Progression and Verification of Situation Calculus Agents with Bounded Beliefs

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

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

Fabio Patrizi & Stavros Vassos
DIAG, Sapienza
Università di Roma
Roma, Italy
lastname@dis.uniroma1.it

ABSTRACT

In this paper we investigate agents that have incomplete information and make decisions based on their beliefs, expressed as situation calculus bounded action theories. Such theories have an infinite object domain, but the number of objects that belong to fluents at each time point is bounded by a given constant. Recently it has been shown that verifying temporal properties over such theories is decidable. Here, we first show that we can actually check whether an arbitrary action theory maintains boundedness. Secondly, we examine progression. Progression can be thought of as capturing the notion of belief states resulting from actions in the situation calculus. In the general case, such belief states can be expressed only in second-order logic. Here, we show that for bounded action theories, progression, and hence belief states, can always be represented in first-order logic. Based on this result, we further prove decidability of temporal verification over online executions, i.e., those executions resulting from agents performing only actions that are feasible according to their beliefs.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Theory, Verification, Language.

Keywords

Situation Calculus, Progression, Bounded Action Theories, Online Execution, Verification of Agents.

1. INTRODUCTION

In this paper, we develop a computationally grounded framework to model and verify agents that operate in infinite domains, have incomplete information and make decisions based on their beliefs, expressed as situation calculus bounded action theories.

The situation calculus [11, 13] is a widely used and expressive first-order logical framework for reasoning about action in which many issues have been addressed, e.g., the frame problem, time, continuous change, complex actions and processes, uncertainty, etc. It is also the basis of the Golog family of agent programming languages [8, 2] and has been used to develop rich theories of agent mental states and action [17].

Here, we use situation calculus action theories to express the mental model of an agent that can deliberate and act in the world. That is we take a first-person view of the logical theory where the theory captures what the agent believes about the domain and how actions affect it. Specifically, we adopt *bounded action theories*[4], for which it was shown that verification of a very expressive class of first-order μ-calculus temporal properties is *decidable*. Bounded action theories are basic action theories [13], where it is entailed that in all situations the number of fluent atoms that are true is bounded by a constant. In such theories, the object domain remains nonetheless infinite, as is the domain of situations. Boundedness can often be safely assumed, since in real domains facts rarely persist indefinitely as everything decays and changes. Moreover, agents often forget facts either because they are not used or because they cannot be reconfirmed. [4] gives many examples of domains modeled as bounded action theories. It also identifies various ways to obtain bounded action theories: 1. by strengthening preconditions to block actions where the bound would be exceeded, 2. by ensuring that actions are *effect bounded* and never make more fluent atoms true than they make false, and 3. by using *fading fluents* whose strength fades over time unless they are reconfirmed.

This work presents three key results for the use of bounded action theories as mental models for agents. First, we show that using the techniques in [4] we *can actually check whether an arbitrary action theory maintains boundedness*. Given that we can usually specify the initial situation description so that it is provably bounded, this allows us to verify that the entire action theory is bounded.

Secondly, we examine *progression*. By progressing the initial situation description over a sequence of action, we obtain a new situation description representing all that is known about the situation after the action sequence. Progression can be thought of as capturing the notion of belief states[1] that result from actions in the situation calculus. This allows us to obtain a computationally grounded model of agents [20], in the sense that such a model captures how the belief states of agents are generated and updated from the action theory, which describes truth and how truth evolves as actions are performed. In the general case, progression, and hence such belief states, can only be expressed in second-order logic [9, 18]. However here, we show that *for bounded action theories, progression, and belief states, can always be represented in first-order logic*. We discuss how a first-order progression can be constructed.

Note that often belief states are a priori thought of as some sort of *first-order theory* whose models are the alternative possible worlds that the agent may be in. However in the situation calculus, first-

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*
Copyright © 2014, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

[1]Here, we assume that the agent's beliefs are always true and use belief and knowledge interchangeably.

order belief states cannot be complete in general (second-order logic is needed [9, 18]) and must be complemented with the full action theory (including a description of the past situations) in order to fully capture what is entailed by the information available to the agent [15, 14]. But when progression is first-order representable, then such first-order belief states are indeed "complete" and no further information (apart from the specification of actions) is needed.

Finally, we move to verification of agents [6, 10], and exploit the above result on progression to show the decidability of verifying temporal properties over the online executions of agents. These are execution paths where the agent only performs actions that can surely be executed according to the agent beliefs. We show that a very rich class of temporal properties expressed in a first-order variant of the μ-calculus can be verified. Notably, in this language, one can easily express that there exists a sequence of actions that reaches a state where a goal is known to be true, even if the agent has incomplete information about the world (as represented by the action theory), and hence we can solve conformant planning [1] in this rich setting, through verification.

2. SITUATION CALCULUS

The *situation calculus* [11, 13] 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 (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 [12, 13]. 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 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); (first-order, or 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 (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.

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 [12, 13]. 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 [13] 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.

3. PROGRESSION AND BELIEF STATES

The progression of a basic 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. In other words, a one-step progression of $\mathcal{D}$ w.r.t. a ground action α is obtained by replacing the initial knowledge base $\mathcal{D}_0$ in $\mathcal{D}$ by a suitable set $\mathcal{D}_\alpha$ of sentences so that the original theory $\mathcal{D}$ and the theory $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_\alpha$ are equivalent w.r.t. how they describe the situation $do(\alpha, S_0)$ and the situations in the future of $do(\alpha, S_0)$.

The seminal paper [9] gives a model-theoretic definition for the progression $\mathcal{D}_\alpha$ of $\mathcal{D}_0$ w.r.t. α and $\mathcal{D}$, which we briefly review. Denote by S_α the situation term $do(\alpha, S_0)$ and let M and M' be structures with the same domains for sorts action and object. We write $M \sim_{S_\alpha} M'$ 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_α, that is, for every relational fluent F, and every variable assignment μ, $M, \mu \models F(\vec{x}, S_\alpha)$ iff $M', \mu \models F(\vec{x}, S_\alpha)$. Then, for $\mathcal{D}_\alpha$ a set of (possibly second-order) sentences uniform in S_α, we say that $\mathcal{D}_\alpha$ is a *progression* of $\mathcal{D}_0$ w.r.t. α if for any structure M, M is a model of $\mathcal{D}_\alpha$ iff there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$. The definition in [9] essentially requires for the two theories $\mathcal{D}$ and $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_\alpha$ that any model of one is indistinguishable from some model of the other w.r.t. how they interpret the situations in S_α and the future of S_α.

Observe that we can take progression as a way to characterize the belief state of an agent in a particular situation, i.e., what the agent believes about the current situation and what may happen in the future. In the context of the situation calculus, the notion of belief states may be captured by *everything that is entailed* in a particular situation.[2] The *progressed* knowledge base is a sentence that essentially represents this. It has been shown that in general, progression, hence this form of belief states, can only be captured in second-order logic [9, 18]. Nonetheless, we will show that for bounded action theories a first-order progression can always be constructed.

4. 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:

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

We can also define $(|\{\vec{x} \mid \phi(\vec{x})\}| < b) \doteq \neg(|\{\vec{x} \mid \phi(\vec{x})\}| \geq b)$. Using this, [4] defines the notion of a fluent $F(\vec{x}, s)$ in situation

[2] In fact, also accounts in epistemic variants of the situation calculus has been studied [16, 7], but here we appeal to an interpretation directly based on entailment.

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 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 it entails that: $\forall s.Executable(s) \supset Bounded_b(s)$. [4] shows that for bounded action theories, verification of sophisticated temporal properties is decidable. It also identifies interesting classes of such theories.

Example 1. Consider a factory where items are moved by robots between available working stations, may be painted when located at a particular station, and shipped out of the factory when placed at the shipping dock. Items may be heavy or fragile, in which case a different type of robot is required for moving them. We also consider an item for which there is incomplete information about its properties as heavy or fragile, in which case a *conformant* plan needs to be obtained for processing it. We introduce the following action theory where fluents and actions have the intuitive meaning.[3]
Action Precondition Axioms:

- $Poss(move(r, x, l), s) \equiv IsRobot(r) \wedge IsLoc(l)$
- $Poss(ship, s) \equiv \exists x\, At(x, ShipDock, s)$
- $Poss(paint(x), s) \equiv At(x, PaintStn, s)$

Successor State Axioms:

- $At(x, l, do(a, s)) \equiv \gamma(x, l, a, s)^+ \vee \neg\gamma^-(x, l, a, s) \wedge At(x, l, s)$, where:

  $\gamma^+(x, l, a, s) \equiv \exists r.a = move(r, x, l) \wedge \exists z At(x, z, s) \wedge$
  $\neg\exists y At(y, l, s) \wedge (Heavy(x) \supset HandlesHeavy(r)) \wedge$
  $(Fragile(x) \supset HandlesFragile(r))$, and

  $\gamma^-(x, l, a, s) \equiv a = ship \wedge At(x, ShipDoc, s) \vee$
  $\exists r \exists l'.a = move(r, x, l') \wedge l \neq l' \wedge$
  $\neg\exists y At(y, l', s) \wedge (Heavy(x) \supset HandlesHeavy(r)) \wedge$
  $(Fragile(x) \supset HandlesFragile(r))$;

- $Painted(x, do(a, s)) \equiv a = paint(x) \vee$
  $Painted(x, s) \wedge \neg Shipped(x, s)$;
- $Shipped(x, do(a, s)) \equiv a = ship(x)$;

Initial State Axioms:

- $IsRobot(r) \equiv r = R1 \vee r = R2$;
- $HandlesHeavy(r) \equiv r = R1$;
- $HandlesFragile(r) \equiv r = R2$;
- $IsLoc(l) \equiv l = Hold1 \vee l = Hold2 \vee l = Hold3 \vee$
  $l = PaintStn \vee l = ShipDock$;
- $At(x, l, S_0) \equiv (x = I1 \wedge l = Hold1) \vee$
  $(x = I2 \wedge l = Hold2) \vee (x = I3 \wedge l = Hold3)$;
- $\forall x \neg Painted(x, S_0) \wedge \forall x \neg Shipped(x, S_0)$;
- $(Heavy(x) \equiv x = I1) \wedge (Fragile(y) \equiv (y = I2 \vee y = I3)) \vee$
  $(Heavy(x) \equiv (x = I1 \vee x = I3)) \wedge (Fragile(y) \equiv y = I2)$.

Note that each station may hold at most one item at any given time. Also, $Shipped(x, s)$ holds if item x has been shipped in the last performed action, while $Painted(x, s)$ keeps track of items that have been painted until $Shipped(x, s)$ becomes true.

It is not difficult to show that this theory is in fact bounded by 5. First note that there are 5 locations initially and as $IsLoc$ is a non-fluent predicate this always remains true (and similarly for the other non-fluent predicates). For the fluent At, initially it is bounded by 3 and the action theory maintains this bound since moving an item replaces one atom of At by another, shipping removes one, and painting has no effect on At. Note that here we do not model the arrival of new items (we will do this in the next example), however since there can be at most one item in each location, even in this case At would remain bounded by 5. For the fluent $Shipped$, it is bounded by 1 as initially it is an empty relation and the action theory ensures that the ship action leaves at most one atom true at each situation, namely the item that was just shipped. Finally, for the fluent $Painted$ it is bounded by 3 as no new items can arrive and only those present in the factory can be painted.

The case of item $I3$ is interesting as it illustrates how incomplete knowledge affects planning. The above action theory entails that *a plan exists such $I3$ is eventually shipped*. In this *conformant plan*, both robots will attempt to move item $I3$ to the shipping dock in sequence with exactly one of them successfully moving it (depending on whether it is fragile or heavy), and then it will be shipped.

On the other hand, for either robot r the action theory does not entail that r can successfully move $I3$ to the shipping dock: in case $I3$ is heavy, only $R1$ can move it, while in case it is fragile only $R2$ can move it. As a result, the plan of robot $R1$ moving $I3$ to the shipping dock and then shipping it is *not feasible* because the agent in control does not know that $I3$ will be at the shipping dock after $R1$ tries to move it and as a result *the ship action will not be known to be executable* (and similarly for a plan that only involves $R2$).

Finally, the theory entails that there exists a plan such that eventually all objects are painted and shipped. We discuss how such statements can be specified and verified in the remainder. □

In the above example, it is straightforward to satisfy the boundedness assumption as the domain of objects that may be affected in any future situation is in fact limited to the objects mentioned in the description of the initial state. Nonetheless, we can easily extend it to the case where arbitrary items may be introduced through an $arrive$ action that brings new objects to the factory.

Example 2. We adapt the theory of Example 1 so that it also includes action $arrive(x)$, where item x is placed in the shipping dock provided that the dock is free and the item is not already in the factory. This can be seen as an exogenous action that is invoked periodically when new items arrive and need to be processed.

The new theory is the same as before except that the following action precondition axiom is added

$$Poss(arrive(x), s) \equiv \neg\exists y At(y, ShipDock) \wedge \neg\exists l At(x, l, s),$$

and $\gamma^+(x, l, a, s)$ in the successor state axiom for $At(x, l, s)$ is replaced by the following formula:

$$\begin{aligned} a = arrive(x) \wedge l = ShipDock \\ \vee \exists r.a = move(r, x, l) \wedge \exists z At(x, z, s) \wedge \neg\exists y At(y, l, s) \\ \wedge (Heavy(x) \supset HandlesHeavy(r)) \\ \wedge (Fragile(x) \supset HandlesFragile(r)). \end{aligned}$$

First, note that as there are infinitely many constants, which are standard names, effectively an unbounded number of items may be handled by subsequent arrive, move, and ship actions. Observe though that since there are only a fixed number of stations in the factory, *at any given situation the number of items present in the factory remains bounded*, in fact by the same number as before. We can reason about this in a similar way as for the previous example.

As before, At is initially bounded by 3 but now the action theory ensures that it remains bounded by 5. This is because new items may arrive at the shipping dock only when the shipping dock is

[3] We omit leading universal quantifiers for readability. For simplicity we use non-fluent predicates, e.g., $IsLoc$; to conform with the assumptions of previous section, such predicates can be modeled by fluents whose SSA preserves their truth value in all situations.

empty. As moving an item replaces one atom of At by another, and shipping an item removes one atom, there can be at most 5 items in the factory, one in each of the 5 available stations. As a result, $Painted$ is also bounded by 5 as only those items present in the factory can be painted, and $Shipped$ is bounded by 1 as before. □

In the previous examples all individuals that may appear in the extensions of fluents are standard names mentioned in the description of the initial knowledge base or in the arguments of subsequent actions. Nonetheless, this is not necessary for boundedness. In the following example we adapt the theory to express that initially there is an item at station $Hold3$ which is either fragile or heavy, but whose identity is not known.

Example 3. Consider again Example 1 and assume that the identity of the object at station $Hold3$ is unknown. The new theory is the same as before except for the initial state axioms for At, $Heavy$, and $Fragile$, now combined into the following:

$$\begin{aligned}\exists i.\ &\neg IsLoc(i) \wedge \neg IsRobot(i) \wedge i \neq I1 \wedge i \neq I2\\ &\wedge \forall x\{At(x,l,S_0) \equiv (x = I1 \wedge l = Hold1)\\ &\quad \vee (x = I2 \wedge l = Hold2) \vee (x = i \wedge l = Hold3)\}\\ &\wedge \forall x \forall y\{(Heavy(x) \equiv x = I1)\\ &\quad \wedge (Fragile(y) \equiv (y = I2 \vee y = i))\\ &\quad \vee (Heavy(x) \equiv (x = I1 \vee x = i))\\ &\quad \wedge (Fragile(y) \equiv y = I2).\}\end{aligned}$$

This shows that the boundedness condition does not require the identity of the individuals involved in the relations to be known. This allows for representing rich scenarios where initially it is only specified that a bounded number of objects will be in the extension of some property, and where the identity of these objects may be discovered later. For example, in an university admissions scenario we may know that at most ten new doctoral students will be admitted, and later on learn who these new students are. □

5. CHECKING BOUNDEDNESS

We show that we can always check whether any BAT maintains boundedness for a given bound. That is if the initial situation description is bounded then the entire theory is too (for all situations).

We capture the notion of being bounded at the next step as:

$$\bigwedge_{A \in \mathcal{A}} \forall \vec{x}.Poss(A(\vec{x}), s) \supset Bounded_{F,b}(do(A(\vec{x}), s)).$$

Notice that each $Bounded_{F,b}(do(A(\vec{x}), s))$ is regressable through $A(\vec{x})$. As a result the formula above is equivalent to a first-order situation calculus formula uniform in s; we call the latter formula $NextOrigBounded_{F,b}(s)$, and we call $NextOrigBounded_b(s)$ the formula $\bigwedge_{F \in \mathcal{F}} NextOrigBounded_{F,b}(s)$.

To check that the theory is bounded by b it is sufficient to verify that the theory entails the temporal formula (expressed in the language of [4]):

$$AGNextOrigBounded_b \doteq \nu Z.NextOrigBounded_b \wedge [-]Z,$$

which expresses that always along any path $NextOrigBounded_b$ holds. Unfortunately deciding whether this formula is entailed by the action theory is directly doable with the techniques in [4] only if the theory is bounded, which is what we want to check. However we can construct a modified version of the action theory that is guaranteed to be bounded and that can be used to do the checking.

Let $\mathcal{D}$ be the action theory. We define a new action theory $\mathcal{DD}$ obtained by augmenting $\mathcal{D}$ as follows:

- $\mathcal{DD}_{S_0} = \mathcal{D}_{S_0} \cup \{\phi[\vec{F}/\vec{F'}] | \phi \in \mathcal{D}_{S_0}\}$
- $\mathcal{DD}_{ssa} = \mathcal{D}_{ssa} \cup \{F'(\vec{x}, do(a,s)) \equiv \Phi(\vec{x}, a, s) \wedge$ $NextOrigBounded_b(s) \mid F(\vec{x}, do(a,s)) \equiv \Phi(\vec{x}, a, s) \in \mathcal{D}_{ssa}\}$
- $\mathcal{DD}_{poss} = \{Poss(A(\vec{x}), s) \equiv \Psi(\vec{x}, a, s) \wedge$ $NextOrigBounded_b(s) \mid Poss(A(\vec{x}), s) \equiv \Psi(\vec{x}, a, s) \in \mathcal{D}_{poss}\}$

Intuitively $\mathcal{DD}$ extends $\mathcal{D}$ with primed copies of fluents, which are axiomatized to act, in any situation, as the original ones as long as the *original theory remains bounded* by b in that situation, otherwise they become empty (and actions cannot be executed according to *Poss*.) It is easy to show the following key property for $\mathcal{DD}$.

LEMMA 1. $\mathcal{DD} \models \forall s.(\forall \hat{s}.\hat{s} < s \supset NextOrigBounded_b(\hat{s})) \supset \forall \vec{x}.(F'(\vec{x}, s) \equiv F(\vec{x}, s))$.

Now we define a new action theory $\mathcal{D}'$ which can be considered a sort of projection of $\mathcal{DD}$ over the primed fluents only. Let $\mathcal{D}'$ be:

- $\mathcal{D}'_{S_0} = \{\phi[\vec{F}/\vec{F'}] | \phi \in \mathcal{D}_{S_0}\}$.
- $\mathcal{D}'_{ssa} = \{F'(\vec{x}, do(a,s)) \equiv \Phi[\vec{F}/\vec{F'}](\vec{x}, a, s) \wedge$ $NextOrigBounded_b[F/F'](s) \mid$ $F(\vec{x}, do(a,s)) \equiv \Phi(\vec{x}, a, s) \in \mathcal{D}_{ssa}\}$
- $\mathcal{D}'_{poss} = \{Poss(A(\vec{x}), s) \equiv \Psi[\vec{F}/\vec{F'}](\vec{x}, a, s) \wedge$ $NextOrigBounded_b[F/F'](s) \mid$ $Poss(A(\vec{x}), s) \equiv \Psi(\vec{x}, a, s) \in \mathcal{D}_{poss}\}$

Notice that $\mathcal{D}'$ is bounded by construction if $\mathcal{D}'_{S_0}$ is, and furthermore it preserves the information about the *original theory* being bounded at the next step, though in terms of primed fluents. Exploiting the above lemma on $\mathcal{DD}$ and the construction of $\mathcal{D}'$, we can show that $\mathcal{D}'$ has the following notable property:

LEMMA 2. $\mathcal{D} \models AGNextOrigBounded_b(S_0)$ *iff* $\mathcal{D}' \models AGNextOrigBounded_b[\vec{F}/\vec{F'}](S_0)$.[4]

Proof (sketch). By Lemma 1, it is immediate to see that $\mathcal{D} \models AGNextOrigBounded_b(S_0)$ implies $\mathcal{D}' \models AGNextOrigBounded_b[\vec{F}/\vec{F'}](S_0)$. For the opposite direction, suppose that $\mathcal{D}' \models AGNextOrigBounded_b[\vec{F}/\vec{F'}](S_0)$, but $\mathcal{D} \models AGNextOrigBounded_b(S_0)$ does not hold. This means that there exists a model of $\mathcal{D}$ and a situation S where $\neg NextOrigBounded_b(S)$ holds, though in all previous situations $s < S$ we have that $NextOrigBounded_b(s)$ holds. Now by Lemma 1, we have that we can construct a model for $\mathcal{D}'$ such that the truth values of F are replicated in F' as long as $NextOrigBounded_b$ holds in the previous situation. So in S, we must have $\neg NextOrigBounded_b(S)$, which is a contradiction. □

By Lemma 2, since $\mathcal{D}'$ is bounded by b if $\mathcal{D}'_{S_0}$ is, it follows that:

THEOREM 1. *Given a BAT whose initial situation description is bounded by b, then checking whether the entire theory is bounded by b is decidable.*

Notice that we pose no restriction on the initial situation description except that it is representable in first-order logic, hence checking its boundedness remains undecidable:

THEOREM 2. *Given a FO description of the initial situation $\mathcal{D}_0$ (a FO theory without functions, except for constants, and interpreted over standard names) and a bound b, it is undecidable to check whether all models of $\mathcal{D}_0$ are bounded by b.*

[4]Notice that $NextOrigBounded_b[\vec{F}/\vec{F'}]$ expresses that in the original theory the next situations are bounded, though now syntactically replacing original fluents with their primed version.

Proof (sketch). By reduction to FO unsatisfiability. Suppose we have an algorithm to check whether a FO theory $\mathcal{D}_0$ is bounded by 0. Then we would have an algorithm to check (un)-satisfiability of $\mathcal{D}_0$. Indeed consider for a fixed fluent $\hat{F}$:

$$\hat{\mathcal{D}}_0 = (\mathcal{D}_0 \wedge \exists \vec{x}.\hat{F}(\vec{x}, S_0)) \vee (\bigwedge_{F \in \mathcal{F}} \forall \vec{x}.\neg F(\vec{x}, S_0))$$

Notice that $\bigwedge_{F \in \mathcal{F}} \forall \vec{x}.\neg F(\vec{x}, S_0)$ has only models bounded by 0, while $\exists \vec{x}.\hat{F}(\vec{x}, S_0)$ has only models with at least one tuple (and thus one object) in $\hat{F}$. Hence we get that $\hat{\mathcal{D}}_0$ is bounded by 0 iff $\mathcal{D}_0$ is unsatisfiable. A similar argument holds for every bound b. □

Nonetheless in many cases we know by construction that the initial situation is bounded. In such cases the proof technique of Th. 1 provides an effective way to check if the entire theory is bounded.

6. PROGRESSING BOUNDED THEORIES

We now proceed to show that all bounded action theories are first-order progressable. We start by showing a general result about progression that holds for more general action theories.

THEOREM 3. *Let $\mathcal{D}$ be a basic action theory without the standard names restriction. Let $Prog(\varphi, \alpha)$ denote the progression of any arbitrary sentence φ uniform in S_0 w.r.t. a ground action α and $(\mathcal{D} - \mathcal{D}_0) \cup \{\varphi\}$.[5] The following holds:*

$$Prog(\bigvee_i \varphi_i, \alpha) \equiv \bigvee_i Prog(\varphi_i, \alpha),$$

where φ_i are (possibly second-order) sentences uniform in S_0.

PROOF. ($\Leftarrow$) Suppose not. Then there exists a model M such that $M \models Prog(\varphi_k, \alpha)$ for some k, and $M \not\models Prog(\bigvee_i \varphi_i, \alpha)$. By the definition of progression, since $M \not\models Prog(\bigvee_i \varphi_i, \alpha)$, there exists no M' such that $M' \models (\mathcal{D}-\mathcal{D}_0) \cup \{\bigvee_i \varphi_i\}$ and $M \sim_{S_\alpha} M'$. Also by the same definition, since $M \models Prog(\varphi_k, \alpha)$, there exists an M'' such that $M'' \models (\mathcal{D}-\mathcal{D}_0) \cup \{\varphi_k\}$ and $M \sim_{S_\alpha} M''$, which, in turn, implies that there exists an M'' such that $M'' \models (\mathcal{D} - \mathcal{D}_0) \cup \{\bigvee_i \varphi_i\}$ and $M \sim_{S_\alpha} M''$. Hence we get a contradiction.

($\Rightarrow$) Suppose not. Then there exists a model M, such that $M \models Prog(\bigvee_i \varphi_i, \alpha)$, and $M \not\models Prog(\varphi_i, s_\alpha)$ for any i. By the definition of progression, since $M \models Prog(\bigvee_i \varphi_i, s_\alpha)$ there exists a M' such that $M' \models (\mathcal{D}-\mathcal{D}_0) \cup \{\bigvee_i \varphi_i\}$ and $M \sim_{S_\alpha} M'$, therefore there exists a model M' such that $M' \models (\mathcal{D} - \mathcal{D}_0) \cup \{\varphi_k\}$ and $M \sim_{S_\alpha} M'$ for some k. Also by the same definition, since $M \not\models Prog(\varphi_i, \alpha)$ for any i, there exists no model M'' such that $M'' \models (\mathcal{D} - \mathcal{D}_0) \cup \{\varphi_i\}$ and $M \sim_{S_\alpha} M''$ for any i. Hence we get a contradiction. □

Now we turn to bounded action theories. By results in [4] it follows that the models of the initial situation description $\mathcal{D}_0$ of any bounded action theory $\mathcal{D}$ can be partitioned into a *finite* set of isomorphism types whose active domain is bounded. Each isomorphism type with an active domain of size b can be captured by a *characteristic sentence* of the form:

$$\exists w_1, \ldots, w_b.AllDistinct(w_1, \ldots, w_b) \wedge \\ \wedge_{i=1}^{n} \forall \vec{x_i}.(F_i(x_i, S_0) \equiv \phi_i(\vec{x}_i, w_1, \ldots, w_b))$$

where $AllDistinct(w_1, \ldots, w_b)$ is a formula of inequalities stating that all $w_1, \ldots, w_b$ assume distinct values; and $\phi_i(\vec{x}_i, w_1, \ldots, w_b)$ is a formula of equalities and inequalities that gives the extension of the fluent $F_i(x_i, S_0)$ in the models of the isomorphism type:

[5]A second-order progression of φ is always guaranteed to exist by results in [9].

THEOREM 4. *The characteristic sentence above captures one isomorphism type of $\mathcal{D}_0$ for a bounded basic action theory $\mathcal{D}$.*

Proof (sketch). Recall that in a bounded $\mathcal{D}$ there is only a finite number of fluent atoms $F(\vec{x}, s)$ for every situation s and in particular in the initial situation S_0 that $\mathcal{D}_0$ specifies. Indeed it can be shown that all models of the characteristic sentence belong to one isomorphism type and all models that are isomorphic to one of them are also models of the characteristic sentence. □

Next we observe that a characteristic sentence for an isomorphism type is actually a *relatively complete initial knowledge base with bounded unknowns*, cf. Definition 3 in [19]. This implies that each of the finite number of the characteristic sentences has in fact a first-order progression (which is expressed again as a relatively complete sentence with bounded unknowns) by Theorem 1 in [19]. Now applying Theorem 3 for the form of $\mathcal{D}_0$ expressed as a disjunction of the (finitely many) characteristic sentences, we get the main result of this section:

THEOREM 5. *All bounded action theories are iteratively first-order progressable.*

Note that while progression applies to a single action, since the result of progression is still a disjunction of characteristic sentences, we can apply it iteratively to deal with arbitrary action sequences.

This view of the knowledge base as a disjunction of a finite set of characteristic sentences provides a practical abstraction based on the boundedness assumption that also illustrates how it can be updated. We next show one step of progression for Example 3.

Example 4. The initial knowledge base can be logically equivalently expressed as the disjunction of two characteristic sentences, $\psi_1 \vee \psi_2$, the first of which is the following:

$$\begin{array}{l} \exists w.Distinct(w, \{R1, R2, Hold1, Hold2, Hold3\}) \wedge \\ \quad Distinct(w, \{PaintStn, ShipDock, I1, I2\}) \wedge \\ \quad \forall r(IsRobot(r) \equiv r = R1 \vee r = R2) \wedge \\ \quad \forall r(HandlesHeavy(r) \equiv r = R1) \wedge \\ \quad \forall l(IsLoc(l) \equiv l = Hold1 \vee l = Hold2 \vee l = Hold3) \wedge \\ \quad \forall x \forall l\{At(x, l, S_0) \equiv (x = I1 \wedge l = Hold1) \vee \\ \qquad (x = I2 \wedge l = Hold2) \vee (x = I3 \wedge l = w)\} \wedge \\ \quad \forall x(Painted(x, S_0) \equiv false) \wedge \forall x(Shipped(x, S_0) \equiv false) \wedge \\ \quad \forall x(Heavy(x) \equiv x = I1) \wedge \forall x(Fragile(y) \equiv (y = I2 \vee y = w)), \end{array}$$

where $Distinct(x, \{t_1, \ldots, t_n\})$ denotes that x is distinct from all elements in the set. The second characteristic sentence is exactly the same except for the last two conjuncts in which item w is represented as heavy instead of fragile.

Now consider the action $move(R1, I1, ShipDock)$. Using the progression method explained in [19] for each characteristic sentence separately, it is easy to show that the progressed version of the knowledge base is logically equivalent to $\psi_1 \vee \psi_2$ after the location of $I1$ is updated in the specification of At. □

7. VERIFYING ONLINE EXECUTIONS

Dynamic properties over online executions of BATs can be expressed using a variant of the μ-calculus, called $\mu\mathcal{LO}$, whose syntax is as follows:

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

where φ is an arbitrary closed uniform *situation-suppressed* (i.e., with all situation arguments in fluents suppressed) situation calculus FO formula, whose constants must appear in $\mathcal{D}$, 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.

As to semantics, since $\mu\mathcal{LO}$ contains formulae with predicate free variables, given an action theory $\mathcal{D}$, we introduce a predicate variable valuation $\mathcal{V}$, i.e., a mapping from predicate variables Z to situation terms. Then we assign semantics to formulae by associating with $\mathcal{D}$ and $\mathcal{V}$ an *extension function* $(\cdot)^{\mathcal{D}}_{\mathcal{V}}$ which maps $\mu\mathcal{LO}$ formulae to subsets of ground situation terms. We denote by Γ the set of *ground executable situation terms*, inductively defined as follows: 1. $S_0 \in \Gamma$; 2. If $\sigma \in \Gamma$, A is an action type with parameters $\vec{x}$, $\vec{n} \in \vec{\mathcal{N}}$ is a vector of names s.t. $|\vec{n}| = |\vec{x}|$, and $\mathcal{D} \models Poss(A(\vec{n}), \sigma)$, then $do(A(\vec{n}), \sigma) \in \Gamma$.

The extension function is defined inductively as follows:

$$
\begin{array}{lcl}
(holds(\varphi))^{\mathcal{D}}_{\mathcal{V}} & = & \{\sigma \in \Gamma \mid \mathcal{D} \models \varphi[\sigma]\} \\
(\neg\Phi)^{\mathcal{D}}_{\mathcal{V}} & = & \Gamma - (\Phi)^{\mathcal{D}}_{\mathcal{V}} \\
(\Phi_1 \wedge \Phi_2)^{\mathcal{D}}_{\mathcal{V}} & = & (\Phi_1)^{\mathcal{D}}_{\mathcal{V}} \cap (\Phi_2)^{\mathcal{D}}_{\mathcal{V}} \\
(\langle - \rangle\Phi)^{\mathcal{D}}_{\mathcal{V}} & = & \{\sigma \in \Gamma \mid \bigvee_A \exists \vec{n} \in \vec{\mathcal{N}}. \\
 & & \quad do(A(\vec{n}), \sigma) \in \Gamma \wedge do(A(\vec{n}), \sigma) \in (\Phi)^{\mathcal{D}}_{\mathcal{V}}\} \\
(Z)^{\mathcal{D}}_{\mathcal{V}} & = & \mathcal{V}(Z) \\
(\mu Z.\Phi)^{\mathcal{D}}_{\mathcal{V}} & = & \bigcap\{\mathcal{E} \subseteq \Gamma \mid (\Phi)^{\mathcal{D}}_{\mathcal{V}[Z/\mathcal{E}]} \subseteq \mathcal{E}\}
\end{array}
$$

Notice that given a closed uniform situation-suppressed situation calculus formula φ, by a slight abuse of notation, we denote by $\varphi[\sigma]$ the formula φ with the situation argument reintroduced and assigned to σ. Also, given a valuation $\mathcal{V}$, a predicate variable Z, and a set $\mathcal{E}$ of situation terms, we denote by $\mathcal{V}[Z/\mathcal{E}]$ the valuation obtained from $\mathcal{V}$ by changing the value of Z to $\mathcal{E}$. Notice also that when a $\mu\mathcal{LO}$ formula Φ is closed (w.r.t. predicate variables), its extension $(\Phi)^{\mathcal{D}}_{\mathcal{V}}$ does not depend on the predicate valuation $\mathcal{V}$. The only formulas of interest in verification are those that are closed.

We say that a theory $\mathcal{D}$ *entails* a closed $\mu\mathcal{LO}$ formula Φ, written $\mathcal{D} \models \Phi$, if $S_0 \in (\Phi)^{\mathcal{D}}_{\mathcal{V}}$ (for any valuation $\mathcal{V}$, which is in fact irrelevant for closed formulas).

We next show some examples. For simplicity we adopt the notation of the well-known logic CTL*, which can be thought of as a fragment of $\mu\mathcal{LO}$. In particular $E\varrho$ and $A\varrho$ respectively express that there exists an (infinite) path (of action executions) satisfying ϱ and that all paths satisfy ϱ; and $G\varrho$ and $F\varrho$ respectively express that along a path ϱ always holds and along a path ϱ eventually holds.

Example 5. For the agent in Example 1, one property that we may want to verify is that it is possible for the agent to eventually know that it has shipped all items that were in the factory. This can be expressed in our language as a least fixpoint formula $\mu Z.\neg\exists x\exists l.At(x,l) \vee \langle - \rangle Z$ or, in CTL*: $EF\neg\exists x\exists l.At(x,l)$.

In the above, we rely on the fact that if there are no items left in the factory, then all items that were there must have been shipped. It is easy to check that the theory of Example 1, $\mathcal{D}_1$, entails this formula. More generally, a formula $EF\varphi$ represents an instance of a conformant planning problem. It is satisfied by a theory if there exists an executable sequence of actions such that afterwards the agent knows that φ holds. In fact we can also show that the above property can always be achieved: $\mathcal{D}_1 \models AGEF\neg\exists x\exists l.At(x,l)$.

Another property that can be shown to hold for this example domain is that it is possible for the agent to eventually know that it has shipped all items that were in the factory and that every shipped item was painted. We express this as follows:
$\mathcal{D}_1 \models E((F\neg\exists x\exists l.At(x,l)) \wedge (G\forall x(Shipped(x) \supset Painted(x)))$.

Example 6. For the agent in Example 2, with associated theory $\mathcal{D}_2$, we can show that: $\mathcal{D}_2 \models EF((\forall l\neg\exists x.At(x,l)) \wedge F(\forall l\exists x.At(x,l)))$, i.e. it is possible to eventually have all items shipped out of the factory and then later to eventually have all locations filled with items. Moreover, we can also show that always if an item is at the shipping dock it can be shipped out:
$\mathcal{D}_2 \models AG((\exists x.At(x, ShipDock)) \supset \langle - \rangle(\neg\exists x.At(x, ShipDock)))$.
However, this is not the case for other locations, as it is possible for all locations to become occupied, at wich point the agent must ship the item at the shipping dock before it can move the item at another location out: $\mathcal{D}_2 \models \neg AG((\exists x\exists l.At(x,l)) \supset \langle - \rangle(\neg\exists x\exists l.At(x,l)))$.

It can be easily seen that because Γ and $\mathcal{N}$ are infinite in general, one cannot check whether $\mathcal{D} \models \Phi$ using an exhaustive search procedure, as typically done in classical μ-calculus model checking [5]. This is obviously true also for bounded theories. However, under the boundedness assumption, the construction of a finite structure becomes possible, which can be used to easily carry out the verification task. The following result, whose proof is discussed in the rest of the section is based on this observation.

THEOREM 6. *Let $\mathcal{D}$ be a BAT bounded by b and Φ a closed $\mu\mathcal{LO}$ formula. Then, deciding whether $\mathcal{D} \models \Phi$ is decidable.*

The proof involves two steps. Firstly, we provide an alternative semantics of $\mu\mathcal{LO}$ formulas, equivalent, or more precisely *online-execution bisimilar* to the one above, that is based on a transition system $T_{\mathcal{D}}$ derived from $\mathcal{D}$, which we call *progression-based*, that captures the state evolution of the theory. In the second step, we exploit the results on progression of bounded theories to show that a finite-state transition system T_F can be effectively constructed, and then prove that it is equivalent, for the purpose of verification, to $T_{\mathcal{D}}$. Since standard model checking algorithms can be executed on T_F, this is enough to guarantee decidability of verification.

We start by introducing the notion of transition system. Formally, an *(online-execution) transition system* (TS) is a tuple $T = \langle Q, q_0, \lambda, \rightarrow \rangle$, where: (i) Q is the set of *possible states*; (ii) $q_0 \in Q$ is the *initial state*; (iii) $\lambda : Q \rightarrow 2^{\mathcal{L}_{S_0}}$ is the *labeling function*, associating each state q with a set $\mathcal{D}_q$ of uniform situation-suppressed sentences over standard names $\mathcal{N}$; (iv) $\rightarrow \subseteq Q \times Q$ is the *transition relation*. As can be seen, this is a special case of standard labelled transition system, where states are labelled by (possibly non first-order) logical theories. We call this class of TSs *online-execution*, to stress that, as it will be clarified later on, they can accommodate all the information relevant to online executions.

Next, we detail the semantics of μ-calculus formulas Φ over a TS T, for a valuation $\mathcal{V}$:

$$
\begin{array}{lcl}
(holds(\varphi))^{T}_{\mathcal{V}} & = & \{q \in Q \mid \lambda(q) \models \varphi\} \\
(\neg\Phi)^{T}_{\mathcal{V}} & = & Q - (\Phi)^{T}_{\mathcal{V}} \\
(\Phi_1 \wedge \Phi_2)^{T}_{\mathcal{V}} & = & (\Phi_1)^{T}_{\mathcal{V}} \cap (\Phi_2)^{T}_{\mathcal{V}} \\
(\langle - \rangle\Phi)^{T}_{\mathcal{V}} & = & \{q \in Q \mid \exists q'.q \rightarrow q' \wedge q' \in (\Phi)^{T}_{\mathcal{V}}\} \\
(Z)^{T}_{\mathcal{V}} & = & \mathcal{V}(Z) \\
(\mu Z.\Phi)^{T}_{\mathcal{V}} & = & \bigcap\{\mathcal{E} \subseteq Q \mid (\Phi)^{T}_{\mathcal{V}[Z/\mathcal{E}]} \subseteq \mathcal{E}\}
\end{array}
$$

We say that T *verifies* a closed $\mu\mathcal{LO}$ formula Φ, written $T \models \Phi$ if $q_0 \in (\Phi)^{\mathcal{D}}_{\mathcal{V}}$ (for any valuation $\mathcal{V}$, which is irrelevant). This is essentially the standard semantics of the μ-calculus [5], with satisfaction replaced by entailment on state labels.

We can now associate a theory $\mathcal{D}$ with its *progression-based* transition system $T_{\mathcal{D}} = \langle Q, q_0, \lambda, \rightarrow \rangle$, defined as follows:

- $Q = \Gamma$ (recall Γ is the set of executable action terms);
- $q_0 = S_0$;
- λ is inductively defined as follows:
  - $\lambda(q_0) = \tilde{\mathcal{D}}_0$, where $\tilde{\mathcal{D}}_0$ is the situation-suppressed version of $\mathcal{D}_0$;
  - if $q \rightarrow q'$ and $q' = do(A(\vec{n}), q)$, for some action type A with parameters $\vec{x}$ and names $\vec{n} \in \vec{\mathcal{N}}$, then $\lambda(q')$ is the situation-suppressed version of the *progression* of $\lambda(q)[q]$ w.r.t. $A(\vec{n})$.
- $\rightarrow \subseteq Q \times Q$ is the *transition relation* s.t. $q \rightarrow q'$ iff $q' = do(A(\vec{n}), q)$, for some action type A and names $\vec{n} \in \vec{\mathcal{N}}$.

Observe that $T_{\mathcal{D}}$ is essentially the (infinite) situation tree labelled by the progressed theory, at each step. As such, it retains all the information entailed by $\mathcal{D}$ at every situation. This, as shown by the following result, is all and only the information needed to evaluate the semantics of any $\mu\mathcal{LO}$ formula.

THEOREM 7. *Let $\mathcal{D}$ be a BAT (not necessarily bounded), and Φ a $\mu\mathcal{LO}$ formula, then* $(\Phi)^{\mathcal{D}}_{\mathcal{V}} = (\Phi)^{T_{\mathcal{D}}}_{\mathcal{V}}$.

Proof (sketch). By induction on the situation terms and the formula, using the classical results about progression [9]. □

As a corollary, we have that: $\mathcal{D} \models \Phi$ iff $T_{\mathcal{D}} \models \Phi$. Thus, we can check whether $\mathcal{D} \models \Phi$, using the progression-based transition system $T_{\mathcal{D}}$, instead of the theory $\mathcal{D}$. Notice, though, that $T_{\mathcal{D}}$ is also infinite. Thus, even assuming $T_{\mathcal{D}}$ available, this result is not enough, alone, to prove decidability of verification. To overcome this obstacle, we construct a finite-state TS that is equivalent to $T_{\mathcal{D}}$ w.r.t. verification, and that we can use to effectively perform the check. To this end, we first introduce the notions of *logical equivalence modulo renaming* between theories, and *online-execution bisimulation* between TSs, together with stating relevant results about them.

Two theories $\mathcal{D}$ and $\mathcal{D}'$, over the same signature and standard names $\mathcal{N}$, are said to be *logically equivalent modulo renaming*, written $\mathcal{D} \sim \mathcal{D}'$, if there exists a bijection $h : \mathcal{N} \rightarrow \mathcal{N}$ s.t. $\mathcal{D} \models h(\mathcal{D}')$ and $\mathcal{D}' \models h^{-}(\mathcal{D})$ (for h^{-} the inverse of h), where: $h(\mathcal{D}')$ stands for the theory obtained from $\mathcal{D}'$ by replacing each constant n occurring in it by $h(n)$; and similarly $h^{-}(\mathcal{D})$ is the theory obtained by replacing each constant of n occurring in $\mathcal{D}$, by $h^{-}(n)$. We say that h *preserves* C, for $C \subseteq \mathcal{N}$ a set of constants, if $h(c) = c$, for every $c \in C$. We write $\mathcal{D} \sim_C \mathcal{D}'$ to denote that $\mathcal{D}$ and $\mathcal{D}'$ are logically equivalent modulo renamings preserving C.

Logical equivalence modulo renaming captures the intuition that $\mathcal{D}$ and $\mathcal{D}'$ have exactly the same models, modulo element renaming. It can be proven that theories that are logically equivalent modulo renaming satisfy exactly the same closed formulas.

THEOREM 8. *Let $\mathcal{D}$ and $\mathcal{D}'$ be two theories over same signature and standard names $\mathcal{N}$, s.t. $\mathcal{D} \sim_C \mathcal{D}'$, for $C \subseteq \mathcal{N}$ a set of constants. Then, for any closed formula φ with constants in C, we have that $\mathcal{D} \models \varphi$ iff $\mathcal{D}' \models \varphi$.*

Thus, to check whether a closed formula φ is entailed by a class of logically equivalent (modulo renaming) theories, it is sufficient to evaluate the formula against an arbitrary representative of the class.

Logical equivalence modulo renaming can be applied to state labels of TSs, to lift standard *bisimulation* to online-execution TSs. To this end, let $T_1 = \langle Q_1, q_{10}, \rightarrow_1, \lambda_1 \rangle$ and $T_2 = \langle Q_2, q_{20}, \rightarrow_2, \lambda_2 \rangle$ be two TSs, and $C \subseteq \mathcal{N}$ a set of constants. An *(online-execution) bisimulation* between T_1 and T_2 preserving C is a relation $B \subseteq Q_1 \times Q_2$ s.t. $B(q_1, q_2)$ implies:

- $\lambda_1(q_1) \sim_C \lambda_2(q_2)$;
- 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 *(online-execution) bisimilar* w.r.t. C, written $T_1 \approx_C T_2$, if $\langle q_{10}, q_{20} \rangle \in B$, for some bisimulation B preserving C. As standard, bisimilarity is an equivalence relation.

A notable property of online-execution bisimulations is that they preserve entailment of $\mu\mathcal{LO}$ formulas.

THEOREM 9. *Given two TSs T_1 and T_2, and a set of constants $C \subseteq \mathcal{N}$, if $T_1 \approx_C T_2$ then, for every closed $\mu\mathcal{LO}$ formula Φ, with constants in C, we have that $T_1 \models \Phi$ iff $T_2 \models \Phi$.*

Proof (sketch). Analogous to bisimulation-invariance theorem for standard μ-calculus [5], using Th. 8 for local conditions, i.e., to evaluate uniform (situation-suppressed) closed FO formulas. □

Observe that this result can hold, in principle, even between an infinite-, say $T_{\mathcal{D}}$, and a finite-state TS, say T_F. When this is the case, the verification can be performed on T_F using standard μ-calculus model checking techniques, which essentially perform fixpoint computations on a finite state space. However, two major obstacles prevent this approach from being effective. Firstly, Th. 9 applies only provided T_F is available, thus the question arises whether it is possible to compute one such TS. Secondly, μ-calculus model checking requires a procedure to check whether state labelings of T_F, i.e., theories, entail atomic subformulas of Φ, a question that is, in general, undecidable. For the former, we next describe an actual procedure. For the latter, it suffices to observe that, under boundedness, decidability of entailment follows straightforwardly as a special case of Th. 15 of [4].

We construct T_F using Algorithm 1, which takes a BAT $\mathcal{D}$ bounded by b as input, and returns a finite-state TS $T_F = \langle U, u_0, \gamma, \rightsquigarrow \rangle$ bisimilar to $T_{\mathcal{D}}$. This procedure generates T_F

Algorithm 1 Computation of a finite-state TS bisimilar to $T_{\mathcal{D}}$.

```
U := {u_0}; γ(u_0) := D̃_0; ⇝ := ∅;
for all action type A with parameters x⃗ do
  let C_γ(u) be the set of constants occurring in γ(u);
  let O ⊂ N be any (finite) set s.t. |O| = |x⃗| and
      O ∩ (C ∪ C_γ(u)) = ∅;
  for all parameter assignments v : x⃗ → C_γ(u) ∪ O do
    let D_u,α be the progression of γ(u) w.r.t. α = A(v(x⃗));
    if there exists u' ∈ U s.t. D_u,α ∼_C γ(u') then
      ⇝ := ⇝ ∪{u ⇝ u'};
    else
      let U := U ⊎ {u'}, for u' a fresh state, with γ(u') = D_u,α,
      and ⇝ := ⇝ ∪{u ⇝ u'};
    end if
  end for
end for
```

by iteratively progressing $\mathcal{D}$, starting from the initial (situation-suppressed) knowledge base. 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, w.r.t. logical equivalence modulo renaming, of progressions that can be obtained after applying all possible actions. This is actually achieved by including in O a distinct value for each parameter, distinct also from all the elements of C and $C_{\gamma(u)}$. Then, to guarantee coverage of all equivalence classes, all action types and all assignments of parameters to

$O \cup C \cup C_{\gamma(u)}$ are considered. Notice that by the boundedness assumption and Th. 5, the progression of $\mathcal{D}_{u,\alpha}$ is indeed computable, an obvious necessary condition in order for the algorithm to terminate. Similarly, it is required that testing the condition of the **if** statement be decidable, which is implied by the next result, once observed that, $\mathcal{D}$ being b-bounded, so are all of the theories labeling the states of U.

THEOREM 10. *Let $\mathcal{D}_{01}$ and $\mathcal{D}_{02}$ be two finite sets of uniform situation-suppressed sentences over standard names $\mathcal{N}$, and let C be a finite set of constants possibly occurring in them. If $\mathcal{D}_{01}$ and $\mathcal{D}_{02}$ are bounded by a given b, then checking whether $\mathcal{D}_{01} \sim_C \mathcal{D}_{02}$ (i.e. $\mathcal{D}_{01}$ and $\mathcal{D}_{02}$ are logically equivalent modulo renaming preserving C) is decidable.*

Proof (sketch). Associate each theory $\mathcal{D}_{0i}$ ($i = 1, 2$) with the formula $\Phi_i = Bounded_b \wedge \exists \vec{x}.AllDistinct(\vec{x}).\mathcal{D}_{0i}[\vec{c}/\vec{x}]$, where $\vec{c}$ are all the constants different from those in C, that occur in $\mathcal{D}_{0i}$, and $\vec{x}$ is a fresh set of variables s.t. $|\vec{c}| = |\vec{x}|$. Finally, with a slight abuse of notation, we use $\mathcal{D}_{0i}[\vec{c}/\vec{x}]$ for the conjunction of formulas in $\mathcal{D}_{0i}$, with the constants in $\vec{c}$ syntactically replaced by the variables in $\vec{x}$. It then suffices to check the equivalence (without modulo renaming) of Φ_1 and Φ_2, which is decidable by boundedness. □

Termination of the algorithm is shown by the following result.

THEOREM 11. *If $\mathcal{D}$ is a BAT bounded by b, then Algorithm 1 terminates and produces a T_F with a finite number of states in U.*

Proof (sketch). Consequence of the fact that $\mathcal{D}$ is bounded by b, thus only finitely many equivalence classes of theories, w.r.t. logical equivalence modulo renaming, exist, which constitute U. □

Finally, we obtain that Algorithm 1 returns a TS bisimilar to $T_{\mathcal{D}}$.

THEOREM 12. *If $\mathcal{D}$ is a BAT bounded by b, then $T_F \approx_C T_{\mathcal{D}}$.*

Proof (sketch). We show the thesis by co-induction. In particular, let $B \subseteq Q \times U$ s.t. $\langle q, u \rangle \in B$ iff $q \sim_C u$. Then it can be shown that B is a bisimulation s.t. $\langle q_0, u_0 \rangle \in B$. □

As a consequence, we obtain the desired result, i.e., Th. 6.

8. CONCLUSION

We have proposed a decidable framework for verifying agents with bounded beliefs operating in infinite state domains. The agent has bounded beliefs if the action theory that models the agent's beliefs and deliberation process entails that the number of tuples that belong to any fluent in any situation is bounded by a constant. We have shown that this boundedness condition is sufficient to ensure that the agent's belief state in any situation can be progressed and remain first-order representable. The framework allows complex subjective temporal properties to be specified and verified over online executions of the agent, i.e., executions where the agent only performs actions that it knows are feasible. As well, we have shown that checking whether boundedness is maintained is decidable.

We observe that in the case where the initial situation description is in the form studied in [19], computing progression becomes particularly easy. In turn, this simplifies verification by making it simple to compute the finite transition system on which the model checking algorithm is applied. Our approach is related but quite different from that in [3]. The latter is based on a version of the situation calculus with a knowledge modality and develops a more restrictive notion of bounded epistemic action theory, 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.

In future work, we want to extend our online executions verification framework to deal with sensing actions, and partially observable actions and forgetting (which helps to maintain boundedness); this will require changes to the specification language as it introduces forms of nondeterminism that are not under the agent's control. We also want to allow some forms of quantification across situations in the specification language. Finally, we want to extend the framework to support the verification of agent programs.

Acknowledgements. The authors acknowledge support of EU Project FP7-ICT 318338 (OPTIQUE) and Sapienza Award 2013 "Spiritlets" project.

9. REFERENCES

[1] B. Bonet. Conformant plans and beyond: Principles and complexity. *Artificial Intelligence*, 174(3-4):245 – 269, 2010.

[2] G. De Giacomo, Y. Lespérance, H. J. Levesque, and S. Sardina. IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents. In *Multi-Agent Programming: Languages, Tools and Applications*. Springer, 2009.

[3] G. De Giacomo, Y. Lespérance, and F. Patrizi. Bounded Epistemic Situation Calculus Theories. In *Proc. of IJCAI'13*.

[4] G. De Giacomo, Y. Lespérance, and F. Patrizi. Bounded Situation Calculus Action Theories and Decidable Verification. In *Proc. of KR'12*.

[5] E. A. Emerson. Model Checking and the Mu-calculus. In *Descriptive Complexity and Finite Models*, 1996.

[6] M. Fisher and M. Wooldridge. On the Formal Specification and Verification of Multi-Agent Systems. *Int. J. Cooperative Inf. Syst.*, 6(1):37–66, 1997.

[7] G. Lakemeyer and H. J. Levesque. Only-knowing: Taking it Beyond Autoepistemic Reasoning. In *Proc. of AAAI'05*.

[8] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *JLP*, 31:59–84, 1997.

[9] F. Lin and R. Reiter. How to Progress a Database. *Artificial Intelligence*, 92(1-2):131–167, 1997.

[10] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *Proc. of CAV'09*.

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

[12] F. Pirri and R. Reiter. Some Contributions to the Metatheory of the Situation Calculus. *J. ACM*, 46(3):261–325, 1999.

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

[14] 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*.

[15] 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*.

[16] R. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1–2):1–39, 2003.

[17] S. Shapiro, Y. Lespérance, and H. J. Levesque. The Cognitive Agent Specification Language and Verification Environment. In *Specification and Verification of Multi-Agent Systems/Programs*. Springer, 2010.

[18] S. Vassos and H. J. Levesque. How to progress a database III. *Artificial Intelligence*, 195:203–221, 2013.

[19] S. Vassos and F. Patrizi. A Classification of First-Order Progressable Action Theories in Situation Calculus. In *Proc. of IJCAI'13*.

[20] M. Wooldridge and A. Lomuscio. A Computationally Grounded Logic of Visibility, Perception, and Knowledge. *Logic Journal of the IGPL*, 9(2):257–272, 2001.