Giuseppe De Giacomo
Yves Lespérance
Fabio Patrizi
Stavros Vassos

Progression and Verification of Situation Calculus Agents with Bounded Beliefs

Abstract. 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. We take a *first-person* view and use the theory to capture what the agent believes about the domain of interest and the actions affecting it. In this paper, we study verification of temporal properties over online executions. These are executions resulting from agents performing only actions that are feasible according to their beliefs. To do so, we first examine progression, which captures belief state update resulting from actions in the situation calculus. We show that, for bounded action theories, progression, and hence belief states, can always be represented as a bounded first-order logic theory. Then, based on this result, we prove decidability of temporal verification over online executions for bounded action theories.

Keywords: Reasoning about actions, Situation calculus, Progression, Online execution, Verification of agent behaviors, Mu-Calculus.

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 [11]. The situation calculus [33,36] is a first-order logical framework for reasoning about action, where several issues have been addressed, such as the frame problem, time, continuous change, complex actions and processes, uncertainty, and many others. It is also the basis of the Golog family of agent programming languages [10,27] and has been used to develop rich theories of agent mental states and actions [44].

Special Issue: Logical Aspects of Multi-Agent Systems
Edited by **Nils Bulling and Wiebe van der Hoek**

Studia Logica
DOI: 10.1007/s11225-015-9626-z
Published online: 05 September 2015

We use situation calculus action theories to express the mental model of an agent that can deliberate and act in the world. We take a *first-person* view and use the theory to capture what the agent believes about the domain of interest and the actions affecting it. In other words, the agent represents its beliefs[1] about the world as a situation calculus theory and uses it to reason and deliberate about what to do. Once the agent has chosen an action it *executes it in the real world*, and follows such an execution in its mental model constituted by the theory.

Essentially the agent works in a sort of *infinite loop* in which, at each iteration, the agent (i) understands which actions are known to be executable in the current state through reasoning, exploiting the (incomplete) information formalized in the situation calculus theory, (ii) choses one among them (using any suitable deliberation mechanism which we do not model here), and (iii) executes it, advancing to the resulting state.[2] This agent execution regime is known in the literature as *online execution* [10,16] and contrasts with so called *offline execution*, in which the agent reasons about possible action executions but does not perform any actions in the real world.

The main goal of this paper is to analyze the agent's online execution capabilities through verification of temporal properties expressed in a first-order variant of the μ-calculus. For instance, we can easily express that there exists a sequence of actions known to be executable that reaches a state where a goal is true, even if the agent has incomplete information about the world (as represented by the action theory), and hence we can check if a conformant plan [5] exists through verification.

Specifically, we adopt *bounded action theories* [11], a particular class of action theories, for which it was shown that verification (over offline executions) of a very expressive class of first-order μ-calculus temporal properties is *decidable*. Bounded action theories are basic action theories [36], which entail that in all situations the number of object tuples in the extension of each fluent 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 reality facts don't persist indefinitely as everything decays and changes. Moreover, agents often forget facts either because they are not used or because they cannot be reconfirmed. Many examples of domains modeled as bounded action theories are reported in [11],

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

[2] In this paper we do not consider *sensing*, but if we did the sensing results obtained by the execution of the action would be incorporated into the theory; see [16] for details.

which also identifies various ways to obtain boundedness: (i) by strengthening preconditions to block actions where the bound would be exceeded; (ii) by ensuring that actions are *effect bounded* and never make more fluent atoms true than they make false; and (iii) by using *fading fluents* whose strength fades over time unless they are reconfirmed.

Towards the goal of devising decidable techniques for verifying properties of online execution in the case of bounded situation calculus action theories, first we examine *progression* [29] for such kind of theories. By progressing the initial situation description over an action we obtain a new situation description representing all that is known about the situation after the action is performed. More specifically, the fragment of the original theory that talks about the initial situation can be considered the *initial belief state*, and similarly the result of progressing such fragment as the result of executing an action can be considered the *belief state after the action*, and so on.[3] In this sense progression can be thought of as capturing belief state update that results from actions in the situation calculus. Unfortunately, in the general case, progression (and hence such belief states) can only be expressed in second-order logic [29,50]. Here, we show that *for bounded action theories, progression, and belief states, can always be represented in first-order logic*, and discuss how a first-order progression can be constructed.

Often, belief states are a priori thought of as some sort of *first-order theory* about the current world state whose models are the possible alternative world states that the agent thinks it may be in [39,40]. However, in the situation calculus, first-order belief states are not complete in general unless one additionally keeps a description of the past situations; to represent updated belief states without keeping such past information, second-order logic is needed [29,50]. When progression is first-order representable, such first-order belief states are indeed "complete" and no further information (apart from the specification of actions) is needed. Hence for bounded theories, by iterating progression steps we obtain a "computationally grounded" model of agents [52], in the sense that such a model captures how the belief states of agents are generated and updated from the action theory, which describes what is true (according to the agent) and how this evolves as actions are performed.

With this result on progression in place, we investigate the verification of online executions of agents. We show that for a very rich class of temporal

[3]Notice that such a belief state will be complemented by the part of the theory that talks about executability and effects of actions, which remains unchanged through the progression.

properties expressed in a first-order variant of the μ-calculus *verification of online executions is decidable*. This result complements the one in [11], which showed decidability of verification for offline executions.

The rest of this paper is organized as follows. In Sect. 2, we briefly review the situation calculus and the notion of online execution. Then in Sect. 3, we recall the definition of progression. In Sect. 4, we go over the notion of bounded action theory from [11] and give some examples. In Sect. 5, we show our first major result, i.e., that the progression of a bounded action theory can always be represented in first-order logic. Following that in Sect. 6, we introduce a language for expressing temporal properties of online executions (a first-order variant of the μ-calculus), and then show our main result, i.e., that verification of such properties over bounded action theories is decidable. Finally in Sect. 7, we discuss related work, while in Sect. 8, we summarize our contributions and discuss future work.

2. The Situation Calculus and Online Executions

The *situation calculus* [33,36] 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 without loss of generality, we assume that there are no functions other than constants (and do) 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 from the last argument which is of sort situation) are assumed to be of sort object. A special predicate $Poss(a, s)$ is used to state that action a is executable in situation s. The abbreviation $Executable(s)$ means that every action performed in reaching situation s was possible in the situation in which it occurred.

Within the language, one can formulate action theories that describe how the world changes as the result of the available actions. Here, we concen-

trate on *basic action theories* (BATs) as proposed in [35,36]. 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) action precondition axioms stating when actions can be legally performed and characterizing *Poss* ($\mathcal{D}_{ap}$); (FO) successor state axioms describing how fluents change between situations ($\mathcal{D}_{ss}$); (FO) unique name axioms for actions and (FO) domain closure axioms on action types ($\mathcal{D}_{una}$); (SO) unique name and domain closure axioms for object constants ($\mathcal{D}_{coa}$); and (FO) axioms describing the initial configuration of the world ($\mathcal{D}_0$), which we assume finite.[4] Note that successor state axioms encode the causal laws of the domain; they take the place of the so-called effect axioms and provide a solution to the frame problem.

We say that a formula $\phi(s)$ is *uniform in a situation term* s if s is the only situation term it contains, and we will use the term *situation-suppressed* to refer to the formula such that the situation argument in fluents is omitted (see [36] for a formal definition). Following standard terminology, sentences are closed formulas of the language with no free variables of any sort.

A basic action theory represents the conditions under which actions are executable, how the world state changes as a result of the actions that are possible, and what information the modeler has about the initial state. Typically such theories are used to support reasoning on "offline executions", where the agent "thinks" about the executability of action sequences and what conditions would hold in the resulting state, without actually executing any action in the real world. In this way, the agent can understand the consequences of acting before actually performing any action. Situation calculus action theories can also be used to support reasoning on "online executions" [10,16], where the agent reasons on the theory to understand which actions are possible and what they bring about, so as to select in an informed way one of them and actually perform it in the real world. We can think of an agent operating online as executing the following

[4] This assumption is in line with that of [29] and, as it will be clearer later on, enables the existence of a finite progression.

procedure using its basic action theory $\mathcal{D}$ and starting in the initial situation S_{now}[5]:

$S_{now} := S_0$;
while (`true`) **do** {
  $A := \{$ ground action $\alpha \mid \mathcal{D} \models Poss(\alpha, S_{now}) \}$;
  choose $\alpha \in A$;
  execute α;
  $S_{now} := do(\alpha, S_{now})$
}

In other words, at each iteration the agent selects one action among those that it *knows/believes* to be executable and executes it, and so on.

The following simple example illustrates the difference between online and offline executions.

Example 1. Consider a domain in which actions α and β are available, and are executable under the following precondition axioms:

$$Poss(\alpha, s) \equiv P(s), \qquad Poss(\beta, s) \equiv \neg P(s).$$

Further let us assume that the following successor state axiom holds for fluent F

$$F(do(a, s)) \equiv (a = \alpha \vee a = \beta) \vee F(s)$$

and that the initial situation description is empty, and hence we do not know whether P or $\neg P$ holds initially. Then by reasoning on offline executions we can infer that

$$\mathcal{D} \models \exists a.Poss(a, S_0) \wedge F(do(a, S_0)).$$

However, there are no online executions leading to F. Indeed the agent cannot infer P and thus doesn't know whether α is in fact executable and similarly it cannot infer $\neg P$ and thus doesn't know whether β is executable. So it cannot select α nor β. In other words, while *some* executable action exists, the agent does not know exactly *which* one can be executed, i.e., no action is *epistemically feasible* for the agent [38]. ■

[5]Reasoning on "online executions" also allow us to incorporate the results of sensing coming from the real world as we execute actions in the action theory [10,16,38]. In this paper we do not consider sensing, however.

In this paper we are interested in *verification of temporal properties over online executions*. In particular, the procedure above if executed for all possible action choices generates a sort of infinite tree representing all possible ways to execute actions online. We are interested in verifying temporal properties over such a tree.

Next we present a basic action theory that will be our running example.

Example 2. 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 introduce the following action theory where fluents and actions have the intuitive meaning.[6]
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:

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

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

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

Initial State Axioms:

- $IsRobot(r) \equiv r = R1 \vee r = R2$;

[6] We omit leading universal quantifiers for readability. Also, for simplicity, we use non-fluent predicates, e.g., $IsLoc$. To conform to the assumptions of the previous section, such predicates can be modeled by fluents whose successor state axioms preserve their truth value in all situations.

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

The successor state axiom for At says that item x is at location l after action a is performed in situation s if and only if either a involved robot r moving x to l and nothing was at l in s, and if x is either heavy or fragile then r handles this, or x was already at l in s, and a was not to ship it when it was at the shipping dock in s, nor was it for a robot r to move it to a different location l' where r can handle x. 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. With regards to item $I3$, there is incomplete information about its properties as heavy or fragile. Thus a *conformant* plan needs to be obtained for processing it. ■

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 $\mathcal{D}$ with respect to 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 with respect to how they describe the situation $do(\alpha, S_0)$ and the situations in the possible futures of $do(\alpha, S_0)$.

In a seminal paper, Lin and Reiter [29] gave a definition for the progression $\mathcal{D}_\alpha$ of $\mathcal{D}_0$ with respect to α and $\mathcal{D}$ as follows. 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;[7] and (ii) M and M' agree on all fluents at S_α, that is, for every relational fluent F, and every variable assignment μ, $M, \mu \models F(\vec{x}, S_\alpha)$ if and only if $M', \mu \models F(\vec{x}, S_\alpha)$. 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 $\mathcal{D}_0$ with respect to α and $\mathcal{D}$ if for any structure M, M is a model of $\mathcal{D}_\alpha$ if and only if there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$.

This definition 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 with respect to how they interpret the situations in S_α and the future of S_α. One technical detail is that according to this definition, some of the situation-independent properties of $\mathcal{D}$ are incorporated into the updated version of the initial knowledge base $\mathcal{D}_\alpha$. In particular $\mathcal{D}_{una} \cup \mathcal{D}_{coa}$ (which is already present in $\mathcal{D} - \mathcal{D}_0$) needs to be in $\mathcal{D}_\alpha$ in order to comply with the definition. We will see later how we can focus on the part of $\mathcal{D}_\alpha$ that does not include $\mathcal{D}_{una} \cup \mathcal{D}_{coa}$, in particular when this is finite and can be constructed by operating on $\mathcal{D}_0$.[8]

Now, observe that we can take progression as a way of characterizing 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 various models of a basic action theory can be seen as characterizing the possible actual states where an agent may be in, while the notion of belief state may be captured by *everything that is entailed* by the theory in a particular situation.[9] The *progressed* knowledge base $\mathcal{D}_\alpha$ is a sentence that essentially represents this.

It has been shown that in the very general case, progression, hence this form of belief states, can only be captured in second-order logic [29,50]. Nonetheless there are cases, such as the so-called *relatively complete* theories [29], for which a first-order progression can always be obtained (an analysis of all the known classes to date can be found in [51]). Next, we proceed to show that for bounded action theories it is also the case that a first-order progression can always be constructed.

---

[7] In our case we do not have situation-independent predicates and the only functions we consider are constants.

[8] A discussion on the need for $\mathcal{D}_{una}$ in $\mathcal{D}_\alpha$ and a slightly more involved definition that separates $\mathcal{D}_{una}$ from the progression can be found in [50, Definition 6, Appendix A].

[9] In fact, accounts in epistemic variants of the situation calculus have also been studied [25,42], but here we appeal to an interpretation directly based on entailment.

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, De Giacomo et al. [11] 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 it entails that: $\forall s. Executable(s) \supset Bounded_b(s)$. De Giacomo et al. [11] shows that for bounded action theories, verification of sophisticated temporal properties is decidable. It also identifies interesting classes of such theories.

Example 2 (continued). It is not difficult to show that the basic action theory we introduced in Example 2 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 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 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 that* $I3$ *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 theory does not entail that r can successfully move $I3$ to the shipping dock: if $I3$ is heavy, only $R1$ can

move it, while if 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 3. We adapt the theory of Example 2 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, *in 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 in 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 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. Consequently, *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 state that initially there is an item at station $Hold3$ whose identity is not known, and which is either fragile or heavy.

Example 4. Consider again Example 2 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:

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

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 a 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. Progressing Bounded Theories

We start by showing general results about progression. First we show that we can remove $\mathcal{D}_{una} \cup \mathcal{D}_{coa}$ from $\mathcal{D}$ when finding a progression for $\mathcal{D}_0$, and then add them back to get a correct progression with respect to the original theory.

LEMMA 1. *Let $\mathcal{D}^*$ be $\mathcal{D} - (\mathcal{D}_{una} \cup \mathcal{D}_{coa})$. If $\mathcal{D}^*_\alpha$ is a progression of $\mathcal{D}_0$ with respect to α and $\mathcal{D}^*$, then $\mathcal{D}^*_\alpha \cup \mathcal{D}_{una} \cup \mathcal{D}_{coa}$ is a progression of $\mathcal{D}$ with respect to α and $\mathcal{D}$.*

PROOF. Assume that the "if" part of the lemma holds, i.e., that $\mathcal{D}^*_\alpha$ is a progression of $\mathcal{D}_0$ with respect to α and $\mathcal{D}^*$. We will show that the "then" part of the lemma holds by considering the definition of progression for $\mathcal{D}^*_\alpha \cup \mathcal{D}_{una} \cup \mathcal{D}_{coa}$ and showing that for any structure M, M is a model of $\mathcal{D}^*_\alpha \cup \mathcal{D}_{una} \cup \mathcal{D}_{coa}$ if and only if there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$. ($\Leftarrow$): Let M be an arbitrary model, and an M' such that $M' \models \mathcal{D}$ and $M \sim_{S_\alpha} M'$. Since $\mathcal{D}_{una} \cup \mathcal{D}_{coa}$ is in $\mathcal{D}$ and mentions only equality atoms between terms of sort object and action, it follows by $M \sim_{S_\alpha} M'$ that also $M \models \mathcal{D}_{una} \cup \mathcal{D}_{coa}$. Now, since $\mathcal{D}^* \subset \mathcal{D}$, it follows that $M' \models \mathcal{D}^*$. Along with $M \sim_{S_\alpha} M'$, by the hypothesis it follows that $M \models \mathcal{D}^*_\alpha$. The other direction is similar. ∎

As we are interested in a knowledge base that remains finite as we progress, this lemma allows us to focus on the part of $\mathcal{D}_\alpha$ that can be maintained to be finite (note that $\mathcal{D}_{una} \cup \mathcal{D}_{coa}$ is infinite because $\mathcal{D}_{coa}$ is infinite), and then reason with $\mathcal{D}_\alpha$ under the assumption of uniqueness of names for actions and objects. In particular, Lemma 1 allows us to look into "composing" a progression from various parts of $\mathcal{D}_0$ that are progressed separately as shown in the next result:

LEMMA 2. *Let $\mathcal{D}$ be a basic action theory and $\mathcal{D}^*$ be $\mathcal{D} - (\mathcal{D}_{una} \cup \mathcal{D}_{coa})$. Let $Prog^*_{\mathcal{D},\alpha}(\phi)$ denote any progression of $\{\phi\}$ with respect to α and $(\mathcal{D}^* - \mathcal{D}_0) \cup \{\phi\}$.*[10] *The following holds:*

$$Prog^*_{\mathcal{D},\alpha}\left(\bigvee_i \varphi_i\right) \equiv \bigvee_i Prog^*_{\mathcal{D},\alpha}(\varphi_i),$$

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

PROOF. ($\Leftarrow$): Suppose not. Then there exists a model M such that, for some k, $M \models Prog^*_{\mathcal{D},\alpha}(\varphi_k)$ but $M \not\models Prog^*_{\mathcal{D},\alpha}(\bigvee_i \varphi_i)$. By the definition of progression, since $M \not\models Prog^*_{\mathcal{D},\alpha}(\bigvee_i \varphi_i)$, 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 $M \models Prog^*_{\mathcal{D},\alpha}(\varphi_k)$, 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

[10] A second-order specification of $Prog^*_{\mathcal{D},\alpha}(\phi)$ is always guaranteed to exist by results in [29].

$M'' \models (\mathcal{D}^* - \mathcal{D}_0) \cup \{\bigvee_i \varphi_i\}$ and $M \sim_{S_\alpha} M''$, hence we get a contradiction. The other direction is similar. ∎

Lemma 2 says that one can obtain a progression of a disjunctive $\mathcal{D}_0$ by progressing separately all of its disjuncts with respect to $\mathcal{D}^*$, and then adding the (infinite) set $\mathcal{D}_{una} \cup \mathcal{D}_{coa}$ by means of Lemma 1.

THEOREM 1. *Let $\mathcal{D}$ be a basic action theory with $\mathcal{D}_0$ of the form $\bigvee_i \varphi_i$, where φ_i are (possibly second-order) sentences uniform in S_0, α be a ground action, and $\mathcal{D}^*$, $Prog^*_{\mathcal{D},\alpha}(\phi)$ as in Lemma 2. Then the following is a progression of $\mathcal{D}_0$ with respect to α and $\mathcal{D}$: $\bigvee_i Prog^*_{\mathcal{D},\alpha}(\varphi_i) \cup \mathcal{D}_{una} \cup \mathcal{D}_{coa}$.*

Now we turn to bounded action theories. Firstly, we show that the initial situation description $\mathcal{D}_0$ of a bounded action theory $\mathcal{D}$ can be replaced by an equivalent disjunctive sentence uniform in S_0. To this end, consider a partition $\mathcal{M}_0 = \{\mathcal{M}_0^1, \mathcal{M}_0^2, \ldots\}$ of the models of $\mathcal{D}$, such that each cell $\mathcal{M}_0^i$ contains all models that agree, up to object renaming, on the intepretation of fluents at S_0 and of all constants. In other words, models in the same cell have isomorphic interpretations of constants and of fluents at S_0.

PROPOSITION 1. *For a bounded action theory, $\mathcal{M}_0$ is finite.*

PROOF. By the boundedness of $\mathcal{D}$ and the finiteness of $\mathcal{D}_0$ it follows that there exist, up to object renaming, only finitely many distinct interpretations of constants and fluents at S_0. Since, for each of such interpretations, $\mathcal{M}_0$ contains at most one cell, it is necessarily finite. ∎

For bounded action theories, the interpretation of fluents at S_0 can be captured (up to object renaming) by a *characteristic sentence*, i.e., a sentence of the form:

$$\exists w_1, \ldots, w_k. AllDist(w_1, \ldots, w_k) \wedge \bigwedge_{i=1}^{n} \forall \vec{x}_i.(F_i(\vec{x}_i, S_0) \equiv \phi_i(\vec{x}_i, w_1, \ldots, w_k)),$$

where: $AllDist(w_1, \ldots, w_k)$ is a formula of inequalities stating that $w_1, \ldots, w_k$ have distinct values, also distinct from any constant in $\mathcal{D}_0$; and $\phi_i(\vec{x}_i, w_1, \ldots, w_k)$ is a formula of the form $\bigvee(\vec{x}_i \circ \vec{t})$, with $\circ \in \{=, \neq\}$ and $\vec{t}$ containing only variables w_i and constants from $\mathcal{D}_0$,[11] that represents, up to object renaming, the extension of the set of fluents $\{F_1 \ldots F_n\}$ in the language at S_0, and the interpretation of constants occurring in such extensions. We observe that k is not the bound b. Indeed, b is a bound on the (maximum) number

[11] Note that each of the disjuncts $\vec{x}_i \circ \vec{t}$, e.g., $\vec{x}_i = \vec{t}$ is in fact a conjunction of the form $x_i^1 = t_1 \wedge \cdots \wedge x_i^m = t_m$, as $\vec{x}_i$ and $\vec{t}$ are vectors of variables and terms.

of tuples contained in the extension of the fluents, while k is a bound on the number of distinct elements that can occur in such extensions. Naturally, k can be derived from b, through the (maximum) arity of fluents, and vice versa. Note that characteristic sentences are FO and uniform in S_0, and that the sets of models of non-equivalent characteristic sentences are disjoint.

EXAMPLE 5. The characteristic sentence:

$$\exists w_1, w_2.(w_1 \neq w_2 \wedge w_1 \neq c \wedge w_2 \neq c) \wedge \forall x_1, x_2. \\ F(x_1, x_2, S_0) \equiv ((x_1 = w_1 \wedge x_2 = w_2) \vee (x_1 = c \wedge x_2 = w_1))$$

captures the models that agree, up to object renaming, on the interpretation of c and F at S_0. In particular, the sentence captures all models such that, for any three distinct objects o, o_1 and o_2, c is interpreted as o and the extension of F at S_0 contains exactly two pairs $\langle o_1, o_2\rangle$ and $\langle o, o_1\rangle$. ■

It can be checked that any two models of $\mathcal{D}$ are isomorphic at S_0 if and only if they satisfy equivalent characteristic sentences. Thus, each cell of $\mathcal{M}_0^i$ can be associated to a characteristic sentence Φ^i that fixes the fluent interpretations at S_0. For each of such sentences, by replacing $\mathcal{D}_0$ with $\{\Phi^i\}$ in $\mathcal{D}$, one obtains a characterization of $\mathcal{M}_0^i$.

PROPOSITION 2. *Let Φ^i be a characteristic sentence associated with $\mathcal{M}_0^i$. Then, $M \in \mathcal{M}_0^i$ if and only if $M \models (\mathcal{D} - \mathcal{D}_0) \cup \{\Phi^i\}$.*

PROOF. Consequence of the fact that $\mathcal{M}_0$ partitions the models of $\mathcal{D}$ and Φ^i captures the fluent interpretations at S_0. ■

By this result and the fact that $\mathcal{M}_0$ is *finite*, i.e., for some ℓ, $\mathcal{M}_0 = \{\mathcal{M}_0^1, \ldots, \mathcal{M}_0^\ell\}$, it turns out that M is a model of $\mathcal{D}$ if and only if it is a model of any one of the theories $(\mathcal{D} - \mathcal{D}_0) \cup \{\Phi^1\}, \ldots, (\mathcal{D} - \mathcal{D}_0) \cup \{\Phi^\ell\}$. Then, it is immediate to see that M is a model of $\mathcal{D}$ if and only if $M \models (\mathcal{D} - \mathcal{D}_0) \cup \{\bigvee_{i=1}^{\ell} \Phi^i\}$. By this, the next result easily follows, which states that the initial situation description $\mathcal{D}_0$ can be rewritten as $\{\bigvee_{i=1}^{\ell} \Phi^i\}$.

THEOREM 2. *Any bounded action theory $\mathcal{D}$ is logically equivalent to $(\mathcal{D} - \mathcal{D}_0) \cup \{\bigvee_{i=1}^{\ell} \Phi^i\}$, where Φ^i is a characteristic sentence of cell $\mathcal{M}_0^i$ of the partition $\mathcal{M}_0 = \{\mathcal{M}_0^1, \ldots, \mathcal{M}_0^\ell\}$ as defined above.*

This result shows the existence of a particular first-order sentence $\Phi_0 = \{\bigvee_{i=1}^{\ell} \Phi^i\}$ uniform in S_0, that can replace $\mathcal{D}_0$ in $\mathcal{D}$, but does not provide a constructive way to obtain Φ_0. The following result implies that, for a bounded action theory, such a sentence is in fact computable.

PROPOSITION 3. *For a bounded action theory $\mathcal{D}$, the characteristic sentences of the cells of $\mathcal{M}_0$ are computable.*

PROOF. Since $\mathcal{D}$ is bounded, one can compute a natural number B such that the fluent extensions in any executable situation, including S_0, contain at most B distinct values. Thus, the characteristic sentences associated with the cells of $\mathcal{M}_0$ use at most B distinct variables, and so we can take as candidate characteristic sentences those where at most B distinct variables occur, which are *finitely many* (once a suitable normal form is fixed).

By Theorem 2 and the fact that non-equivalent characteristic sentences have disjoint sets of models, it follows that a candidate characteristic sentence Φ is such that $(\mathcal{D} - \mathcal{D}_0) \cup \{\Phi\} \models \mathcal{D}$ if and only if it characterizes some cell $\mathcal{M}_0^i$. For bounded action theories, it can be checked that the entailment $(\mathcal{D} - \mathcal{D}_0) \cup \{\Phi\} \models \mathcal{D}$ holds if and only if $\{\Phi\} \cup \mathcal{D}_{coa} \models \mathcal{D}_0$ holds, that is, one can check whether $\Phi \models \mathcal{D}_0$ only on models enforcing $\mathcal{D}_{coa}$. Notice that any model of Φ has finite (in fact, bounded) fluent extensions. Based on this and the fact that Φ and $\mathcal{D}_0$ are essentially FO sentences, by exploiting results about making FO sentences *domain independent* (see, e.g., [28]), i.e., such that their satisfaction does not depend on the object domain of a model, one can show that checking whether a model M satisfying Φ also satisfies $\mathcal{D}_0$ is decidable.

Thus, a way to obtain the desired characteristic sentences is to take all the candidate sentences Φ such that $\Phi \models \mathcal{D}_0$. To this end, one can observe that since the models of each Φ have the same (finite) fluent extensions (up to renaming), they satisfy the same FO domain-independent sentences. Therefore, to check whether $\Phi \models \mathcal{D}_0$, one can take any model M (satisfying $\mathcal{D}_{coa}$) such that $M \models \Phi$, and check whether $M \models \mathcal{D}_0$. In our case, the fluent extensions of a model satisfying a characteristic sentence can be obtained from the characteristic sentence itself. ■

Next, we observe that the syntactic form of the characteristic sentence of each cell is essentially the same as that of *relatively complete initial knowledge bases with bounded unknowns*, defined in [51] (cf. Definition 3), for which a FO progression (expressed again as a relatively complete sentence with bounded unknowns) always exists (cf. Theorem 1 in [51]). Therefore, given $\mathcal{D}$, one can apply Theorem 2 to rewrite $\mathcal{D}_0$ as a disjunction of (finitely many) characteristic sentences, and then progress each of them separately, using Theorem 1, to compute a progression of $\mathcal{D}$. Note that, while progression applies to a single action, since the resulting progression is still a disjunction of characteristic sentences, we can apply it iteratively to deal with arbitrary (finite) action sequences.

So, with Theorem 1 we have shown that we can split the progression of a disjunctive $\mathcal{D}_0$ into the disjunction of separate progressions, and with

Theorem 2 that we can rewrite any $\mathcal{D}_0$ (of a bounded action theory) into an appropriate form of disjuncts, each of which we can progress separately using Theorem 1 in [51]. Theorem 3 below essentially shows how the main idea behind the progression mechanism in [51] can be extended to progress initial knowledge bases that are *disjunctions of* relatively complete initial knowledge bases with bounded unknowns.

THEOREM 3. *All bounded action theories are iteratively first-order progressable.*

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 4.

EXAMPLE 6. 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:

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

The second sentence, ψ_2, is the same as ψ_1 except for the last two conjuncts, in which item w is characterized as heavy instead of fragile:

$$\forall x(Heavy(x) \equiv x = I1 \vee x = w) \wedge \forall x(Fragile(y) \equiv y = I2).$$

Now consider action $move(R1, I1, ShipDock)$. Using the progression method of [51] for each characteristic sentence, we obtain a progressed version of the knowledge base, $\psi_1' \vee \psi_2'$, which is identical to $\psi_1 \vee \psi_2$, except that the location of $I1$ is updated in the specification of At. For ψ_1', we have:

$$\begin{aligned}
&\ldots \\
&\quad \forall x \forall l\{At(x, l, S_0) \equiv (x = I1 \wedge l = ShipDock) \vee \\
&\qquad\quad (x = I2 \wedge l = Hold2) \vee (x = I3 \wedge l = w)\} \wedge \\
&\ldots
\end{aligned}$$

As to ψ_2', it is obtained by replacing the specification of At in ψ_2 with the one above. ■

6. Verifying Online Executions

To express properties over online executions of BATs, we introduce a specific first-order variant of the μ-calculus [21,45]. The main characteristic of μ-calculus is its ability to express directly least and greatest fixpoints of (predicate-transformer) operators formed using formulae relating the current state to the next one. By using such fixpoint constructs one can easily express sophisticated properties defined by induction or co-induction. The μ-calculus is known to be one of the most powerful temporal logics, subsuming both linear time logics, such as LTL, and branching time logics such as CTL and CTL* [3].

Our variant of the μ-calculus, called $\mu\mathcal{LO}$, is able to express first-order properties *logically implied* in a situation (as opposed to expressing first-order properties *true* in a situation as in [11]). This is needed since online executions depend on what is logically implied, which, in a first-person view, corresponds to what the agent believes. To do so the "atomic" $\mu\mathcal{LO}$ formulas have the form $holds(\varphi)$ and express that the FO (closed) formula φ is logically implied by the BAT in the current situation. The syntax of $\mu\mathcal{LO}$ 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 *situation-suppressed* (i.e., with all situation arguments in fluents suppressed) situation calculus FO closed formula, whose constants must appear in $\mathcal{D} \setminus (\mathcal{D}_{una} \cup \mathcal{D}_{coa})$, 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 $\nu Z.\Phi = \neg\mu Z.\neg\Phi[Z/\neg Z]$. Intuitively, $\langle - \rangle\Phi$ holds in a situation if there exists an action (that is known to be executable) after which Φ holds, and $[-]\Phi$ holds in a situation if for all actions (that are known to be executable), Φ holds afterwards. As usual in the μ-calculus, formulae of the form $\mu Z.\Phi$ (and $\nu Z.\Phi$) must satisfy *syntactic monotonicity* of Φ with respect to 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 eventually 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. Note that our

$\mu\mathcal{LO}$ language does not allow for quantification across situations. However one can mitigate this limitation by using fluents to refer to objects across situations.

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 sets of ground 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[12]:

- $S_0 \in \Gamma$;
- If $\sigma \in \Gamma$, A is an action type with parameters $\vec{x}$, $\vec{n} \in \vec{\mathcal{N}}$ is a vector of names such that $|\vec{n}| = |\vec{x}|$, 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:

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

With a slight abuse of notation, given a closed situation-suppressed formula φ, 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 (with respect to 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 $\mu\mathcal{LO}$.

[12] Intuitively Γ is the set of situations forming the tree of online executions mentioned in Sect. 2.

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 7. For the agent in Example 2, one property of interest to verify is whether it is possible for the agent to eventually know that it has shipped all items that were in the factory. This can be expressed as the least fixpoint formula $\mu Z.\, holds(\neg\exists x \exists l.At(x,l) \vee \langle - \rangle Z)$ or, in CTL*, $EF\, holds(\neg\exists x \exists l.\, At(x,l))$.

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 2, $\mathcal{D}_2$, 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: $\mathcal{D}_2 \models AGEF\, holds(\neg\exists x \exists l.At(x,l))$. Another property that can be shown to hold for this 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: $\mathcal{D}_2 \models E(F\, holds(\neg\exists x \exists l.At(x,l)) \wedge G\, holds(\forall x(Shipped(x) \supset Painted(x))))$. ■

Example 8. For the agent in Example 3, with associated theory $\mathcal{D}_3$, we can show that: $\mathcal{D}_3 \models EF(holds(\forall l \neg\exists x.At(x,l)) \wedge F\, holds(\forall l.IsLoc(l) \supset \exists x.At(x,l)))$, 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 next: $\mathcal{D}_3 \models AG(holds(\exists x.At(x, ShipDock)) \supset \langle - \rangle holds(\neg\exists x.At(x, ShipDock)))$. However, this is not the case for other locations, e.g. $Hold1$, as it is possible for all locations to become occupied, at which point the agent must ship the item at the shipping dock before it can move the item at any other location: $\mathcal{D}_3 \models \neg AG(holds(\exists x.At(x, Hold1)) \supset \langle - \rangle holds(\neg\exists x.At(x, Hold1)))$. ■

We observe that because Γ and $\mathcal{N}$ (thus the object sort) are infinite, one cannot check whether $\mathcal{D} \models \Phi$ using an exhaustive search procedure, as typically done, e.g., in standard μ-calculus model checking [21]. This is true also for bounded theories, which can be infinite-state. However, under the assumption of boundedness, a finite structure can be constructed, which can be used to carry out the verification task. Such a construction is based on an abstraction process which amounts to clustering all the situations

whose corresponding states are isomorphic, so as to generate a finite-state transition system that can be used for the verification. The following result, proven in the rest of the section, is based on the construction of such a transition system.

THEOREM 4. *Let $\mathcal{D}$ be a BAT bounded by b and Φ a closed $\mu\mathcal{LO}$ formula. Then, checking whether $\mathcal{D} \models \Phi$ is decidable.*

The proof involves two steps. Firstly, we provide an alternative semantics for $\mu\mathcal{LO}$ formulas, equivalent to the one above. Such a semantics is defined on top of a transition system $T_{\mathcal{D}}$ derived from $\mathcal{D}$, which we call *progression-based* and that captures the evolution of the domain according to $\mathcal{D}$. The second step exploits our results about progression of bounded theories, to show that a finite-state transition system T_F can be effectively constructed, which is equivalent, for the purpose of verification, to $T_{\mathcal{D}}$. Since standard model checking algorithms can be executed on T_F, this implies that the verification of $\mu\mathcal{LO}$ formulas is decidable.

We start by introducing the notion of transition system (TS). An *(online-execution) transition system* is a tuple $T = \langle Q, q_0, \lambda, \rightarrow \rangle$, where:

- Q is the set of *possible states*;
- $q_0 \in Q$ is the *initial state*;
- $\lambda : Q \rightarrow 2^{\tilde{\mathcal{L}}}$ is the *labeling function*, associating each state q with a set $\mathcal{D}_q$ of situation-suppressed sentences over $\mathcal{N}$ ($\tilde{\mathcal{L}}$ denotes the set of situation calculus situation-suppressed sentences);
- $\rightarrow \subseteq Q \times Q$ is the *transition relation.*

As can be seen, this is a special case of standard labelled TS, where states are labelled by (possibly non first-order) logical theories. We call this class of TSs *online-execution* to stress that they can accommodate all the information relevant to online executions.

The semantics of a $\mu\mathcal{LO}$ formula Φ over a TS T, under a valuation $\mathcal{V}$, is as follows:

$$
\begin{array}{ll}
(\mathit{holds}(\varphi))^T_{\mathcal{V}} & = \{q \in Q \mid \lambda(q) \models \varphi\} \\
(\neg\Phi)^T_{\mathcal{V}} & = Q - (\Phi)^T_{\mathcal{V}} \\
(\Phi_1 \wedge \Phi_2)^T_{\mathcal{V}} & = (\Phi_1)^T_{\mathcal{V}} \cap (\Phi_2)^T_{\mathcal{V}} \\
(\langle - \rangle \Phi)^T_{\mathcal{V}} & = \{q \in Q \mid \exists q'. q \rightarrow q' \wedge q' \in (\Phi)^T_{\mathcal{V}}\} \\
(Z)^T_{\mathcal{V}} & = \mathcal{V}(Z) \\
(\mu Z.\Phi)^T_{\mathcal{V}} & = \bigcap \{\mathcal{E} \subseteq Q \mid (\Phi)^T_{\mathcal{V}[Z/\mathcal{E}]} \subseteq \mathcal{E}\}
\end{array}
$$

We say that T *entails* 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 [21], with satisfaction replaced by entailment on state labels.

Every action theory $\mathcal{D}$ induces a (family of) so-called *progression-based* TS $T_{\mathcal{D}} = \langle Q, q_0, \lambda, \rightarrow \rangle$, defined as follows:

- $Q = \Gamma$;
- $q_0 = S_0$;
- λ is inductively defined as follows:
  - $\lambda(q_0) = \tilde{\mathcal{D}_0} \cup \mathcal{D}_{coa} \cup \mathcal{D}_{una}$, 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 a *progression* of $(\mathcal{D} - \mathcal{D}_0) \cup \lambda(q)[S_0]$ with respect to $A(\vec{n})$, where $\lambda(q)[S_0]$ stands for the theory obtained from $\lambda(q)$ by re-introducing the situation argument and assigning it to S_0 in any fluent atom.[13]
- $\rightarrow \subseteq Q \times Q$ is the *transition relation* such that $q \rightarrow q'$ if and only if $q' = do(A(\vec{n}), q)$, for some action type A and names $\vec{n} \in \vec{\mathcal{N}}$.

Intuitively, $T_{\mathcal{D}}$ is the (infinite) situation tree of the theory with each situation labelled by a progression of $\mathcal{D}$ with respect to some ground action term, and taking the preceding situation as initial situation. As the following result shows, $T_{\mathcal{D}}$ retains all the information entailed by $\mathcal{D}$ at every situation, and can thus be used to interpret $\mu\mathcal{LO}$ formulae.

THEOREM 5. *Let $\mathcal{D}$ be a basic action theory (not necessarily bounded), and Φ a $\mu\mathcal{LO}$ formula. Then* $(\Phi)^{\mathcal{D}}_{\mathcal{V}} = (\Phi)^{T_{\mathcal{D}}}_{\mathcal{V}}$.

PROOF. By induction on the structure of Φ. For $\Phi = holds(\varphi)$, let $\sigma = do(\alpha_n, do(\ldots, do(\alpha_1, S_0) \cdots)) \in \Gamma$. By the definition of $T_{\mathcal{D}}$, the labeling of σ in $T_{\mathcal{D}}$, i.e., $\lambda(\sigma)$, is obtained by iteratively progressing $\mathcal{D}$ with respect to $\alpha_1, \ldots, \alpha_n$ (and then suppressing the situation term). Then, by definition of progression, $\mathcal{D} \models \varphi[\sigma]$ if and only if $\lambda(\sigma) \models \varphi$, that is, $\sigma \in (\Phi)^{\mathcal{D}}_{\mathcal{V}}$ if and only if $\sigma \in (\Phi)^{T_{\mathcal{D}}}_{\mathcal{V}}$. For $\Phi = \langle - \rangle \Phi'$, by the definition of $T_{\mathcal{D}}$, we have that for any $\sigma' \in \Gamma$, $\sigma' = do(\alpha, \sigma)$ for some α iff $\sigma \rightarrow \sigma'$ in $T_{\mathcal{D}}$. Then, because by the induction hypothesis, $\sigma' \in (\Phi')^{\mathcal{D}}_{\mathcal{V}}$ if and only if $\sigma' \in (\Phi')^{T_{\mathcal{D}}}_{\mathcal{V}}$, it follows that

[13] Notice that, in general, there exist many possible, logically equivalent, progressions, hence the family of TSs.

$\sigma \in (\langle - \rangle \Phi')^{\mathcal{D}}_{\mathcal{V}}$ if and only if $\sigma \in (\langle - \rangle \Phi')^{T_{\mathcal{D}}}_{\mathcal{V}}$. The cases of boolean connectives, $\Phi = Z$, and $\Phi = \mu Z.\Phi'$ are immediate. ∎

As a corollary, we have that: $\mathcal{D} \models \Phi$ if and only if $T_{\mathcal{D}} \models \Phi$. Thus, one can check whether $\mathcal{D} \models \Phi$ using $T_{\mathcal{D}}$ instead of $\mathcal{D}$. However, $T_{\mathcal{D}}$ is also infinite, thus this result alone does not help in showing decidability of the verification problem. To overcome this obstacle, we construct a further finite-state TS that is equivalent to $T_{\mathcal{D}}$ with respect to the verification of $\mu\mathcal{LO}$ formulas, and that we can use to effectively perform the check. To this end, we introduce the notions of *logical equivalence modulo renaming* between theories and *online-execution bisimulation* between TSs, and state relevant results about them.

Two theories $\mathcal{D}$ and $\mathcal{D}'$ over the same signature and names $\mathcal{N}$, are said to be *logically equivalent modulo renaming (of constants)*, written $\mathcal{D} \sim \mathcal{D}'$, if there exists a bijection $h : \mathcal{N} \to \mathcal{N}$ such that $\mathcal{D} \models h(\mathcal{D}')$ and $h(\mathcal{D}') \models \mathcal{D}$, where $h(\mathcal{D}')$ stands for the theory obtained from $\mathcal{D}'$ by replacing every constant n it mentions by $h(n)$. We say that h *preserves* a set of constants $C \subseteq \mathcal{N}$, if $h(n) = n$, for every $n \in C$. We write $\mathcal{D} \sim_C \mathcal{D}'$ to denote that $\mathcal{D}$ and $\mathcal{D}'$ are logically equivalent modulo renaming preserving C.

Logical equivalence modulo renaming formalizes the intuition that $\mathcal{D}$ and $\mathcal{D}'$ have exactly the same models, modulo renaming of constants. It can be seen that when two theories are logically equivalent modulo renaming, they satisfy the same closed formulas, modulo renaming of constants.

THEOREM 6. *Let $\mathcal{D}$ and $\mathcal{D}'$ be two action theories over the same signature and names $\mathcal{N}$, such that $\mathcal{D} \sim \mathcal{D}'$. Then, for any closed (situation calculus) formula φ, we have that $\mathcal{D} \models \varphi$ if and only if $\mathcal{D}' \models h(\varphi)$, for h a witness of $\mathcal{D} \sim \mathcal{D}'$, and $h(\varphi)$ the formula obtained from φ, by renaming each constant n as $h(n)$.*

PROOF. Immediate consequence of the definition of logical equivalence modulo renaming. ∎

In particular, this result implies that formulae mentioning only preserved constants can be left unchanged.

COROLLARY 1. *If $\mathcal{D} \sim_C \mathcal{D}'$, for $\mathcal{D}$ and $\mathcal{D}'$ action theories over the same signature and names $\mathcal{N}$, then, for any closed (situation calculus) formula φ mentioning only constants in C, we have that $\mathcal{D} \models \varphi$ if and only if $\mathcal{D}' \models \varphi$.*

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) C-preserving bisimulation* between T_1 and T_2 is a relation $B \subseteq Q_1 \times Q_2$ such that $B(q_1, q_2)$ implies:

- $\lambda_1(q_1) \sim_C \lambda_2(q_2)$;
- for every transition $q_1 \rightarrow_1 q_1'$, there exists a transition $q_2 \rightarrow_2 q_2'$, such that $\langle q_1', q_2' \rangle \in B$;
- for every transition $q_2 \rightarrow_2 q_2'$, there exists a transition $q_1 \rightarrow_1 q_1'$, such that $\langle q_1', q_2' \rangle \in B$.

T_1 and T_2 are said to be *(online-execution) bisimilar* with respect to C, written $T_1 \approx_C T_2$, if $\langle q_{10}, q_{20} \rangle \in B$, for some bisimulation B preserving C. As usual, bisimilarity is an equivalence relation.

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

THEOREM 7. *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 $T_1 \models \Phi$ if and only if $T_2 \models \Phi$.*

PROOF. The proof is by induction on the structure of Φ and essentially analogous to that of the bisimulation-invariance theorem for standard μ-calculus [21]. In fact, the only difference is in the case of atomic formulae ($\Phi = \varphi$) and the *next* operator ($\Phi = \langle - \rangle \Phi'$). For the former, the thesis is a direct consequence of Corollary 1 as, by the definition of online-execution bisimulation, $\lambda(q_{10}) \sim_C \lambda(q_{20})$. For the latter, we first observe that, by the definition of online-execution bisimulation, there exists a transition $q_{10} \rightarrow_1 q_1$ if and only if there exists a transition $q_{20} \rightarrow_2 q_2$ such that $\langle q_1, q_2 \rangle \in B$. Now, one can see that $q_1 \in (\Phi')_{\mathcal{V}}^{T_1}$ if and only if the TS $T_1' = \langle Q_1, q_1, \rightarrow_1, \lambda_1 \rangle$, obtained from T_1 by setting the initial state to q_1, is such that $T_1' \models \Phi'$, and analogously for q_2 and $T_2' = \langle Q_2, q_2, \rightarrow_2, \lambda_2 \rangle$. Also, it is immediate to see that because $T_1 \approx_C T_2$, we have that $T_1' \approx_C T_2'$. But then, by the induction hypothesis, $T_1' \models \Phi'$ if and only if $T_2' \models \Phi'$, and the thesis follows. ∎

Observe that this result holds even between an infinite-state TS, 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 by adapting standard μ-calculus model checking techniques, which essentially perform fixpoint computations on a finite state space. Unfortunately, two major obstacles prevent this approach from being

effective at this stage. Firstly, Theorem 7 applies only provided T_F is available while, thus far, we have no guarantee that it is actually computable.

Secondly, μ-calculus model checking requires a procedure to check whether state labelings of T_F, which are essentially FO theories, entail atomic subformulas of Φ, a problem that is, in general, undecidable.

For the former problem, we next describe a procedure for the construction of a finite-state TS T_F that we then prove to be online-execution bisimilar to $T_{\mathcal{D}}$. The latter problem can be easily overcome by resorting to Theorem 15 of [11], which states that the verification problem for bounded theories is decidable for a variant of $\mu\mathcal{LO}$ that covers the case that concerns us.

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

1: **let** C be the (finite) set of constants occurring in $\mathcal{D} \setminus (\mathcal{D}_{una} \cup \mathcal{D}_{coa})$;
2: **let** $U := \emptyset$; $F := \{u_0\}$; $\lambda(u_0) := \tilde{\mathcal{D}}_0 \cup \mathcal{D}_{coa} \cup \mathcal{D}_{una}$; $\rightarrow := \emptyset$;
3: **while** $F \neq \emptyset$ **do**
4: **pick** $u \in F$; **let** $F := F \setminus \{u\}$;
5: **for all** action types A with parameters $\vec{x}$ **do**
6: **let** $C_{\lambda(u)}$ be the set of constants occurring in $\lambda(u) \setminus (\mathcal{D}_{una} \cup \mathcal{D}_{coa})$;
7: **let** $O \subset \mathcal{N}$ be any set s.t. $|O| = |\vec{x}|$ and $O \cap (C \cup C_{\lambda(u)}) = \emptyset$;
8: **for all** assignments $v : \vec{x} \rightarrow O \cup C \cup C_{\lambda(u)}$ s.t. $\lambda(u) \models Poss(A(v(\vec{x})))$ **do**
9: **let** $\mathcal{D}_\alpha$ be the progression of $(\mathcal{D} - \mathcal{D}_0) \cup \lambda(u)$ with respect to $\alpha = A(v(\vec{x}))$;
10: **if** there exists $u' \in U$ s.t. $\mathcal{D}_\alpha \sim_C \lambda(u')$ **then**
11: **let** $\rightarrow := \rightarrow \cup \{u \rightarrow u'\}$;
12: **else**
13: **let** $U := U \uplus \{u'\}$, for u' a fresh state; $\lambda(u') = \mathcal{D}_\alpha$;
14: **let** $\rightarrow := \rightarrow \cup \{u \rightarrow u'\}$; $F := F \cup \{u'\}$;
15: **end if**
16: **end for**
17: **end for**
18: **end while**

We construct T_F using Algorithm 1. The algorithm takes a basic action theory $\mathcal{D}$ bounded by b as input and returns a finite-state TS $T_F = \langle U, u_0, \rightarrow, \lambda \rangle$ bisimilar to $T_{\mathcal{D}}$. T_F is obtained by iteratively progressing $\mathcal{D}$, starting from the initial situation and expanding the frontier states in F (lines 5–17). However, at each step, the theory is not progressed in all possible ways, that is with respect to all the infinitely many executable ground action terms; instead, only a finite subset of such terms is considered (see lines 6–7). Notice that by the boundedness assumption and Theorem 3, the progression of $\mathcal{D}_\alpha$ is computable ($\mathcal{D}_{una}$ and $\mathcal{D}_{coa}$ need not be explicitly represented but can be assumed, and reasoning can be performed under these assumptions), which is clearly a necessary condition for the algorithm to terminate (line 9

could not be completed otherwise). In addition, for the algorithm to be well-defined, it is required that testing the condition of the **if** statement (line 10) be decidable. This fact is a consequence of the next result, once observed that, $\mathcal{D}$ being b-bounded, so are the models of the labels of the states in U.

THEOREM 8. *Let $\mathcal{D}_{01}$ and $\mathcal{D}_{02}$ be finite sets of uniform situation-suppressed sentences over $\mathcal{N}$, and let $C \subseteq \mathcal{N}$ be a finite set of constants. If $\mathcal{D}_{01}$ and $\mathcal{D}_{02}$ are bounded by b, then checking whether $\mathcal{D}_{01} \sim_C \mathcal{D}_{02}$ is decidable.*

PROOF. Associate $\mathcal{D}_{0i}$ $(i = 1, 2)$ with the formula $\Phi_i = \mathit{Bounded}_b \wedge \exists \vec{x}.\mathit{AllDist}(\vec{x}).\mathcal{D}_{0i}[\vec{c}/\vec{x}]$, where $\vec{c}$ contains all the constants in $\mathcal{D}_{0i}$ not occurring in C, $\vec{x}$ is a fresh set of variables such that $|\vec{c}| = |\vec{x}|$, and, by slight abuse of notation, $\mathcal{D}_{0i}[\vec{c}/\vec{x}]$ stands for the conjunction of the formulas in $\mathcal{D}_{0i}$, with the constants in $\vec{c}$ syntactically replaced by the variables in $\vec{x}$. It can then be seen that checking whether $\mathcal{D}_{01} \sim_C \mathcal{D}_{02}$ is equivalent to checking whether Φ_1 and Φ_2 are logically equivalent (according to the standard definition). Notice that each Φ_i imposes only constraints on the objects occurring in the extension of some fluent, while it does not constrain the remaining objects. In particular, it leaves the object sort free. Moreover, such formulas constrain the fluent extensions to be bounded. As a result, to check whether $\Phi_1 \equiv \Phi_2$, it is sufficient checking whether the finite models of the two formulas such that the object sort contains only the values occurring in some fluent, match. But since such models are finite and, up to object renaming, finitely many, this is decidable. ■

Termination of the algorithm is guaranteed by the following result.

THEOREM 9. *If $\mathcal{D}$ is a BAT bounded by b, then Algorithm 1 terminates and produces a TS T_F with a finite number of states in U.*

PROOF. We observe that, under boundedness, there exist only finitely many equivalence classes of theories, with respect to logical equivalence modulo renaming. This holds, in particular, for theories containing situation-suppressed formulas only, such as those labeling the states of T_F. Termination is a consequence of this observation and the fact that, by the **if** statement, a new state u' (to expand) is added to F only if no other state u is present in U with a labelling that is logically equivalent modulo renaming to that of u'. ■

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

THEOREM 10. *If $\mathcal{D}$ is a BAT bounded by b, then $T_F \approx_C T_{\mathcal{D}}$, for C the set of constants occurring in $\mathcal{D} \setminus (\mathcal{D}_{una} \cup \mathcal{D}_{coa})$.*

PROOF. Define $B \subseteq Q \times U$ such that $\langle q, u\rangle \in B$ if and only if $\lambda(q) \sim_C \lambda(u)$. Obviously, $\langle q_0, u_0\rangle \in B$, as $\lambda(q_0) = \lambda(u_0)$, by the definition of $T_{\mathcal{D}}$ and the construction of T_F. We show that B is an online-execution bisimulation. To this end, consider a pair $\langle q, u\rangle \in B$ and observe that, by the definition of $T_{\mathcal{D}}$, a transition $q \to q'$ exists in $T_{\mathcal{D}}$ if and only if, for some ground action $A(\vec{n})$, $\lambda(q) \models \mathit{Poss}(A(\vec{n}))$ (with *Poss* situation-suppressed). Similarly, by the way T_F is constructed, a transition $u \to u'$ exists in T_F if and only if $\lambda(u) \models \mathit{Poss}(A(\vec{m}))$, for some ground action $A(\vec{m})$.

Assume $q \to q'$, thus $\lambda(q) \models \mathit{Poss}(A(\vec{n}))$, for some $A(\vec{n})$. We next prove that for any choice of O in Algorithm 1, there exists an action term $A(v(\vec{x}))$ such that $\lambda(u) \models \mathit{Poss}(A(v(\vec{x})))$. To see this, observe that, by definition of $\sim_C$, there exists a bijection h witnessing $\lambda(q) \sim_C \lambda(u)$. Assume first that h maps the objects of $\vec{n}$ into objects of the set $O \cup C \cup C_{\lambda(u)}$. In this case, the fact that $\lambda(u) \models \mathit{Poss}(A(v(\vec{x})))$ is implied by Theorem 6, for $v(\vec{x}) = h(\vec{n})$. Otherwise, there exists an element n of $\vec{n}$ such that $h(n) \notin O \cup C \cup C_{\lambda(u)}$. In this case, we modify h into a bijection h' analogous to h except that $h(n)$ is swapped with some value $h(n') \in O$ such that n' does not occur in $\vec{n}$. This is always possible because $|\vec{x}| = |O|$ and $O \cap (C \cup C_{\lambda(u)}) = \emptyset$. It can be checked that the so-obtained h' is also a witness of $\lambda(q) \sim_C \lambda(u)$. In addition, by the constraints on the choice of O, one can iterate these changes, to finally obtain an h' such that $h'(\vec{n}) \in O \cup C \cup C_{\lambda(u)}$, for which the previous case applies. Thus, there exists a transition $u \to u'$ in $T_{\mathcal{D}}$.

The fact that $\lambda(q') \sim_C \lambda(u')$ follows from the observation that when we progress theories that are logically equivalent modulo renaming, through the same ground actions (modulo the same renaming), we end up with progressed theories that are logically equivalent modulo the same renaming (it is easy to show that, otherwise, Theorem 6 would be violated). In particular, $\lambda(q')$ and $\lambda(u')$ are obtained as progressions of, respectively, $(\mathcal{D} - \mathcal{D}_0) \cup \lambda(q)$ and $(\mathcal{D} - \mathcal{D}_0) \cup \lambda(u)$, which are logically equivalent modulo renaming, as so are $\lambda(q)$ and $\lambda(u)$.

Proving the other requirement of the bisimulation relation is simpler, as any ground action term considered in the construction of T_F has a corresponding term in $T_{\mathcal{D}}$, and no surgery is required on h. ∎

Once T_F is obtained, a variant of the standard algorithm for μ-calculus model checking [21] can be applied to check whether $T_F \models \Phi$. This observation, together with Theorems 10 and 7, implies the desired result, i.e., Theorem 4.

7. Related Work

There has been growing interest in reasoning about and verifying agent programs. Most work on verification of agent systems/programs uses propositional modal logics and model checking techniques [3,32]. These include [6, 18,47], and [53], which all focus on model-checking of BDI programs. Model checking (and satisfiability) in these propositional modal logics is decidable. But such logics can only represent finite domains and finite state systems.

There is also some work that uses first-order logical formalisms and can deal with infinite domains. This includes work that uses theorem proving techniques, such as Shapiro et al.'s CASLve verification environment [43,44] for multi-agent ConGolog programs based on an extended version of the situation calculus with knowledge and goal fluents. Another approach first developed by [8] uses fixpoint approximation techniques reminiscent of model checking, in combination with "characteristic graphs", which can finitely represent a Golog program's configuration graph. De Giacomo et al. [15] and Sardina and De Giacomo [37] also use these techniques. But note that for these first-order formalisms, verification is undecidable in general. So these approaches have no termination guarantees and are hence sound but not complete.

First-order reasoning about action formalisms such as the situation calculus are very general and expressive. So until recently, decidability results for reasoning in the situation calculus had been few, e.g., [46] for an argumentless fluents fragment, and [23] for a description logic-like 2 variables fragment. It is the case that situation calculus basic action theories support regression to reduce reasoning about a given future situation to reasoning about the initial situation [36], and generalizations of this result such as just-in-time histories [17] can also be exploited. However, these techniques cannot be used to verify general temporal properties.

A significant advance was [11], where the class of bounded action theories in the situation calculus is identified for which verification of temporal properties is decidable. In such theories, the number of object tuples that belong to the extension of fluents is bounded in every situation. But the object domain remains infinite, and an infinite run may involve an unbounded number of objects. Claßen et al. [9] also identifies cases where verification of ConGolog programs is decidable. In both of these works, properties are verified over offline executions. In this paper, we essentially extend such approaches, in particular [11], to verify temporal properties over online executions. By the way, note that in [14] it is shown that if the initial situation description

is bounded, then one can verify using the techniques of [11] that an action theory remains bounded in all executable situations.

Verification of infinite states systems is of interest not only for AI, but also for other areas of computer science. There is also substantial work that uses model checking techniques on infinite state systems. However, in most of this work the emphasis is on studying recursive control rather than on a rich data oriented state description; typically data are either ignored or finitely abstracted, see e.g., [7]. There has recently been some attention paid in the field of business processes and services to including data into the analysis of processes [20,22,24]. Interestingly, while we have verification tools that are quite good for dealing with data and processes separately, when we consider them together, we get infinite-state transition systems, which resist classical model checking approaches to verification. Lately, there has been some work on developing verification techniques that can deal with such infinite-state processes [1,2,4,19]. In particular [2,4] brings forth the idea of exploiting state boundedness to get decidability for verification of infinite-state data-aware systems.

Note that in this paper, we take a first-person view of the action theory as representing the agent's beliefs, so the notion of belief is metatheoretic. There has also been work on versions of the situation calculus that incorporate an additional knowledge/belief modality, thus taking a third-person view of knowledge/belief. This can be done by adapting the possible worlds model of knowledge to the situation calculus, as first proposed by [34]. Scherl and Levesque [41,42] formalized this approach in the context of Reiter's basic action theories [36] and showed that regression could be used to answer epistemic queries about a given ground situation. Lakemeyer and Levesque [25,26] also developed a first-order modal version of the epistemic situation calculus.

The approach in this paper is related to but quite different from that in [12], which builds on a version of the situation calculus with a knowledge modality [41]. Such an approach is especially interesting when there are several agents working from their own first-person account simultaneously, and we can consider their relationship to a third-person (modeler) account [43]. However De Giacomo et al. [12] uses a notion of bounded epistemic action theory that is more restrictive than ours, in that it requires that the number of object tuples that the agent thinks may belong to any given fluent in a situation is bounded. In other words, the total number of tuples summed over all epistemic alternatives (in the situation), is required to be bounded. Here, instead, we only require that in any possible world (i.e., model of the

theory), the number of distinct tuples in the extension of any fluent in any given situation is bounded.

There is also a lot of work on progression. As mentioned in Sect. 5, the notion of progression for basic action theories was first introduced by Lin and Reiter [29]. They were also first to investigate restrictions that guarantee a first-order progression: a restriction on the form of $\mathcal{D}_0$, namely the relatively complete databases, and a restriction on the type of available actions, namely the *context-free* assumption for actions, each of which guarantees that a first-order progression can always be found. There is a lot of related work to this direction. Liu and Levesque [31] introduced the *local-effect* assumption for actions which was later shown by Vassos et al. [49] to be a sufficient condition for ensuring that a first-order progression can be found, while Liu and Lakemeyer [30] extended this further to the so-called normal actions. Among the most relevant to this work is the recent result about progressing a $\mathcal{D}_0$ that is relatively complete with bounded unknowns [51], where essentially $\mathcal{D}_0$ is similar to a database with named nulls and constraints on the values of those. In this work, Vassos and Patrizi [51] also give a classification of all known restrictions (or classes of basic action theories) for which a first-order progression can always effectively be computed.

Finally, note that in [13] the online executions verification for bounded action theories framework presented in this paper is adapted to handle sensing actions. There, a first-order variant of linear time logic [48] is used to specify the properties to be verified. It is also shown that one can always obtain a first-order progression for sensing actions.

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.[14] We have assumed that the object domain

[14]The assumption that an action can only occur in an online execution if the agent believes that it is executable is perhaps too strong for exogenous actions; in this case, it

is isomorphic to an infinite set of standard names. Since we are concerned with online executions, it would be strange to allow for the occurrence of actions that the agent cannot even name. But note that all the results we have shown hold even if we drop domain closure for objects.

In the case where the initial situation description is in the form studied in [51], computing progression becomes particularly easy. This simplifies verification by making it simple to compute the finite transition system on which the model checking algorithm is applied.

In future work, we want to extend our online executions verification framework to deal with 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 the support of: Ripartizione Diritto allo Studio, Università e Ricerca Scientifica of Provincia Autonoma di Bolzano–Alto Adige, under project VeriSynCoPateD (*Verification and Synthesis from Components of Processes that Manipulate Data*); EU Commission, under the IP project n. FP7-318338 Optique (*Scalable End-user Access to Big Data*); the Natural Sciences and Engineering Research Council of Canada under grant RGPIN-2015-03756 (Specification, Verification, and Synthesis of Autonomous Adaptive Agents); and EU FP7 project CIP-621137 VOICE (Virtual Open Incubation Ecosystem).

Footnote 14 continued
might be better to just require that the agent not believe that the action is not executable. But then the agent should come to know that the action was indeed executable when she observes it, and exogenous actions become analogous to sensing actions. Sensing is beyond the scope of this paper, so here we assume that an exogenous action can only occur when the agent believes that it can. We leave relaxing this assumption and handling the knowledge producing effects of exogenous actions for future work.

References

[1] Bagheri Hariri, B., D. Calvanese, G. De Giacomo, R. De Masellis, and P. Felli, Foundations of relational artifacts verification, in *Proceedings of BPM*, 2011.

[2] Bagheri Hariri, B., D. Calvanese, G. De Giacomo, A. Deutsch, and M. Montali, Verification of relational data-centric dynamic systems with external services, in *Proceedings of PODS*, 2013.

[3] Baier, C., J.-P. Katoen, and K. Guldstrand Larsen, *Principles of Model Checking*, MIT Press, Cambridge, 2008.

[4] Belardinelli, F., A. Lomuscio, and F. Patrizi, Verification of agent-based artifact systems, *Journal of Artificial Intelligence Research* 51:333–376, 2014.

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

[6] Bordini, R. H., M. Fisher, C. Pardavila, and M. Wooldridge, Model checking agentspeak, in *Proceedings of AAMAS*, 2003.

[7] Burkart, O., D. Caucal, F. Moller, and B. Steffen, Verification of infinite structures, in *Handbook of Process Algebra*, Elsevier Science, Amsterdam, 2001.

[8] Classen, J., and G. Lakemeyer, A logic for non-terminating Golog programs, in *Proceedings of KR*, 2008.

[9] Classen, J., M. Liebenberg, G. Lakemeyer, and B. Zarriess, Exploring the boundaries of decidable verification of non-terminating Golog programs, in *Proceedings of AAAI*, 2014.

[10] De Giacomo, G., 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, Berlin, 2009.

[11] De Giacomo, G., Y. Lespérance, and F. Patrizi, Bounded situation calculus action theories and decidable verification, in *Proceedings of KR*, 2012.

[12] De Giacomo, G., Y. Lespérance, and F. Patrizi, Bounded epistemic situation calculus theories, in *Proceedings of IJCAI*, 2013.

[13] De Giacomo, G., Y. Lespérance, F. Patrizi, and S. Vassos, LTL verification of online executions with sensing in bounded situation calculus, in *Proceedings of ECAI*, 2014.

[14] De Giacomo, G., Y. Lespérance, F. Patrizi, and S. Vassos, Progression and verification of situation calculus agents with bounded beliefs, in *Proceedings of AAMAS*, 2014.

[15] De Giacomo, G., Y. Lespérance, and A. R. Pearce, Situation calculus based programs for representing and reasoning about game structures, in *Proceedings of KR*, 2010.

[16] De Giacomo G., and H. J. Levesque, An incremental interpreter for high-level programs with sensing, in *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*, Springer, Berlin, 1999.

[17] De Giacomo, G., and H. J. Levesque, Projection using regression and sensors, in *Proceedings of IJCAI*, 1999.

[18] Dennis, L. A., M. Fisher, M. P. Webster, and R. H. Bordini, Model checking agent programming languages, *Automated Software Engineering* 19(1):5–63, 2012.

[19] Deutsch, A., R. Hull, F. Patrizi, and V. Vianu, Automatic verification of data-centric business processes, in *Proceedings of ICDT*, 2009.

[20] Dumas, M., W. M. P. van der Aalst, and A. H. M. ter Hofstede, *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley, Hoboken, 2005.

[21] Emerson, E. A., Model checking and the Mu-calculus, in *Descriptive Complexity and Finite Models. Proceedings of a DIMACS Workshop*, 1996.

[22] Gerede, C. E., and J. Su, Specification and verification of artifact behaviors in business process models, in *Proceedings of ICSOC*, 2007.

[23] Gu, Y., and M. Soutchanski, Decidable reasoning in a modified situation calculus, in *Proceedings of IJCAI*, 2007.

[24] Hull, R., Artifact-centric business process models: Brief survey of research results and challenges, in *Proceedings of OTM 2008 Confederated International Conferences*, 2008.

[25] Lakemeyer, G., and H. J. Levesque, Situations, si! situation terms, no!, in *Proceedings of KR*, 2004.

[26] Lakemeyer, G., and H. J. Levesque, Semantics for a useful fragment of the situation calculus, in *Proceedings of IJCAI*, 2005.

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

[28] Libkin, L., Embedded finite models and constraint databases, in *Finite Model Theory and Its Applications*, Springer, Heidelberg, 2007.

[29] Lin, F., and R. Reiter, How to progress a database, *Artificial Intelligence* 92(1–2):131–167, 1997.

[30] Liu, Y., and G. Lakemeyer, On first-order definability and computability of progression for local-effect actions and beyond, in *Proceedings of IJCAI*, 2009.

[31] Liu, Y., and H. J. Levesque, Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions, in *Proceedings of IJCAI*, 2005.

[32] Lomuscio, A., H. Qu, and F. Raimondi, MCMAS: A model checker for the verification of multi-agent systems, in *Proceedings of CAV*, 2009.

[33] McCarthy, J., and P. J. Hayes, Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence* 4:463–502, 1969.

[34] Moore, R. C., A formal theory of knowledge and action, in J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Common Sense World*, Ablex Publishing, Norwood, NJ, 1985, pp. 319–358.

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

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

[37] Sardina, S., and G. De Giacomo, Composition of ConGolog programs, in *Proceedings of IJCAI*, 2009.

[38] Sardina, S., G. De Giacomo, Y. Lespérance, and H. J. Levesque, On the semantics of deliberation in IndiGolog—from theory to implementation. *Annals of Mathematics and Artificial Intelligence* 41(2–4):259–299, 2004.

[39] Sardina, S., G. D. Giacomo, Y. Lespérance, and H. J. Levesque, On ability to autonomously execute agent programs with sensing, in *Proceedings of AAMAS*, 2004.

[40] Sardina, S., G. D. Giacomo, Y. Lespérance, and H. J. Levesque, On the limits of planning over belief states under strict uncertainty, in *Proceedings of KR*, 2006.

[41] Scherl, R. B., and H. J. Levesque, The frame problem and knowledge-producing actions, in *Proceedings of AAAI*, 1993.

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

[43] Shapiro, S., Y. Lespérance, and H. J. Levesque, The cognitive agents specification language and verification environment for multiagent systems, in *Proceedings of AAMAS*, 2002.

[44] Shapiro, S., Y. Lespérance, and H. J. Levesque, The cognitive agent specification language and verification environment, in M. Dastani, K. Hindriks, and J.-J. C. Meyer (eds.), *Specification and Verification of Multi-agent Systems/Programs*, Springer, Berlin, 2010, pp. 289–316.

[45] Stirling, C., *Modal and Temporal Properties of Processes*, Springer, Heidelberg, 2001.

[46] Ternovskaia, E., Automata theory for reasoning about actions, In *Proceedings of IJCAI*, 1999.

[47] van Riemsdijk, M., L. Atefnoaei, and F. de Boer, Using the Maude term rewriting language for agent development with formal foundations, in M. Dastani, K. V. Hindriks, and J.-J. C. Meyer (eds.), *Specification and Verification of Multi-agent Systems*. Springer, Berlin, 2010, pp. 255–287.

[48] Vardi, M. Y., An automata-theoretic approach to linear temporal logic, in *Proceedings of Banff Higher Order Workshop*, 1995.

[49] Vassos, S., G. Lakemeyer, and H. J. Levesque, First-order strong progression for local-effect basic action theories, in *Proceedings of KR*, 2008.

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

[51] Vassos, S., and F. Patrizi, A classification of first-order progressable action theories in situation calculus, in In *Proceedings of IJCAI*, 2013.

[52] Wooldridge, M., and A. Lomuscio, A computationally grounded logic of visibility, perception, and knowledge, *Logic Journal of the IGPL* 9(2):257–272, 2001.

[53] Yadav, N., and S. Sardina, Reasoning about BDI agent programs using ATL-like logics, in *Proceedings of JELIA*, 2012.

Progression and Verification of SitCalc Agents with Bounded Beliefs

G. De Giacomo, S. Vassos
Sapienza University of Rome
Rome, Italy

Y. Lespérance
York University
Toronto, ON, Canada

F. Patrizi
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy
patrizi@dis.uniroma1.it