(a, end[h])$ and $h \cdot (a, 0) \in Exec_{\mathcal{M}}$ only if $\mathcal{M} \models \neg SF(a, end[h])$.

Now the agent does not really know which model corresponds to the real environment. It only knows what is specified by the action theory itself and the values sensed so far ($\mathcal{D} \cup \{Sensed(h)\}$). So the only thing it can do is to consider all *possible* executable histories $h \in Exec_{\mathcal{M}}$ *for some model $\mathcal{M}$ of $\mathcal{D}$*.

Next, we look at the set $Exec_{\mathcal{D}}$ of *executable histories wrt an action theory $\mathcal{D}$*, where we use consistency to determine possible sensing results. We define the set $Exec_{\mathcal{D}}$ inductively as follows:

- $\epsilon \in Exec_{\mathcal{D}}$;
- if $h \in Exec_{\mathcal{D}}$ and $\mathcal{D} \cup \{Sensed(h)\} \models Poss(a, end[h])$ and $\mathcal{D} \cup \{Sensed(h \cdot (a, v))\}$ is consistent, then $h \cdot (a, v) \in Exec_{\mathcal{D}}$.

Notice that $Exec_{\mathcal{D}}$ contains all the histories built by ensuring that at every step it is known that the agent is able to execute action a ($\mathcal{D} \cup Sensed(h) \models Poss(a, end[h])$) and that the sensing results are consistent with what has been sensed along the history, i.e., $\mathcal{D} \cup Sensed(h \circ (a, v))$ is consistent, which amounts to having $h \cdot (a, 1) \in Exec_{\mathcal{D}}$ only if $\mathcal{D} \cup \{Sensed(h) \wedge SF(a, end[h])\}$ is consistent and $h \cdot (a, 0) \in Exec_{\mathcal{D}}$ only if $\mathcal{D} \cup \{Sensed(h) \wedge \neg SF(a, end[h])\}$ is consistent.

It can be shown that for every history the two notions coincide.

Theorem 1 *Let $\mathcal{D}$ be an action theory. Then for every history h:*

$$h \in Exec_{\mathcal{D}} \ \textit{ iff } \ h \in Exec_{\mathcal{M}} \textit{ for some model } \mathcal{M} \textit{ of } \mathcal{D}$$

Proof (sketch). Notice that in $Exec_{\mathcal{D}}$, for every prefix h' of h we require only consistency of $\mathcal{D} \cup Sensed(h')$, that is the existence of a model $\mathcal{M}'_{h'}$ such that $\mathcal{M}_{h'} \models \mathcal{D} \cup Sensed(h')$. So for the if direction it is sufficient to observe that $\mathcal{M}$ can serve as the model $\mathcal{M}_{h'}$ for all prefixes h' of h. For the only-if direction, while it is true that for all prefixes h' the model $\mathcal{M}_{h'}$ such that $\mathcal{M}' \models \mathcal{D} \cup Sensed(h')$ changes, we must have a model $\mathcal{M}_h$ for the history h itself such that $\mathcal{M}_h \models \mathcal{D} \cup Sensed(h)$, hence $h \in Exec_{\mathcal{M}_h}$. □

Now we extend this to "infinite histories" or *runs*. A run ϱ is an infinite set of histories (each of which is finite) defined inductively as follows:

- $\epsilon \in \varrho$;
- if $h \in \varrho$, then there *exists exactly one* ground action and one sensing result v such that $h \cdot (a, v) \in \varrho$.

We use the usual notation: $(a_0, v_1) \cdot (a_1, v_1) \cdot \cdots$ to denote runs. We say that a run ϱ is executable wrt a model $\mathcal{M}$, written $\varrho \in Exec_{\mathcal{M}}$, iff for all $h \in \varrho$ we have $h \in Exec_{\mathcal{M}}$. Similarly, we say that a run ϱ is executable wrt an action theory $\mathcal{D}$, written $\varrho \in Exec_{\mathcal{D}}$, iff for all $h \in \varrho$ we have $h \in Exec_{\mathcal{D}}$.

The crucial question is if the above theorem applies to runs as well. The answer is that in general this is *not* the case [17], as shown by the well-known tree chopping example below.

Example 1 *Consider an agent that wants to cut down a tree. Assume that the agent has an action chop to chop at the tree, and also assume that it can always find out whether the tree is down by doing the (binary) sensing action look. If the sensing result is* 1*, then the tree is down; otherwise the tree remains up. There is also a fluent $RemainingChops(n, s)$, where we assume that n ranges over the natural numbers $\mathbb{N}$ and whose value is unknown to the agent, and which is meant to represent how many chop actions are still required in s to bring the tree down. The action theory $\mathcal{D}$ is characterized by the following initial situation description, precondition, sensing and successor state axioms: the union of:*

$$\exists n. RemainingChops(n, S_0)$$

$$\begin{aligned} &RemainingChops(n, do(a, s)) \equiv \\ &\quad (a = chop \wedge n \neq 0 \wedge RemainingChops(n+1, s)) \vee \\ &\quad (a = chop \wedge n = 0 \wedge \exists m. m \leq 1 \wedge RemainingChops(m, s)) \vee \\ &\quad (a \neq chop \wedge RemainingChops(n, s)); \end{aligned}$$

$$JustChopped(do(a, s)) \equiv a = chop;$$

$$Poss(chop, s) \equiv True; \qquad Poss(look, s) \equiv True;$$

$$\begin{aligned} &SF(chop, s) \equiv True; \\ &SF(look, s) \equiv (RemainingChops(0, s)). \end{aligned}$$

Notice that the sentence $\exists n. RemainingChop(n, S_0)$, says that there exists some $n \in \mathbb{N}$, though unknown and unbounded, such that the tree will fall after n chops. However, the theory does not *entail the sentence $RemainingChop(k, S_0)$ for any constant $k \in \mathbb{N}$.*

For this reason for every model M and every run $\varrho \in Exec_{\mathcal{M}}$ such that chop is repeated infinitely often, we have that along the run sooner or later the tree will be down (i.e., $RemainingChops(0, s)$ holds for some s along the run). Using LTL introduced later (where we use situation suppressed sentences) we have that :

$$(\Box \Diamond holds(JustChopped)) \supset \Diamond \Box holds(RemainingChops(0))$$

On the other hand, if we consider runs $\varrho \in Exec_{\mathcal{D}}$, this property does not hold. This is because at each point in the run we can find a possibly different *model in which the tree is not down yet.*

Interestingly, the above example shows that expressing the remaining chops using natural numbers plays a crucial role, and a formal characterization requires second-order logic. We prove now an important result stating that if the initial situation description $\mathcal{D}_0$ is expressed in FOL then theorem 1 holds also on (infinite) runs. The result comes as a consequence of the *compactness of FOL*, which guarantees that: *if all finite subsets of an infinite set of FOL formulas are consistent then the entire set is consistent* (see, e.g.,[8]).

Theorem 2 *Let $\mathcal{D}$ be an action theory with initial situation description $\mathcal{D}_0$ expressed in FOL. Then for every run ϱ:*

$$\varrho \in Exec_{\mathcal{D}} \ \textit{ iff } \ \varrho \in Exec_{\mathcal{M}} \textit{ for some model } \mathcal{M} \textit{ of } \mathcal{D}$$

Proof (sketch). For the if direction, as before we can use the $\mathcal{M}$ to witness of satisfiability of $\mathcal{D} \cup \{Sensed(h)\}$ for all histories

h in the run, by observing that $h \in Exec_{\mathcal{M}}$ implies that $\mathcal{M} \models \mathcal{D} \cup \{Sensed(h)\}$. Hence we have that for all $h \in \varrho$, $h \in Exec_{\mathcal{D}}$. For the only if direction we need to exploit compactness of FOL. Specifically we observe that every $Sensed(h)$ can be regressed into a FOL formula $\mathcal{R}[Sensed(h)]$ over the initial situation (observe that $Sensed(h)$ is conjunction of formulas of the form $SF(a, end[h'])$ or $\neg SF(a, end[h'])$ for some prefix h' of h). We also denote by $\mathcal{D}_{uo}(h)$ the axioms in $\mathcal{D}_{uo}$ that enforce the unique name assumption for all constants mentioned in $D_0 \cup \{R[Sensed(h)]\}$, notice that such set of axioms is finite. Checking consistency of $\mathcal{D} \cup \{Sensed(h)\}$ is equivalent to checking consistency of the FOL theory $\mathcal{D}_h \cup \{\mathcal{R}[Sensed(h)]\}$ where $\mathcal{D}_h = \mathcal{D}_0 \cup \mathcal{D}_{uo}(h) \cup \mathcal{D}_{ca}$. Now if $\varrho \in Exec_{\mathcal{D}}$ then for all $h \in \varrho$, $\mathcal{D}_h \cup \{\mathcal{R}[Sensed(h)]\}$ is consistent, but then, by compactness, the entire run ϱ is "consistent", or more precisely the set $\{\mathcal{D}_h \cup \{\mathcal{R}[sensed(h)]\} \mid h \in \varrho\}$ is consistent. Hence, there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{D}_h \cup \{\mathcal{R}[sensed(h)]\}$ for every $h \in \varrho$, which means that $\mathcal{M} \models \mathcal{D} \cup \{Sensed(h)\}$ for all $h \in \varrho$, and hence $\varrho \in Exec_{\mathcal{M}}$. □

4 Progression over Histories

The progression of an action theory is the problem of *updating* the initial description of the world in $\mathcal{D}_0$ so that it reflects the current state of the world after some actions have been performed. When there is no sensing, a one-step progression of $\mathcal{D}$ wrt a *physical* ground action a is obtained by replacing the initial knowledge base $\mathcal{D}_0$ in $\mathcal{D}$ by a suitable set $\mathcal{D}_a$ of sentences so that the original theory $\mathcal{D}$ and the theory $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_a$ are equivalent wrt how they describe the situation $do(a, S_0)$ and the situations in the future of $do(a, S_0)$.

The seminal paper [12] gives a model-theoretic definition for the progression $\mathcal{D}_a$ of $\mathcal{D}_0$ wrt a physical action a and $\mathcal{D}$, which we will slightly *extend to account for sensing*. First we review the $M \sim_{S_a} M'$ relation. Let S_a be the situation term $do(a, S_0)$ and M and M' be structures with the same domains for sorts action and object. We write $M \sim_{S_a} M'$ if: *(i)* M and M' have the same interpretation of all situation-independent predicate and function symbols; and *(ii)* M and M' agree on all fluents at S_a, that is, for every fluent F, and every variable assignment μ, $M, \mu \models F(\vec{x}, S_a)$ iff $M', \mu \models F(\vec{x}, S_a)$.

Now let a be a ground action term of the form $A(\vec{c})$ and v a sensing result. Then, for $\mathcal{D}_a$ a set of (possibly second-order) sentences uniform in S_a, we say that $\mathcal{D}_a$ is a *progression* of $\mathcal{D}_0$ wrt (a, v) if for any structure M, M is a model of $\mathcal{D}_a$ iff there is a model M' of $\mathcal{D} \cup \{\varphi\}$ such that $M \sim_{S_a} M'$, where φ is the positive or negative literal of atom $SF(a, do(a, S_0))$ depending on the value v. This definition essentially requires for the two theories $\mathcal{D}$ and $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_a$ that any model of one is indistinguishable from some model of the other wrt how they interpret the (atomic) *history* (a, v) and *future histories with this prefix*. The only difference with [12] is the use of the literal φ of the sensing-result atom SF. Observe that in the case that there are no sensing actions then the two definitions coincide.

Lin and Reiter [12] showed that in general there are cases where a FO progression does not always exist. Nonetheless, recent work in [6] shows that when $\mathcal{D}$ is bounded this can be guaranteed for the case without sensing. Here we show that this is also true when we consider sensing actions. Observe that this is not trivial as the progressed knowledge base needs to incorporate in a FO representation also the effect of sensing expressed by means of an arbitrary uniform FO formula in the right-hand side of the corresponding sensed fluent axiom in $\mathcal{D}_{sf}$. On the other hand progression with respect to a sensing action does not affect the extension of fluents in each model in any way; it may only "remove" or "filter out" some of the models of $\mathcal{D}$ so that only those that comply with the sensing result remain.

We make use of the properties of bounded theories and results in [5, 6] which allows characterizing the models of the initial situation description of any bounded action theory by means of a *finite* number of *characteristic sentences* each of which qualifies as a *relatively complete initial knowledge base with bounded unknowns* [25].

Theorem 3 *All bounded action theories with sensing are iteratively first-order progressable over histories.*

Proof. Let us consider an action theory $\mathcal{D}$ with sensing and the case of the atomic action history (a, v). If a is a physical action, then a FO progression $\mathcal{D}_a$ (that is again bounded) can be computed as the disjunction of the progression of each characteristic sentence ϕ_i separately as a relatively complete KB [6]. For the case that a is a sensing action of the form $A(\vec{c})$, note that SSAs do not affect the extension of fluents and consider $\mathcal{D}_a$ as the disjunction $\bigvee_i \phi_i(S_0/S_a) \wedge \varphi$,[6] where ϕ_i are the characteristic sentences as before and φ is the right-hand side of the corresponding ground sensing axiom $SF(A(\vec{c}), S_a) \equiv \phi_A(\vec{c}, S_a)$ or its negation depending on the sensing result v. We can show that $\mathcal{D}_a$ qualifies as a progression of $\mathcal{D}_0$ wrt (a, v). (⇒): Let M be a model such that $M \models \mathcal{D}_a$. Then $M \models \phi_i(S_0/S_a) \wedge \varphi$ for some i. Construct M' identical to M except that the extension of fluents in S_0 is copied from their extension in S_a, and for the rest of the situations the extensions are specified by the rest of the axioms in $\mathcal{D}$. Then $M' \models \mathcal{D}_0$ since ϕ_i is a characteristic sentence for $\mathcal{D}_0$, $M \sim_{S_a} M'$ and $M' \models \mathcal{D}$ by construction of M', and also $M' \models \varphi$ since $M \models \varphi$ and $M \sim_{S_a} M'$. (⇐): Assume that there exists a model M' of $\mathcal{D} \cup \{\varphi\}$ such that $M \sim_{S_a} M'$ with M otherwise arbitrary. Then $M' \models \phi_i$ for some i and since a is a sensing action also $M' \models \phi_i(S_0/S_a)$. By the hypothesis it follows then that $M' \models \phi_i(S_0/S_a) \wedge \varphi$. Since $M \sim_{S_a} M'$ it follows that $M \models \phi_i(S_0/S_a) \wedge \varphi$ and by construction of $\mathcal{D}_a$ it follows that $M \models \mathcal{D}_a$. Finally, note that as the theory is bounded over all situations, $\mathcal{D}_a$ can always be rewritten as a disjunction of characteristic sentences (for both physical and sensing actions), thus we can progress iteratively to deal with arbitrary histories. □

5 Verification of Online Executions with Sensing

Linear Time Logic. Dynamic properties over online executions can be expressed using a First-Order variant of Linear Time Logic (FO LTL) [15], whose syntax is as follows:

$$\varphi ::= holds(\phi) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

where ϕ is an arbitrary closed uniform *situation-suppressed* (i.e., with all situation arguments in fluents suppressed) situation calculus FO formula, whose constants must appear in $\mathcal{D}$. The logic is closed under the boolean connectives and includes the usual unary temporal operator $\bigcirc$ (*next-time*) and the binary temporal operator $\mathcal{U}$ (*until*). Intuitively, $\bigcirc\varphi$ says that φ holds at the *next* instant, $\varphi_1 \,\mathcal{U}\, \varphi_2$ says that at some future instant φ_2 will hold and *until* that point φ_1 holds. Also, common abbreviations are introduced: (i) standard boolean abbreviations, $\vee$ (or) and $\supset$ (implies); (ii) $\Diamond\varphi$ which stands for $\mathtt{true}\,\mathcal{U}\,\varphi$, and says that φ will *eventually* hold; (iii) $\Box\varphi$, which stands for $\neg\Diamond\neg\varphi$, and says that from the current instant on φ will *always* hold.

The semantics of FO LTL is given in terms of infinite *traces* in the standard way [15]. A trace is an infinite sequence $\pi = \Pi_0 \cdot \Pi_1 \cdot \cdots$, where, in our case, each Π_i is a possibly infinite set of formulas of the form $holds(\phi)$. Given a trace π, we inductively define when a FO LTL formula φ *is true* at position i ($i \in \mathbb{N}$) in π, in symbols $\pi, i \models \varphi$:

[6] $\phi(\sigma/\sigma')$ denotes the result of replacing every occurrence of σ in ϕ by σ'.

- $\pi, i \models holds(\phi)$ iff $holds(\phi) \in \Pi_i$.
- $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$.
- $\pi, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$.
- $\pi, i \models \bigcirc\varphi$ iff $\pi, i+1 \models \varphi$.
- $\pi, i \models \varphi_1 \mathcal{U} \varphi_2$ iff for some j such that $i \leq j$, we have that $\pi, j \models \varphi_2$ and for all $i \leq k < j$, we have that $\pi, k \models \varphi_1$.

We say that a trace π *satisfies* a FO LTL formula φ, written $\pi \models \varphi$, iff $\pi, 0 \models \varphi$. We denote by $\mathcal{L}(\varphi)$ the set of traces π such that $\pi \models \varphi$. The language $\mathcal{L}(\varphi)$ can be recognized by a *finite* Büchi automaton. These are finite state automata that accept infinite runs, by requiring that some accepting state is visited infinitely often [3]. In our case, additional care is needed in handling atomic formulas $holds(\phi)$, which require the use of FOL reasoning.

Given a history h, we denote by $thms[h]$ the set $\{holds(\phi) \mid \mathcal{D} \cup Sensed(h) \models \phi[end[h]]\}$. Then, given a run $\varrho = (a_0, v_0) \cdot (a_1, v_1) \cdot \cdots$ we define the corresponding trace $\pi_\varrho = thms[\epsilon] \cdot thms[(a_0, v_0)] \cdot thms[(a_0, v_0) \cdot (a_1, v_1)] \cdot \cdots$.

In our setting, the agent reasons about the possible online executions, to verify whether FO LTL properties of interest hold. To do so, we rely on *model checking*, which, in our context, amounts to checking that all online executions of an agent following an action theory $\mathcal{D}$ satisfy a FO LTL formula φ, written $\mathcal{D} \models \varphi$.[7] Formally, $\mathcal{D} \models \varphi$ iff for all runs ϱ such that $\varrho \in Exec_{\mathcal{M}}$, for some $\mathcal{M}$, we have $\pi_\varrho \models \varphi$. Notice that, by Theorem 2, if the initial situation description of the action theory is expressed in FOL, then the above definition can be simplified into: $\mathcal{D} \models \varphi$ iff for all runs $\varrho \in Exec_{\mathcal{D}}$ we have $\pi_\varrho \models \varphi$.

We can define a transition system (TS) that contains exactly all the traces corresponding to the online executable runs of an action theory $\mathcal{D}$. Formally, such a TS is defined as $T_{\mathcal{D}} = \langle Q, q_0, \lambda, \rightarrow\rangle$, where:

- $Q = Exec_{\mathcal{D}}$ is the set of *states*;
- $q_0 = \epsilon$ is the *initial state*;
- λ is the *labeling function*, mapping each state q to the set of uniform situation-suppressed closed formulas $\lambda(q) = thms(q)$;
- $\rightarrow \subseteq Q \times Q$ is the *transition relation*, such that $q \rightarrow q'$ iff $q' = q \cdot (a, v)$ for some a and v.

It is easy to see that $T_{\mathcal{D}}$ generates all the traces corresponding to the runs in $Exec_{\mathcal{D}}$.[8] We denote the set of traces of $T_{\mathcal{D}}$ by $\mathcal{L}(T_{\mathcal{D}})$. We now observe that to check whether $\mathcal{D} \models \varphi$ we can check whether $\mathcal{L}(T_{\mathcal{D}}) \cap \mathcal{L}(\neg\varphi) = \emptyset$. The problem however is that $T_{\mathcal{D}}$ is infinite so the usual model checking techniques based finite Büchi automata [3] do not work. This is obviously true also for bounded theories. However, under the boundedness assumption, the construction of a finite faithful abstraction of $T_{\mathcal{D}}$ becomes possible.

Theorem 4 *Let $\mathcal{D}$ be an action theory bounded by b and φ a FO LTL formula. Then, checking whether $\mathcal{D} \models \varphi$ is decidable.*

Proof (sketch). We show this result following the lines of [6].

The first step is to actually use progression for labeling states of transition systems. Using progression we can re-define the labeling function of $T_{\mathcal{D}}$ as: $\lambda(q) = \{holds(\phi) \mid Prog(q) \models \phi[S_0]\}$. By the correctness of progression we have that this new specification defines the same labeling function as the original one.

Now we observe that the actual value of constants that occur in the formulas of $\lambda(q)$ and are not mentioned in $\mathcal{D}$ is not relevant when checking the FO LTL formula φ (as it can only mention constants in $\mathcal{D}$ by definition). As a result these constants can be renamed arbitrarily, without affecting the result. Formally, two labelings Π and Π' are *logically equivalent modulo renaming*, written $\Pi \sim \Pi'$, if there exists a bijection $g : \mathcal{N} \rightarrow \mathcal{N}$ s.t. $\Pi \models g(\Pi')$ and $\Pi' \models g^{-}(\Pi)$ (for g^{-} the inverse of g), where: $g(\Pi')$ stands for the set of formulas obtained from Π' by replacing each constant n in Π' not occurring in $\mathcal{D}$ by $g(n)$; and similarly $g^{-}(\Pi)$ is the theory obtained by replacing each constant of n in Π not occurring in $\mathcal{D}$, by $g^{-}(n)$.

Next we define bisimulation between TS as usual but using as local condition the above equivalence modulo renaming. A *bisimulation* between T_1 and T_2 is a relation $B \subseteq Q_1 \times Q_2$ s.t. $B(q_1, q_2)$ implies:

- $\lambda_1(q_1) \sim \lambda_2(q_2)$;
- for every transition $q_1 \rightarrow_1 q_1'$, there exists a transition $q_2 \rightarrow_2 q_2'$, s.t. $\langle q_1', q_2'\rangle \in B$;
- for every transition $q_2 \rightarrow_2 q_2'$, there exists a transition $q_1 \rightarrow_1 q_1'$, s.t. $\langle q_1', q_2'\rangle \in B$.

T_1 and T_2 are said to be *bisimilar* written $T_1 \approx T_2$, if $\langle q_{10}, q_{20}\rangle \in B$, for some bisimulation B. As usual bisimilarity is an equivalence relation, which implies trace equivalence [3]. Hence if $T_1 \approx T_2$ then T_1 and T_2 generate the same traces modulo renaming.

Next we construct a finite transition system T_F, using Algorithm 1, which takes an action theory $\mathcal{D}$ bounded by b as input, and returns a finite-state TS $T_F = \langle Q, q_0, \partial, \rightarrow\rangle$ whose labels in the state denote finite FOL theories, such that when we close them deductively we get a TS that is bisimilar to $T_{\mathcal{D}}$.

Algorithm 1 Computation of a finite-state TS.

```
$Q := \{q_0\}$; $\partial(q_0) := \mathcal{D}_0$ after suppressing situations;
let $\rightarrow$ be the empty relation;
let $C$ be the set of constants occurring in $\mathcal{D}$;
repeat
  let $q \in Q$ and $C_{\partial(q)}$ be the set of constants occurring in $\partial(q)$;
  for all action types $A$ with parameters $\vec{x}$ do
    let $\vec{o} \subset \mathcal{N}$ be any (finite) set s.t. $|\vec{o}| = |\vec{x}|$ and
        $\vec{o} \cap (C \cup C_{\partial(q)}) = \emptyset$;
    for all parameter substitutions $\theta : \vec{x} \rightarrow C_{\partial(q)} \cup C \cup \vec{o}$ do
      let $\mathcal{D}_{q,(a,v)}$ be the situation suppressed progression of
      $\partial(q)$ wrt $a = A(\vec{x}\theta)$ and sensing result $v$, such that
      $\partial(q)[S_0] \models Poss(a, S_0)$ and $(\mathcal{D} - \mathcal{D}_o) \cup \partial(q)[S_0] \cup$
      $Sensed[(a, v)]$ is consistent;
      if there exists $q' \in Q$ s.t. $\mathcal{D}_{q,(a,v)} \sim \partial(q')$ then
        $\rightarrow := \rightarrow \cup \{q \rightarrow q'\}$;
      else
        let $Q := Q \uplus \{q'\}$, for $q'$ a fresh state, with $\partial(q') =$
        $\mathcal{D}_{q,(a,v)}$, and $\rightarrow := \rightarrow \cup \{q \rightarrow q'\}$;
      end if
    end for
  end for
until (transition relation $\rightarrow$ does not change any more)
```

This procedure generates T_F by iteratively progressing $\mathcal{D}$, starting from the (situation-suppressed) initial situation description. In doing so, not all the infinitely many executable actions are considered for progression at each step, but only a finite subset. These can be chosen so as to obtain one representative for each equivalence class, wrt logical equivalence modulo renaming, of progressions that can be obtained after applying all possible actions. This is actually achieved by including in $\vec{o}$ a distinct value for each parameter, distinct also from all the elements of C and $C_{\partial(q)}$. Then, to guarantee coverage of all equivalence classes, all action types and all assignments of parameters to $\vec{o} \cup C \cup C_{\partial(q)}$ are considered. Notice that by the boundedness assumption and Th. 3, the progression of $\mathcal{D}_{q,(a,v)}$ is computable, an obvious necessary condition for the algorithm to

[7] To deal with temporal logics, which assume an infinite future, we assume, wlog, that histories can be extended by at least one action. If needed, we add a *no-op* dummy action without effects that can always be performed.

[8] A transition system generates, through λ and $\rightarrow$, all traces corresponding to its infinite paths.

terminate. Similarly, testing the condition of the **if** statement is decidable, as $\mathcal{D}$ is b-bounded and so are all of the theories labeling the states of Q. Termination of the algorithm follows by the fact that $\mathcal{D}$ is bounded by b, thus only finitely many equivalence classes of theories, wrt logical equivalence modulo renaming, exist, which constitute Q.

Now, let us consider the TS T'_F obtained from T_F by substituting the labeling function ∂ with λ such that $\lambda(q) = \{holds(\phi) \mid \partial(q) \vDash \phi\}$. Then we have that $T'_F \approx T_{\mathcal{D}}$. This can be proved by co-induction, showing that $q \sim q'$, with q from $T_{\mathcal{D}}$ and q' from T'_F is indeed a bisimulation, which includes the initial states of the two TSs. Finally, since bisimilar TSs generate equivalent traces, the result follows. $\Box$

Notice that Algorithm 1 in the proof provides a practical way to generate the TS to model check.

Example 2 *Consider an agent in a Candy-Crush-like domain, where candies of different flavours arrive in a continuous flow. The agent has a bag of bounded capacity, where candies can be stored. The actions available to the agent are: (i) $grab$ a new candy from the flow; (ii) $pick$ a stored candy from the bag; (iii) $store$ the candy in its hand (grabbed or picked) into the bag (if space is available in the bag); (iv) $taste$ a candy to sense its flavour; (v) $discard$ the candy in its hand; (vi) eat a candy, if its flavour is cherry. We assume we have fluents $Grabbed$, $Eaten$ and $Discarded$ to model that the respective actions have just been executed. The following formula expresses that if the agent grabs new candies forever, then it must eat or discard candies forever (as it cannot accumulate candies forever):*

$$\Box\Diamond holds(\exists x.Grabbed(x))) \supset \Box\Diamond holds(\exists x.Eaten(x) \vee Discarded(x)).$$

We close the section by observing that we do not allow for quantification "across situations", i.e., for including temporal operators within the scope of FO quantifiers. This feature can be shown to lead to undecidability even in very simple cases, as verification in this setting can be reduced to model checking of FO LTL with *freezing quantifiers* [2]. As noted in [5], however, in a rich formalism like the situation calculus, even when restricted to bounded action theories, the impact of such limitations can be mitigated by recording in the current situation information from past situations. For instance, one can introduce a finite number of "registers", i.e., fluents storing at most one tuple, and use them to refer to tuples across situations. Then we can write, e.g. (assuming for simplicity that the mentioned fluents have all the same arity), the formula:

$$\Box(holds(\exists\vec{x}.Reg_i(\vec{x}) \wedge F(\vec{x})) \supset \Diamond holds(\exists\vec{y}.Reg_i(\vec{y}) \wedge F'(\vec{y}))),$$

which says that whenever the tuple referred to by register i has property F, then eventually it comes to have property F'.

6 Conclusion

In this paper, we have proposed a first-person, computationally grounded account of agents reasoning about their online executions with sensing, by checking sophisticated FO linear time logic properties (without quantification across situations) over situation calculus action theories. For bounded theories, we have shown that progression over histories that include sensing results is always first-order, and that verification of FO LTL properties is decidable. One key result has been showing that while reasoning, we can ignore the distinction between getting possible sensing values from a possible model (for all models) and getting simply all consistent sensing values, in the case of a first-order initial situation description.

It is possible to relate our first-person account to a third-person account (a modeler's perspective) involving a knowledge operator in the logic [20, 4]. This is especially interesting when there are several agents working from their own first-person account simultaneously, and their relationship to a third-person (modeler) account [22]. We will investigate this in future work. Finally, we would like to extend the proposed first-person account with partial observability of actions as in [1], while remaning computationally grounded. This would allow us to model a wider range of settings as bounded theories.

Acknowledgements This research has been partially supported by the EU project Optique (FP7-IP-318338), the Sapienza Award 2013 "SPIRITLETS: SPIRITLET-based Smart Spaces", and NSERC.

REFERENCES

[1] F. Bacchus, J. Y. Halpern, and H. J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artif. Intell.*, 111(1-2):171–208, 1999.
[2] 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'13*.
[3] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
[4] G. De Giacomo, Y. Lespérance, and F. Patrizi. Bounded Epistemic Situation Calculus Theories. In *Proc. of IJCAI'13*.
[5] G. De Giacomo, Y. Lespérance, and F. Patrizi. Bounded Situation Calculus Action Theories and Decidable Verification. In *Proc. of KR'12*.
[6] G. De Giacomo, Y. Lespérance, F. Patrizi, and S. Vassos. Progression and Verification of Situation Calculus Agents with Bounded Beliefs. In *Proc. of AAMAS'14*. To appear.
[7] G. De Giacomo and H. J. Levesque. An incremental interpreter for high-level programs with sensing. In *Logical Foundations for Cognitive Agents*, pages 86–102. 1999.
[8] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.
[9] C. Fritz and S. McIlraith. Decision-theoretic Golog with qualitative preferences. In *KR*, pages 153–163, 2006.
[10] H. J. Levesque. What is planning in the presence of sensing? In *AAAI*, pages 1139–1146, 1996.
[11] H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. MIT Press, 2001.
[12] F. Lin and R. Reiter. How to Progress a Database. *Artificial Intelligence*, 92(1-2):131–167, 1997.
[13] J. McCarthy and P. J. Hayes. Some Philosophical Problems From the StandPoint of Artificial Intelligence. *Machine Intell.*, 4:463–502, 1969.
[14] F. Pirri and R. Reiter. Some Contributions to the Metatheory of the Situation Calculus. *J. ACM*, 46(3):261–325, 1999.
[15] A. Pnueli. The temporal logic of programs. In *FOCS*, 1977.
[16] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
[17] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On Ability to Autonomously Execute Agent Programs with Sensing. In *Proc. of AAMAS'04*.
[18] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the Limits of Planning over Belief States under Strict Uncertainty. In *Proc. of KR'06*.
[19] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the semantics of deliberation in indigolog - from theory to implementation. *Ann. Math. Artif. Intell.*, 41(2-4):259–299, 2004.
[20] R. B. Scherl and H. J. Levesque. The frame problem and knowledge-producing actions. In *AAAI*, pages 689–695, 1993.
[21] R. B. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artif. Intell.*, 144(1-2):1–39, 2003.
[22] S. Shapiro, Y. Lespérance, and H. J. Levesque. The cognitive agents specification language and verification environment for multiagent systems. In *AAMAS*, pages 19–26, 2002.
[23] E. Ternovska. Automata theory for reasoning about actions. In *IJCAI*, pages 153–159, 1999.
[24] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, pages 238–266, 1995.
[25] S. Vassos and F. Patrizi. A Classification of First-Order Progressable Action Theories in Situation Calculus. In *Proc. of IJCAI'13*.
[26] M. Wooldridge. Computationally Grounded Theories of Agency. In *Proc. of ICMAS*, pages 13–22, 2000.