Artificial Intelligence 237 (2016) 172–203

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Bounded situation calculus action theories

Giuseppe De Giacomo [a,*], Yves Lespérance [b,*], Fabio Patrizi [c,**]

[a] Dipartimento di Ingegneria informatica, automatica e gestionale, Sapienza Università di Roma, Italy
[b] Department of Electrical Engineering and Computer Science, York University, Toronto, ON, Canada
[c] KRDB Research Centre – Faculty of Computer Science, Free University of Bozen–Bolzano, Italy

ARTICLE INFO

Article history:
Received 7 June 2015
Received in revised form 7 April 2016
Accepted 20 April 2016
Available online 3 May 2016

Keywords:
Knowledge representation
Reasoning about action
Situation calculus
Verification

ABSTRACT

In this paper,[1] we investigate bounded action theories in the situation calculus. A bounded action theory is one which entails that, in every situation, the number of object tuples in the extension of fluents is bounded by a given constant, although such extensions are in general different across the infinitely many situations. We argue that such theories are common in applications, either because facts do not persist indefinitely or because the agent eventually forgets some facts, as new ones are learned. We discuss various classes of bounded action theories. Then we show that verification of a powerful first-order variant of the μ-calculus is decidable for such theories. Notably, this variant supports a controlled form of quantification across situations. We also show that through verification, we can actually check whether an arbitrary action theory maintains boundedness.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

The situation calculus [65,75] is a well-known first-order formalism with certain second-order features for representing dynamically changing worlds. It has proved to be an invaluable formal tool for understanding the subtle issues involved in reasoning about action. Its comprehensiveness allows us to place all aspects of dynamic systems in perspective. Basic action theories let us capture change as a result of actions in the system [73], while high-level languages such as Golog [58] and ConGolog [26] support the representation of processes over the dynamic system. Aspects such as time [74], knowledge and sensing [79], probabilities and utilities [14], and preferences [11], have all been addressed. The price of such a generality is that decidability results for reasoning in the situation calculus are rare, e.g., [86] for an argument-less fluents fragment, and [49] for a description logic-like two-variable fragment. Obviously, we have the major feature of being able to rely on regression to reduce reasoning about a given future situation to reasoning about the initial situation [75]. Generalizations of this basic result such as just-in-time histories [33] can also be exploited. However, when we move to temporal properties, virtually all approaches are based on assuming a finite domain and a finite number of states, and often rely on propositional modal logics and model checking techniques [6,63]. There are only few exceptions such as [21,32,82], which develop incomplete fixpoint approximation-based methods.

* Corresponding authors.
** Principal corresponding author.
E-mail addresses: degiacomo@dis.uniroma1.it (G. De Giacomo), lesperan@cse.yorku.ca (Y. Lespérance), patrizi@dis.uniroma1.it (F. Patrizi).

[1] A preliminary version of this paper appeared as [27].

http://dx.doi.org/10.1016/j.artint.2016.04.006
0004-3702/© 2016 Elsevier B.V. All rights reserved.

In this paper, we present an important new result on decidability of the situation calculus, showing that *verification of bounded action theories is decidable*. Bounded action theories are basic action theories [75], where it is entailed that in all situations, the number of object tuples that belong to the extension of any fluent is bounded. In such theories, the object domain remains nonetheless infinite and an infinite run may involve an infinite number of objects, though at every single situation the number of objects we predicate on is finite and, in fact, bounded.

But why should we believe that practical domains conform to this boundedness assumption? While it is often assumed that the law of inertia applies and that fluent atoms persist indefinitely in the absence of actions that affect them, we all know that pretty much everything eventually decays and changes. We may not even know how the change may happen, but nevertheless know that it will. Another line of argument for boundedness is epistemic. Agents remember facts that they use and periodically try to confirm them, often by sensing. A fact that never gets used is eventually forgotten. If a fact can never be confirmed, it may be given up as too uncertain. Given this, it seems plausible that in several contexts an agent's knowledge, in every single moment, can be assumed to be bounded. While these philosophical arguments are interesting and relate to some deep questions about knowledge representation, one may take a more pragmatic stance, and this is what we do here. We identify some interesting classes of bounded action theories and show how they can model typical example domains. We also show how we can transform arbitrary basic action theories into bounded action theories, either by blocking actions that would exceed the bound, or by having persistence (frame axioms) apply only for a finite number of steps. Moreover we show that we can effectively check whether any arbitrary theory with a bounded initial situation description remains bounded in all executable situations (to do so we need to use verification).

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

This work is of interest not only for AI, but also for other areas of computer science. In particular it is of great interest for the work on data-aware business processes and services [53,45,38]. Indeed while there are well-established results and tools to analyze business processes and services, without considering the data manipulated, when data are taken into account results are scarce. The present work complements that in, e.g., [37,4,9,5,10], and hints at an even more profound relevance of the situation calculus in those areas [64]. More generally, our results can be recast in other formalisms for reasoning about action, both in AI and in CS.

The rest of the paper is organized as follows. In Section 2, we briefly review the situation calculus and basic action theories. Then in Section 3, we define bounded action theories. Then, in Section 4, we discuss various ways of obtaining bounded action theories, while showing that many practical domains can be handled. In Section 5, we introduce the $\mu\mathcal{L}_p$ language that we use to express first-order temporal properties. After that, we show that verification of $\mu\mathcal{L}_p$ properties over bounded action theories is decidable, first in the case where we have complete information about the initial situation (Section 6), and then in the general incomplete information case (Section 7). In Section 8, we characterize the worst-case computational complexity of the problem as EXPTIME-complete. In Section 9, we give a technique based on our verification results, to check whether an arbitrary basic action theory maintains boundedness. In Section 10, we review the related literature. Finally, in Section 11, we conclude the paper by discussing future work topics.

2. Preliminaries

The *situation calculus* [65,75] 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., $Holding(x, s)$, meaning that the robot is holding object x in situation s). For simplicity, and without loss of generality, we assume that there are no functions other than constants and no non-fluent predicates. We denote fluents by F and the finite set of primitive fluents by $\mathcal{F}$. The arguments of fluents (apart from the last argument which is of sort situation) are assumed to be of sort object.

Within this language, one can formulate action theories that describe how the world changes as the result of the available actions. Here, we concentrate on *basic action theories* as proposed in [67,75]. We also assume that there is a *finite number of action types*. Moreover, we assume that there is a countably infinite set of object constants $\mathcal{N}$ for which the unique name

assumption holds. However, we do not assume domain closure for objects.[2] As a result, a basic action theory $\mathcal{D}$ is the union of the following disjoint sets of first-order (FO) and second-order (SO) axioms:

- $\mathcal{D}_0$: (FO) *initial situation description axioms* describing the initial configuration of the world (such a description may be complete or incomplete);
- $\mathcal{D}_{poss}$: (FO) *precondition axioms* of the form

  $$Poss(A(\vec{x}), s) \equiv \phi_A(\vec{x}, s),$$

  one per action type, stating the conditions $\phi_A(\vec{x}, s)$ under which an action $A(\vec{x})$ can be legally performed in situation s; these use a special predicate $Poss(a, s)$ meaning that action a is executable in situation s; $\phi_A(\vec{x}, s)$ is a formula of the situation calculus that is uniform in s. A *formula is uniform in* s if it mentions no other situation term but s and does not mention *Poss* (see [75] for a formal definition);
- $\mathcal{D}_{ssa}$: (FO) *successor state axioms* of the form

  $$F(\vec{x}, do(a, s)) \equiv \phi_F(\vec{x}, a, s),$$

  one per fluent, describing how the fluent changes when an action is performed; the right-hand side (RHS) $\phi_F(\vec{x}, a, s)$ is again a situation calculus formula uniform in s; successor state axioms encode the causal laws of the world being modeled; they take the place of the so-called effect axioms and provide a solution to the frame problem;
- $\mathcal{D}_{ca}$: (FO) unique name axioms for actions and (FO) domain closure on action types;
- $\mathcal{D}_{uno}$: (FO) unique name axioms for object constants in $\mathcal{N}$;
- Σ: (SO) foundational, domain independent, axioms of the situation calculus [67].

We say that a situation s is *executable*, written $Executable(s)$, if every action performed in reaching s was executable in the situation in which it occurred.

One of the key features of basic action theories is the existence of a sound and complete *regression mechanism* for answering queries about situations resulting from performing a sequence of actions [67,75]. In a nutshell, the regression operator $\mathcal{R}^*$ reduces a formula ϕ about a particular future situation to an equivalent formula $\mathcal{R}^*[\phi]$ about the initial situation S_0, by basically substituting fluent relations with the right-hand side formula of their successor state axioms. Here, we shall use a simple *one-step only* variant $\mathcal{R}$ of the standard regression operator $\mathcal{R}^*$ for basic action theories. Let $\phi(do(\alpha, s))$ be a formula uniform in the situation $do(\alpha, s)$. Then $\mathcal{R}[\phi(do(\alpha, s))]$ stands for the *one-step* regression of ϕ through the action term α, which is itself a formula uniform in s.

3. Bounded action theories

Let b be some natural number. We use the notation $|\{\vec{x} \mid \phi(\vec{x})\}| \geq b$, meaning that there exist at least b distinct tuples that satisfy ϕ, to stand for the following first-order logic (FOL) formula:

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

We also use the notation $|\{\vec{x} \mid \phi(\vec{x})\}| < b$, meaning that there are fewer than b distinct tuples that satisfy ϕ, to stand for: $\neg(|\{\vec{x} \mid \phi(\vec{x})\}| \geq b)$.

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

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

i.e., fluent F is bounded by b in situation s if there are fewer than b distinct tuples in the extension of F in situation s.

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

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

i.e., every fluent is bounded by b in situation s.

We say that an action theory $\mathcal{D}$ is *bounded* by b if every executable situation is bounded by b, formally:

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

Example 1. Consider a warehouse where items are moved around by a robot (a similar example is formalized in [31]). There are k storage locations where items can be stored. There is also a shipping dock where new items may arrive and stored items may be shipped out. We can axiomatize this domain as follows.

[2] Such an assumption is made in [27], where *standard names* [57] are used to denote objects. Thus, the results here generalize those in [27].

We have the following action precondition axioms[3]:

$$Poss(move(x,l,l'),s) \equiv At(x,l,s) \wedge IsLoc(l') \wedge \neg\exists y.At(y,l',s)$$

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

$$Poss(ship(x),s) \equiv At(x,ShipDock,s)$$

The first axiom says that, in situation s, the robot can perform action $move(x,l,l')$, i.e., move object x from location l to l', if and only if x is at l in s and l' is a location where no object is present in s. The second precondition axiom says that action $arrive(x)$ is executable in situation s, i.e., object x may arrive at the warehouse in s, if and only if the shipping dock is empty and x is not somewhere else in the warehouse. The last axiom says that object x can be shipped in situation s if it is at the shipping dock in s.

For the fluent At, we have the following successor state axiom:

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

where

$$\gamma^+(x,l,a,s) = \exists l'.a = move(x,l',l) \wedge At(x,l',s) \wedge IsLoc(l) \wedge \neg\exists y.At(y,l,s)$$
$$\vee\, a = arrive(x) \wedge l = ShipDock \text{ and}$$

$$\gamma^-(x,l,a,s) = \exists l'.a = move(x,l,l') \wedge l' \neq l \wedge IsLoc(l') \wedge \neg\exists y.At(y,l',s)$$
$$\vee\, a = ship(x) \wedge At(x,ShipDock,s)$$

This says that object x is at location l in the situation that results from doing action a in s if and only if $\gamma^+(x,l,a,s)$ holds or if x is already at l in s and $\gamma^-(x,l,a,s)$ does not hold. $\gamma^+(x,l,a,s)$ specifies the conditions under which action a makes object x be at location l in situation s, i.e., if a is to move x to a free location l from another location l' where x was in s, or a is x arriving and l is the shipping dock. $\gamma^-(x,l,a,s)$ specifies the conditions under which action a makes object x cease to be at location l in situation s, i.e., a is to move x to a different location that is free, or is to ship x.

We specify the initial situation with the following initial state axioms:

$$\forall x,l.\neg At(x,l,S_0)$$

$$IsLoc(l) \equiv l = ShipDock \vee l = SL_1 \vee \ldots \vee l = SL_k$$

We also have unique name axioms for the locations. For clarity, $IsLoc$ is a non-fluent predicate, although it is easy to recast it as a fluent that is unaffected by any action.

It is not difficult to show that this theory is bounded by k+1. First note that there are k+1 locations initially and the set of locations never changes, so $IsLoc$ is bounded by k+1. For fluent At, it is initially bounded by 0, but the $arrive$ action can augment its extension. However, the action theory ensures there can be at most one item at each of the k+1 locations. Thus At remains bounded by k+1. Therefore, the theory is bounded by k+1. Observe that, as there are infinitely many constants denoting distinct objects, effectively an unbounded number of items may be handled by subsequent $arrive$, $move$, and $ship$ actions. Despite this, the theory remains bounded. □

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

4. Obtaining bounded action theories

Before focusing on verification, in this section we look at various interesting sufficient conditions that guarantee that a basic action theory is bounded. Later, in Section 9, we will see that it is actually possible to use verification to check whether any arbitrary basic action theory, with a bounded initial situation description, is indeed bounded.

4.1. Bounding by blocking

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

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

$$Poss(a(\vec{x}),s) \equiv \Phi(\vec{x},s) \wedge \mathcal{R}[Bounded_b(do(a(\vec{x}),s))] \tag{1}$$

[3] Throughout this paper, we assume that all free variables in a formula are implicitly universally quantified from the outside. Occasionally, to be clear, we will write $\forall\varphi$ to denote the universal closure of φ explicitly.

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

Proof. By (1) it is guaranteed that any executable action leads to a bounded situation. Hence by induction on executable situations, we obtain the thesis. □

Example 2. Suppose that we have a camera on a smartphone or tablet computer. We could model the storage of photos on the device using a fluent $PhotoStored(p, s)$, meaning that photo p is stored in the device's memory (in situation s). Such a fluent might have the following successor state axiom:

$$PhotoStored(p, do(a, s)) \equiv a = takePhoto(p) \vee PhotoStored(p, s) \wedge a \neq deletePhoto(p)$$

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

$$Poss(takePhoto(p), s) \equiv True$$

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

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

$$Poss(takePhoto(p), s) \equiv \neg PhotoStored(p, s) \wedge |\{p' \mid PhotoStored(p', s)\}| < b - 1 \vee PhotoStored(p, s) \wedge |\{p' \mid PhotoStored(p', s)\}| < b$$

$$Poss(deletePhoto(p), s) \equiv PhotoStored(p, s) \wedge |\{p' \mid PhotoStored(p', s)\}| < b + 1.$$

Note how the condition on the right hand side of the first axiom above ensures there are fewer than b photos stored after the action of taking a photo p occurs. Clearly, the resulting theory is bounded by b (assuming that the original theory is bounded by b in S_0). □

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

Example 3. Let's extend the previous example by supposing that the device also maintains a contacts directory. We could model this using a fluent $InPhoneDir(name, number, photo, s)$, with the following successor state axiom:

$$InPhoneDir(na, no, p, do(a, s)) \equiv a = add(na, no, p) \vee InPhoneDir(na, no, p, s) \wedge a \neq deleteName(na) \wedge a \neq deleteNumber(no)$$

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

$$Poss(add(na, no, p), s) \equiv \neg\exists na', no', p'.InPhoneDir(na', no', p', s) \wedge (na' = na \vee no' = no) \wedge PhotoStored(p, s) \wedge |\{p' \mid PhotoStored(p', s)\}| < b \wedge |\{\langle na', no', p'\rangle \mid InPhoneDir(na', no', p', s)\}| < b - 1$$

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

We observe that this kind of bounded action theories are really modeling a capacity constraint on every fluent,[4] which may block actions from being executed. As a result, an action may be executable in a situation in the original theory, but not executable in the bounded one. Thus an agent may want to "plan" to find a sequence of actions that would make the action executable again. In general, to avoid dead-ends, one should carefully choose the original action theory on which the bound is imposed, in particular there should always be actions that remove tuples from fluents.

[4] The bound b applies to each fluent individually, so the total number of tuples in a situation is bounded by $|\mathcal{F}|b$. Instead, one could equivalently impose a global capacity bound on the total number of tuples for which some fluent holds in a situation.

4.2. Effect bounded action theories

Let's consider another sufficient condition for boundedness. Without loss of generality we can take the general form of successor state axioms to be as follows:

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

We say that fluent F is *effect bounded* if, for every action a and situation s:

$$|\{\vec{o} \mid \Phi_F^+(\vec{o}, a, s)\}| \leq |\{\vec{o'} \mid \Phi_F^-(\vec{o'}, a, s)\}|,$$

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

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

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

Proof. By induction on executable situations. □

Example 4. Many axiomatizations of the Blocks World are not effect bounded. For instance, suppose that we have fluents $OnTable(x, s)$, i.e., block x is on the table in situation s, and $On(x, y, s)$, i.e., block x is on block y in situation s, with the following successor state axioms:

$$OnTable(x, do(a, s)) \equiv a = moveToTable(x) \vee OnTable(x, s) \wedge \neg\exists y.a = move(x, y)$$

$$On(x, y, do(a, s)) \equiv a = move(x, y) \vee On(x, y, s) \wedge \neg\exists z.(z \neq y \wedge a = move(x, z)) \wedge a \neq moveToTable(x)$$

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

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

$$On(x, y, do(a, s)) \equiv a = move(x, y) \vee On(x, y, s) \wedge \neg\exists z.(z \neq y \wedge a = move(x, z))$$

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

$$Poss(move(x, y), s) \equiv x \neq Table \wedge x \neq y \wedge \neg\exists z.On(z, x, s) \wedge \exists z.(z \neq y \wedge On(x, z, s)) \wedge (y = Table \vee \neg\exists z.On(z, y, s))$$

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

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

$$Poss(replace(x, y), s) \equiv x \neq Table \wedge y \neq Table \wedge x \neq y \wedge \neg\exists z.On(z, x, s) \wedge \exists z.On(x, z, s) \wedge \neg\exists z.On(z, y, s) \wedge \neg\exists z.On(y, z, s)$$

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

$$On(x, y, do(a, s)) \equiv a = move(x, y) \vee \exists z.(a = replace(z, x) \wedge On(z, y, s)) \vee On(x, y, s) \wedge \neg\exists z.(z \neq y \wedge a = move(x, z)) \wedge \neg\exists z.(z \neq y \wedge a = replace(x, z)),$$

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

Example 5. For another simple example (perhaps more practical), let's look at how we could specify the "favorite web sites" menu of an Internet application. We can assume that there is a fixed number of favorite web sites positions on the menu, say 1 to k. We can replace what is at position n on the menu by the URL u by performing the action $replace(n, u)$. This can be axiomatized as follows:

$$FavoriteSites(n, u, do(a, s)) \equiv a = replace(n, u) \vee FavoriteSites(n, u, s) \wedge \neg\exists u'.(u' \neq u \wedge a = replace(n, u'))$$

$$Poss(replace(n, u), s) \equiv n \in [1..k] \wedge \exists u'.FavoriteSites(n, u', s)$$

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

The $FavoriteSites$ fluent is typical of many domain properties/relations, such as the passengers in a plane, the students in a class, or the cars parked in a parking lot, where we can think of the relation as having a finite capacity, and where we can reassign the objects that are in it. In some cases, the capacity bound may be difficult to pin down, e.g., the guests at a wedding, although the capacity is by no means unbounded. As well, there are definitely examples where we need an unbounded theory, e.g., to model a pushdown automata that can recognize a particular context-free language. The situation calculus is a very expressive language that accommodates this, for instance, it has been used to model Turing machines [75]. One might arguably want an unbounded "favorite sites" menu or contacts directory, although this seems hardly practical. Another interesting question is how such capacity constraints might apply to a complex agent such as a robot that is modeling its environment. Clearly, such a robot would have limitations with respect to how many environment features/objects/properties it can memorize and track. Finally, note that the condition $|\{\vec{o} \mid \Phi_F^+(\vec{o}, a, s)\}| \leq |\{\vec{o}' \mid \Phi_F^-(\vec{o}', a, s)\}|$ is not a FO formula and it is difficult (in fact, undecidable) in general to determine whether a basic action theory is effect bounded. But as our examples illustrate, there are many instances where it is easy to show that the bounded effects condition holds.

4.3. Fading fluents action theories

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

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

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

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

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

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

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

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

Example 6. Imagine a sort of "vacuum cleaner world" where a robotic vacuum cleaner may clean a room or region r [76]. If a room/region is used, then it becomes unclean. We could model this using a fluent $IsClean(r, s)$ with the following successor state axiom:

$$IsClean(r, do(a, s)) \equiv a = clean(r) \vee IsClean(r, s) \wedge \neg a = use(r)$$

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

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

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

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

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

Example 7. Consider a robot that can move objects around. We might model this using a fluent $At(objet, location, s)$ with the following successor state axiom:

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

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

Example 8. As a final example, consider a softbot that keeps track of which hosts are online. We might model this using a fluent $NonFaulty(host, s)$ with the following successor state axiom:

$$NonFaulty(h, do(a, s)) \equiv a = pingS(h) \vee NonFaulty(h, s) \wedge a \neq pingF(r)$$

Here the action $pingS(h)$ means that the host h has been pinged successfully, and the action $pingF(h)$ means that the host h has not responded to a pinging within the allocated time. As time passes, we may not want to assume that currently non-faulty hosts remain non-faulty. If we apply the proposed transformation to this specification, we obtain a theory where information about hosts being non-faulty fades. The agent must periodically ping the host successfully to maintain its knowledge that the host is non-faulty. Notice that, obviously, the theory does not provide a full account of the mechanisms that regulate the hosts' availability, but only the knowledge that the softbot has about the hosts. □

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

5. Expressing dynamic properties

To express properties about Situation Calculus action theories, we introduce a specific logic, inspired by the μ-calculus [40,15], one of the most powerful temporal logics, subsuming, in the propositional setting, both linear time logics, such as Linear Temporal Logic (LTL) [69] and Property-Specification Language (PSL) [39], and branching time logics, such as Computational Tree Logic CTL [20] and CTL* [41]. The main characteristic of the μ-calculus is its ability to express directly least and greatest fixpoints of (predicate-transformer) operators formed using formulas 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. This is the reason why virtually all logics used in verification can be considered as fragments of the μ-calculus. Technically, the μ-calculus separates local properties, asserted on the current state or on states that are immediate successors of the current one, from properties talking about states that are arbitrarily far away from the current one [15]. The latter are expressed through the use of fixpoints. Our variant of the μ-calculus is able to express first-order properties over situations while, at the same time, allowing for a controlled form of first-order quantification across situations, inspired by [5], where the quantification ranges over objects that persist in the extension of some fluents across situations.

Formally, we define the logic $\mu\mathcal{L}_p$ as:

$$\begin{aligned} \Phi ::= & \varphi \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \exists x.\textsc{live}(x) \wedge \Phi \mid \\ & \textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi \mid \textsc{live}(\vec{x}) \wedge [-]\Phi \mid Z \mid \mu Z.\Phi \end{aligned}$$

In addition, we use the usual FOL abbreviations for $\vee$, $\supset$, $\equiv$, and $\forall$, plus the standard μ-calculus abbreviation $\nu Z.\Phi = \neg\mu Z.\neg\Phi[Z/\neg Z]$, where we denote with $\Phi[Z/\neg Z]$ the formula obtained from Φ by substituting each occurrence of Z with $\neg Z$. Let us comment on some aspects of $\mu\mathcal{L}_p$:

- φ in the expression above is an arbitrary (possibly open) uniform *situation-suppressed* (i.e., with all situation arguments in fluents suppressed) situation calculus FO formula, in which the only constants that may appear are those explicitly mentioned in the situation calculus theory beyond $\mathcal{D}_{uno}$, i.e., those occurring in $\mathcal{D}_{poss} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_0$.[5] Observe that quantification inside φ is not subject to any restriction; in particular, $\textsc{live}(\cdot)$ is not required.
- Z is an SO (0-ary) predicate variable, denoting a set of situations (the SO assignment to Z is parameterized by the FO assignment to the individual variables, see later).
- $\mu Z.\Phi$ and $\nu Z.\Phi$ are *fixpoint formulas* which denote, respectively, the *least* and the *greatest fixpoint* of the formula Φ seen as a predicate transformer $\lambda Z.\Phi$ on sets of situations. To guarantee the existence of fixpoints, as usual in the μ-calculus, formulas 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.
- $\mu Z.\Phi$ and $\nu Z.\Phi$ may contain free individual variables, which are those of Φ; technically, these act as *parameters* of the fixpoint formula, i.e., the value of fixpoints $\mu Z.\Phi$ and $\nu Z.\Phi$ is determined only once an assignment to the free individual variables is given, see, e.g., [59] (Chap. 10).
- $\textsc{live}(x_1, \ldots, x_n)$ stands for $\bigwedge_{i \in \{1,\ldots,n\}} \textsc{live}(x_i)$. We assume that in $\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi$ and $\textsc{live}(\vec{x}) \wedge [-]\Phi$, the variables $\vec{x}$ are exactly the free individual variables of Φ, after we have substituted each bound predicate variable Z in Φ by the corresponding binding fixpoint formula $\mu Z.\Phi'$ or $\nu Z.\Phi'$. Observe that Z is not further substituted in Φ'.[6]
- The Boolean connectives have their usual meaning. Quantification over individuals in $\exists x.\textsc{live}(x) \wedge \Phi$ and $\forall x.\textsc{live}(x) \supset \Phi$ (i.e., $\neg \exists x.\textsc{live}(x) \wedge \neg \Phi$) has the expected meaning, with the proviso that individuals over which quantification ranges must belong to the active domain of the current situation, i.e., belong to the extension of some fluent in the current situation, as required by $\textsc{live}(\cdot)$.
- Intuitively, the use of $\textsc{live}(\cdot)$ in $\mu\mathcal{L}_p$ ensures that objects are only considered in quantification across situations if they persist along the system evolution, while the evaluation of a formula with objects that are not present in the current extension of the fluents trivially evaluates to either false for $\exists$ or true for $\forall$. In particular:
  - $\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi$ denotes the set of situations s such that for *some possible next situation* s', i.e., such that $s' = do(a, s)$ for some action a executable in s, we have that Φ holds in s', with the variables occurring free in Φ, $\vec{x}$, assigned to objects in the active domain of s.
  - $\textsc{live}(\vec{x}) \wedge [-]\Phi$ denotes the set of situations s such that in *all* possible next situations s', we have that Φ holds in s', with the variables occurring free in Φ assigned to objects in the active domain of s.
  - $\textsc{live}(\vec{x}) \supset \langle - \rangle \Phi$ (i.e., $\neg(\textsc{live}(\vec{x}) \wedge [-]\neg\Phi)$) denotes those situations s such that for *some* possible next situation s', we have that Φ holds in s', *as long as* the variables occurring free in Φ are assigned to objects in the active domain of s.
  - $\textsc{live}(\vec{x}) \supset [-]\Phi$ (i.e., $\neg(\textsc{live}(\vec{x}) \wedge \langle - \rangle \neg\Phi)$) denotes those situations s such that for *all* possible next situations s', we have that Φ holds in s', *as long as* the variables occurring free in Φ are assigned to objects in the active domain of s.

Next we turn to semantics. Since $\mu\mathcal{L}_p$ contains formulas with free individual and predicate variables, given a model $\mathcal{M}$ of an action theory $\mathcal{D}$ with object domain Δ and situation domain $\mathcal{S}$, we introduce a *valuation* (v, V) formed by an *individual variable valuation* v which maps each individual variable x to an object $v(x)$ in Δ, and a *parameterized predicate variable valuation* V, which, given the valuation of the individual variables v, maps each predicate variable Z to a subset $V(v, Z)$ of situations in $\mathcal{S}$ (notice that for each individual variable valuation v the mapping may change). Given a valuation (v, V), we denote by $(v, V)[x/d]$ the valuation (v', V') such that: *(i)* for every individual variable $y \neq x$, we have that $v'(y) = v(y)$ and $v'(x) = d$, *(ii)* for every predicate variable Z, we have that $V'(v', Z) = V(v', Z)$. Sometimes we also use the notation $v[\vec{x}/\vec{d}]$, to denote the valuation v' such that, for every individual variable y that is not a component of $\vec{x}$, we have $v'(y) = v(y)$, and $v'(x_i) = d_i$ for all components x_i of $\vec{x}$. Analogously, we denote by $(v, V)[Z/\mathcal{E}]$ the valuation (v', V') such that: *(i)* for every individual variable x, we have $v'(x) = v(x)$, *(ii)* for every predicate variable $Y \neq Z$, we have $V'(v', Y) = V(v, Y)$, and *(iii)* $V'(v', Z) = \mathcal{E}$. Also we denote by $adom^{\mathcal{M}}(s)$, the *active (object) domain* of situation s in the model $\mathcal{M}$, which is the set of all objects occurring in some $F^{\mathcal{M}}(s)$ ($F \in \mathcal{F}$) or as the denotation in $\mathcal{M}$ of a constant in the set C of object constants occurring in $\mathcal{D}_{poss} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_0$. Then, we assign semantics to formulas by associating to a model $\mathcal{M}$ and a valuation (v, V), an *extension function* $(\cdot)^{\mathcal{M}}_{(v,V)}$ which maps $\mu\mathcal{L}_p$ formulas to the inductively defined subsets of $\mathcal{S}$, as shown in Fig. 1 (for clarity, we interpret explicitly the abbreviation $\nu Z.\Phi$). Notice that, given a (possibly open) uniform

[5] Clearly, this assumption can be avoided by adding to the initial situation description, a new "dummy" fluent that holds for a bounded number of constants.

[6] More precisely, in $\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi$ and $\textsc{live}(\vec{x}) \wedge [-]\Phi$ we require that $\vec{x} = free(\Phi)$, where $free(\Phi)$ is inductively defined as follows:

$free(\varphi)$ are the free variables of the FO formula φ
$free(\neg\Phi) = free(\Phi)$
$free(\Phi_1 \wedge \Phi_2) = free(\Phi_1) \cup free(\Phi_2)$
$free(\exists x.\textsc{live}(x) \wedge \Phi) = free(\Phi) - \{x\}$
$free(\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi) = free(\Phi)$
$free(\textsc{live}(\vec{x}) \wedge [-]\Phi) = free(\Phi)$
$free(Z) = \begin{cases} \emptyset & \text{if } Z \text{ is unbound} \\ free(\mu Z.\Phi') & \text{if } Z \text{ is bound by } \mu Z.\Phi' \end{cases}$
$free(\mu Z.\Phi) = free(\Phi[Z/Z'])$ where $\Phi[Z/Z']$ is Φ with Z substituted by a fresh (unbound) predicate variable Z'.

$$
\begin{array}{ll}
(\varphi)^{\mathcal{M}}_{(v,V)} & = \{s \in \mathcal{S} \mid \mathcal{M}, v \models \varphi[s]\} \\
(\neg\Phi)^{\mathcal{M}}_{(v,V)} & = \mathcal{S} - (\Phi)^{\mathcal{M}}_{(v,V)} \\
(\Phi_1 \wedge \Phi_2)^{\mathcal{M}}_{(v,V)} & = (\Phi_1)^{\mathcal{M}}_{(v,V)} \cap (\Phi_2)^{\mathcal{M}}_{(v,V)} \\
(\exists x.\, \textsc{live}(x) \wedge \Phi)^{\mathcal{M}}_{(v,V)} & = \{s \in \mathcal{S} \mid \exists d \in adom^{\mathcal{M}}(s).\, s \in (\Phi)^{\mathcal{M}}_{(v,V)[x/d]}\} \\
(\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi)^{\mathcal{M}}_{(v,V)} & = \{s \in \mathcal{S} \mid \vec{x}/\vec{d} \in v \text{ and } \vec{d} \subseteq adom^{\mathcal{M}}(s) \text{ and} \\
 & \quad \exists a.\, (a,s) \in Poss^{\mathcal{M}} \text{ and } do^{\mathcal{M}}(a,s) \in (\Phi)^{\mathcal{M}}_{(v,V)}\} \\
(\textsc{live}(\vec{x}) \wedge [-] \Phi)^{\mathcal{M}}_{(v,V)} & = \{s \in \mathcal{S} \mid \vec{x}/\vec{d} \in v \text{ and } \vec{d} \subseteq adom^{\mathcal{M}}(s) \text{ and} \\
 & \quad \forall a.\, (a,s) \in Poss^{\mathcal{M}} \text{ implies } do^{\mathcal{M}}(a,s) \in (\Phi)^{\mathcal{M}}_{(v,V)}\} \\
(Z)^{\mathcal{M}}_{(v,V)} & = V(v,Z) \\
(\mu Z.\Phi)^{\mathcal{M}}_{(v,V)} & = \bigcap\{\mathcal{E} \subseteq \mathcal{S} \mid (\Phi)^{\mathcal{M}}_{(v,V)[Z/\mathcal{E}]} \subseteq \mathcal{E}\} \\
(\nu Z.\Phi)^{\mathcal{M}}_{(v,V)} & = \bigcup\{\mathcal{E} \subseteq \mathcal{S} \mid \mathcal{E} \subseteq (\Phi)^{\mathcal{M}}_{(v,V)[Z/\mathcal{E}]}\}
\end{array}
$$

Fig. 1. $\mu\mathcal{L}_p$ extension function $(\cdot)^{\mathcal{M}}_{(v,V)}$.

situation-suppressed situation calculus formula φ, by a slight abuse of notation, we denote by $\varphi[s]$ the corresponding formula with the situation calculus argument reintroduced and assigned to situation s.

Intuitively, the extension function $(\cdot)^{\mathcal{M}}_{(v,V)}$ assigns the following meaning to the $\mu\mathcal{L}_p$ constructs[7]:

- The extension of $\mu Z.\Phi$ is the *smallest subset* $\mathcal{E}_\mu$ of situations such that, assigning to Z the extension $\mathcal{E}_\mu$, the resulting extension of Φ is contained in $\mathcal{E}_\mu$ (with the assignments of the individual variables and the other predicate variables given by v and V, respectively). That is, the extension of $\mu Z.\Phi$ is the *least fixpoint* of the operator $(\Phi)^{\mathcal{M}}_{(v,V)[Z/\mathcal{E}]}$. Notice that for each valuation of the free individual variables in Φ, this operator is different: the free variables act as *parameters* of the predicate transformer $\lambda Z.\Phi$.
- Similarly, the extension of $\nu Z.\Phi$ is the *greatest subset* $\mathcal{E}_\nu$ of situations such that, assigning to Z the extension $\mathcal{E}_\nu$, the resulting extension of Φ contains $\mathcal{E}_\nu$. That is, the extension of $\nu Z.\Phi$ is the *greatest fixpoint* of the operator $(\Phi)^{\mathcal{M}}_{(v,V)[Z/\mathcal{E}]}$.

Observe that when a $\mu\mathcal{L}_p$ formula Φ is closed, its extension $(\Phi)^{\mathcal{M}}_{(v,V)}$ does not depend on the valuation (v, V). In fact, the only formulas of interest in verification are those that are closed. We say that a theory $\mathcal{D}$ *entails* a closed $\mu\mathcal{L}_p$ formula Φ, written $\mathcal{D} \models \Phi$, if, for every model $\mathcal{M}$ of $\mathcal{D}$, it is the case that $S_0^{\mathcal{M}} \in (\Phi)^{\mathcal{M}}_{(v,V)}$ (for any valuation (v, V), which is in fact irrelevant for closed formulas).

We can express arbitrary temporal/dynamic properties using least and greatest fixpoint constructions. For instance, to say that it is possible to reach a situation where (the closed formula) Φ holds, we use the least fixpoint formula $\mu Z.\Phi \vee \langle - \rangle Z$. This corresponds to a well-known CTL formula, namely $EF\Phi$ [20]. Instead, $\mu Z.\Phi \vee [-]Z$ expresses that, no matter which actions are executed, a situation where Φ holds is eventually reached. This corresponds to the CTL formula $AF\Phi$. Similarly, we can use the greatest fixpoint formula $\nu Z.\Phi \wedge [-]Z$ to express that Φ must hold in all possible reachable situations. Again, this corresponds to the well-known CTL formula $AG\Phi$. Instead, $\nu Z.\Phi \wedge \langle - \rangle Z$ expresses that Φ holds in the current situation and there exists always the possibility of moving to a next situation where Φ continues to hold. This corresponds to the CTL formula $EG\Phi$. For convenience we sometime use the CTL notation as abbreviation for the above fixpoint formulas.

Example 9. We show several examples of properties that we may want to verify for the warehouse robot domain of Example 1. First, suppose that we want to say that it is possible to eventually have shipped all items that are in the factory. This can be expressed in our language as a least fixpoint formula Φ_{eg9}[8]:

$$\mu Z.(\forall x, l.\neg At(x,l)) \vee \langle - \rangle Z$$

In the above, we rely on the fact that if there are no items left in the factory, then all items that were there must have been shipped. It is easy to check that the theory of Example 1, $\mathcal{D}_1$, entails that this formula holds in the initial situation S_0, formally $\mathcal{D}_1 \models \Phi_{eg9}$. In fact, we can also show that the above property always holds:

$$\mathcal{D}_1 \models \nu Z.\Phi_{eg9} \wedge [-]Z.$$

[7] By mentioning situations explicitly, it is also possible to define these operators directly in second-order logic as follows [34]:

$$
\begin{array}{rcl}
\mu Z.\Phi[s] & \equiv & \forall Z.(\forall \hat{s}.\Phi[\hat{s}] \supset Z(\hat{s})) \supset Z(s) \\
\nu Z.\Phi[s] & \equiv & \exists Z.(\forall \hat{s}.Z(\hat{s}) \supset \Phi[\hat{s}]) \wedge Z(s)
\end{array}
$$

Note that Φ may contain free individual and predicate variables, and indeed these remain free in $\mu Z.\Phi$ and $\nu Z.\Phi$. In this paper, we prefer to leave the situation implicit to allow for interpreting formulas over arbitrary transition systems, including finite ones, and hence relating our logic to standard μ-calculus.

[8] Note that more generally, a formula $\mu Z.\varphi \vee \langle - \rangle Z$, i.e., $EF\varphi$ in CTL, represents an instance of a planning problem, if we assume complete information on the initial situation; it is entailed by a theory if there exists an executable sequence of actions such that the goal φ holds afterward. If instead we have incomplete information, the formula only tells us that in each model there exists a sequence of actions, but such sequences may be different in different models.

The formula Φ_{eg9} corresponds to the CTL formula $EF(\forall x, l.\neg At(x,l))$, while the formula above corresponds to $AGEF(\forall x, l.\neg At(x,l))$.

A second property that we may want to verify is that it is possible to eventually have all items shipped out of the factory and then later to eventually have all locations filled with items. This can be expressed as follows:

$$\mathcal{D}_1 \models \mu X.[(\forall x, l.\neg At(x,l)) \wedge \mu Y.(\forall l.IsLoc(l) \supset \exists x.At(x,l)) \vee \langle - \rangle Y)] \vee \langle - \rangle X$$

or, equivalently, in CTL notation:

$$\mathcal{D}_1 \models EF[(\forall x, l.\neg At(x,l)) \wedge EF(\forall l.IsLoc(l) \supset \exists x.At(x,l))].$$

Our next example concerns a safety property; we can show that it is always the case that if an item is at the shipping dock it can be moved away or shipped out next:

$$\mathcal{D}_1 \models \nu Z.[\forall x.At(x, ShipDock) \supset \langle - \rangle \neg At(x, ShipDock)] \wedge [-]Z$$

Notice that x is quantified across situations, although $\textsc{live}(x)$ is not explicitly written, as implied by $At(x, ShipDock)$. In CTL notation we write (informally, EX stands for "there exists a successor such that"):

$$\mathcal{D}_1 \models AG[\forall x.At(x, ShipDock) \supset EX \neg At(x, ShipDock)].$$

However, this is not the case for other locations. Indeed, all locations, including the shipping dock, could become occupied, hence no movement between locations would be possible, except for shipping the item out from the shipping dock:

$$\mathcal{D}_1 \models \neg \nu Z.[\forall l.(\textsc{live}(l) \supset \forall x.(At(x,l) \supset (\textsc{live}(l) \supset \langle - \rangle \neg At(x,l)))] \wedge [-]Z$$

which simplifies to (also observing that $At(x,l)$ implies $\textsc{live}(x)$ and $\textsc{live}(l)$):

$$\mathcal{D}_1 \models \neg \nu Z.[\forall x, l.At(x,l) \supset \langle - \rangle At(x,l)] \wedge [-]Z.$$

But it is always possible to clear a location in two steps:

$$\mathcal{D}_1 \models \nu Z.[\forall l.(\exists x.At(x,l) \supset (\langle - \rangle(\textsc{live}(l) \wedge \langle - \rangle(\neg \exists x.At(x,l)))))] \wedge [-]Z$$

The above involves quantification across situations, and we require the location involved to persist (it trivially does).

Now, let's consider another example where we quantify across situations. We may want to say that it is always the case that if an item is in the warehouse, it is possible to have it persist until it is eventually shipped out (i.e., removed from all locations):

$$\mathcal{D}_1 \models \nu X.[\forall x.(\exists l.At(x,l)) \supset \mu Y.(\forall l.\neg At(x,l)) \vee \textsc{live}(x) \wedge \langle - \rangle Y] \wedge [-]X.$$

Note that the weaker property that it is always the case that if an item is in the warehouse, it is possible to have it shipped out eventually *if it persists* also holds:

$$\mathcal{D}_1 \models \nu X.[\forall x.(\exists l.At(x,l)) \supset \mu Y.(\forall l.\neg At(x,l)) \vee (\textsc{live}(x) \supset \langle - \rangle Y)] \wedge [-]X.$$

Finally, consider the property that if an item o is eventually shipped (i.e., $\forall l.\neg At(o,l)$ and hence o is not in the active domain) there is a future situation where o eventually comes back (i.e., it arrives back to the shipping dock and hence reappears in the active domain). While we cannot write this property in $\mu\mathcal{L}_p$ because o does not persist after it has been shipped, the property itself is trivially true. Indeed, if an object o disappears from the active domain, the theory does not predicate on o anymore. Hence the object o will appear back only if introduced by the action $arrive(o)$. Observe however that, since the theory is not predicating on o, if we can use o as the actual parameter of the action, then we can use as actual parameter any other object not in the active domain. That is, if $arrive(x)$ can be instantiated with some object outside the active domain (which is indeed the case) then it can be trivially instantiated with o. Analogously, the property that o eventually comes back no matter what actions are executed is trivially false. Indeed, if we can instantiate $arrive(x)$ with o, then we can instantiate it with every object outside the active domain. Thus, we can trivially have an infinite sequence of actions where $arrive(o)$ never occurs. This discussion hints at the essential aspect of persistence in quantification across: when objects persist we can express meaningful temporal properties over them, while when they do not, temporal properties trivialize, see Section 10. □

Observation 1. Observe that we do not have actions as parameters of $[-]\cdot$ and $\langle - \rangle \cdot$. However, we can easily remember the last action performed, and in fact a finite sequence of previous actions. To do this, for each action type $A(\vec{x})$, we introduce a fluent $Last_A(\vec{x}, s)$ with successor state axiom:

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

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

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

Similarly for the third last action, etc. Notice that each of these fluents has an extension in each situation containing at most one tuple of objects corresponding to the parameters $\vec{x}$ of the corresponding action A.

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

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

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

Observation 2. While $\mu\mathcal{L}_p$ allows for quantification over objects that persist across situations, the expressiveness of bounded action theories often makes its use avoidable. For instance, we can easily introduce a finite number of "registers", i.e., fluents that store only one tuple, which can be used to store and refer to tuples across situations. We can do this by introducing fluents $Reg_i(\vec{x}, s)$ and two actions $setReg_i(\vec{x})$ and $clearReg_i$ to set and clear the register Reg_i respectively. These are axiomatized as follows:

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

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

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

This formula says that there exists a sequence of actions where eventually the tuple referred to by register i has property F and there is an action after which it has property F'. Note also that this approach can be used to handle some cases of quantification over objects that do not persist across situations.

6. Verification of bounded action theories with complete information on S_0

We now show that verifying $\mu\mathcal{L}_p$ properties against bounded action theories is decidable. In this section we focus on action theories with complete information on the initial situation. The case of incomplete information is addressed in the next section. In particular, we assume that the extension of all fluents in the initial situation S_0 is given as a (bounded) database. We further assume that the domain of interpretation for objects Δ is also given. Notice that, as a consequence of the presence of infinitely many object constants and the unique name assumption on them $\mathcal{D}_{uno}$, Δ must be infinite.[9] As a result of these two assumptions, we have that the action theory $\mathcal{D}$ admits only one model $\mathcal{M}_\Delta$ [67] which, by a slight abuse of terminology, we call *the* model of the action theory $\mathcal{D}$ (though in order to define it we need Δ as well).

Our main result is the following.

Theorem 4. *Let $\mathcal{D}$ be a bounded action theory with initial situation described by a (bounded) database and with infinite object domain Δ, and let Φ be a closed $\mu\mathcal{L}_p$ formula. Then, checking whether $\mathcal{D} \models \Phi$ is decidable.*

The proof is structured as follows. Firstly, we show that action terms can be eliminated from $\mu\mathcal{L}_p$ formulas without loss of generality (cf. Section 6.1). Exploiting this, we show that only the fluent extensions in each situation, and not the situations themselves, are relevant when evaluating $\mu\mathcal{L}_p$ formulas (cf. Section 6.2). In this step, we also prove that checking FO formulas and answering FO queries *locally*, i.e., on a given situation, are, under boundedness, respectively decidable and effectively computable.

Then, based on the observations above, we introduce *transition systems* as alternative structures (to the models of situation calculus action theories), over which $\mu\mathcal{L}_p$ formulas can be evaluated. Transition systems are less rich than the models of situation calculus action theories, as they do not reflect, in general, the structure of the situation tree. Yet, they can accommodate the information of models needed to evaluate $\mu\mathcal{L}_p$ formulas (cf. Sections 6.3 and 6.4). In this step, we define the notion of *persistence-preserving bisimulation*, i.e., a variant of standard bisimulation which requires a certain kind of isomorphism to exist between bisimilar states and their successors (cf. page 187), and prove that persistence-preserving bisimilar transition systems preserve the truth-value of $\mu\mathcal{L}_p$ formulas (cf. Theorem 13). This is a key step in the proof,

[9] By the way in case of action theories with a given finite object domain, verification becomes easily reducible to model checking, since the corresponding situation calculus model is bisimilar to a finite propositional transition system.

which allows us to reduce the verification of $\mu\mathcal{L}_p$ formulas over an infinite transition system to that over a bisimilar transition system that is finite.

In the third and fundamental step (Section 6.5), we carry out a *faithful abstraction* operation, and show how to actually construct a finite transition system that is persistence-preserving bisimilar to the one, infinite, induced by *the* model of the action theory (cf. Procedure 1 and Theorems 15 and 16). Finally, we prove that verification is decidable on finite transition systems, thus on the one induced by the model of the action theory (cf. Theorem 17).

The rest of this section details these steps.

6.1. Suppressing action terms

Under uniqueness of action names, domain closure for actions, and the fact that action types are finitely many, we can remove, without loss of generality, action terms from uniform situation calculus formulas.

Theorem 5. *For every, possibly open, situation calculus FO formula $\varphi(\vec{x}, s)$ uniform in s and with free variables $\vec{x}$, all of object sort, there exists a situation calculus formula $\varphi'(\vec{x}, s)$ uniform in s, where no action terms occur, such that*

$$\mathcal{D}_{ca} \models \forall(\varphi(\vec{x}, s) \equiv \varphi'(\vec{x}, s)).$$

Proof. By induction on the structure of φ. If φ is $F(\vec{t}, s)$, we have that, by definition, $\vec{t}$ can only contain object terms, so φ' is $F(\vec{t}, s)$. If φ is $A(\vec{y}) = A'(\vec{y'})$, with $\vec{x} \subseteq \vec{y} \cup \vec{y'}$, if $A = A'$, then φ' is $\vec{y} = \vec{y'}$, else φ' is $\bot$. The case of Boolean connectives is straightforward. If φ is $\exists a.\phi(\vec{x}, a, s)$, consider the formula $\varphi'' = \bigvee_{A \in \mathcal{A}} \exists \vec{y}_A.\phi_A(\vec{x}, \vec{y}_A, s)$, with ϕ_A obtained from $\phi(\vec{x}, a, s)$, by replacing each occurrence of a with $A(\vec{y}_A)$, where $\vec{y}_A$ are fresh variables. We obviously have: $\mathcal{D}_{ca} \models \forall(\varphi \equiv \varphi'')$. Now, for each ϕ_A, let ϕ'_A be a formula containing no action terms, such that $\mathcal{D}_{ca} \models \forall(\phi_A \equiv \phi'_A)$. By induction hypothesis, such a ϕ'_A exists. Finally, let $\varphi' = \bigvee_{A \in \mathcal{A}} \exists \vec{y}_A.\phi'_A(\vec{x}, \vec{y}_A, s)$. Clearly, φ' contains no action terms and is uniform in s. By considering unique name axioms for actions and domain closure for action types ($\mathcal{D}_{ca}$), we can see that $\mathcal{D}_{ca} \models \forall(\varphi'' \equiv \varphi')$. Thus, since $\mathcal{D}_{ca} \models \forall(\varphi \equiv \varphi'')$, the thesis follows, i.e., $\mathcal{D}_{ca} \models \forall(\varphi \equiv \varphi')$. □

This result immediately extends to $\mu\mathcal{L}_p$, since in $\mu\mathcal{L}_p$ formulas, only uniform (situation suppressed) situation calculus FO subformulas can occur.

Theorem 6. *Any $\mu\mathcal{L}_p$ formula Φ can be rewritten as an equivalent $\mu\mathcal{L}_p$ formula Φ', where no action terms occur, such that $\mathcal{D}_{ca} \models \forall(\Phi \equiv \Phi')$.*

On the basis of this theorem, without loss of generality, we always rewrite $\mu\mathcal{L}_p$ formulas so that actions do not occur in them.

6.2. Suppressing situation terms

Since the FO components of $\mu\mathcal{L}_p$ formulas are situation-suppressed, situations are obviously irrelevant when checking $\mu\mathcal{L}_p$ formulas; more precisely, the FO components (thus the whole logic) are sensitive only to the interpretation of fluents (and constants) at each situation, while the situations themselves are not relevant. The impact of this observation on the evaluation of $\mu\mathcal{L}_p$ formulas in the general case will become evident in Section 6.4. Here, we focus on the *local* evaluation of FO components (on the interpretation of a single situation), or more specifically of FO situation calculus formulas uniform in s, and present some notable results that, besides being interesting *per se*, will be useful later on.

Given a basic action theory $\mathcal{D}$, we denote by $\mathcal{F}$ the set of its fluent symbols and by C the (finite) set of constants in $\mathcal{N}$ explicitly mentioned in $\mathcal{D}$, beyond $\mathcal{D}_{uno}$. Then, given a model $\mathcal{M}$ of $\mathcal{D}$ with object domain Δ and a situation s, it is natural to associate s with a FO interpretation $\mathcal{I}_{\mathcal{M}}(s) \doteq \langle \Delta, \cdot^{\mathcal{I}} \rangle$, where: *(i)* for every $c \in C$, $c^{\mathcal{I}} = c^{\mathcal{M}}$ and *(ii)* for every (situation-suppressed) fluent F of $\mathcal{D}$, $F^{\mathcal{I}} = \{\vec{d} \mid \langle \vec{d}, s \rangle \in F^{\mathcal{M}}\}$. The following result is an obvious consequence of the definitions above.

Theorem 7. *For any possibly open FO situation-suppressed situation calculus formula φ uniform in s, any situation s, and any object variable valuation v, we have that $\mathcal{M}, v \models \varphi[s]$ if and only if $\mathcal{I}_{\mathcal{M}}(s), v \models \varphi$.*

In other words, when evaluating a uniform FO situation-calculus formula on a situation, one needs only focus on the interpretation relative to the situation of interest.

Next, we show that, despite the object domain's infiniteness, for bounded action theories, we have decidability of FO formulas evaluation. Even more, we obtain that we can compute the answers to FO queries on specific situations. Notice that the latter result is not obvious, in that the object domain is infinite and, thus, so could be the answer. Importantly,

these results imply that we can check action executability and compute the effects of action executions, two facts that we will strongly leverage when checking $\mu\mathcal{L}_p$ formulas.

We start by showing some results concerning the decidability of FO formulas evaluation in an interpretation with finite predicate extensions, but infinite domain. Consider a finite set $\mathcal{F}$ of predicate symbols (situation-suppressed fluents) and a finite set C (a subset of $\mathcal{N}$) of constant symbols; then, a (FO) *interpretation* $\mathcal{I}$, over an infinite domain Δ, is a tuple $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$, where $\cdot^{\mathcal{I}}$ assigns an extension $F^{\mathcal{I}}$ over Δ to each predicate symbol $F \in \mathcal{F}$, and a distinct object $c^{\mathcal{I}} \in \Delta$ to every constant in C. The *active domain* of an interpretation $\mathcal{I}$, denoted $adom(\mathcal{I})$, is the set of all the individuals occurring in the extension of some fluent $F \in \mathcal{F}$, or interpreting some constant $c \in C$, in $\mathcal{I}$. Moreover, for simplicity, we assume that all the constants mentioned in FO formulas of interest belong to C.

First, let us recall a classical result saying that FO formulas (with no function symbols other than constants) can always be rewritten as formulas with quantified variables ranging only over the active domain of the interpretation. For an interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$, we define the *restriction of $\mathcal{I}$ to its active domain* as the interpretation $\tilde{\mathcal{I}} = \langle adom(\mathcal{I}), \cdot^{\mathcal{I}} \rangle$. In words, $\tilde{\mathcal{I}}$ is the same interpretation as $\mathcal{I}$, except that the object domain is replaced by the active domain. For technical convenience, given an interpretation $\mathcal{I}$, a FO formula φ, and a (FO) variable valuation v, we define the interpretation $\tilde{\mathcal{I}}^{\varphi,v}$ as the interpretation $\langle adom(\mathcal{I}) \cup D_{\varphi,v}, \cdot^{\mathcal{I}} \rangle$, where $D_{\varphi,v} \subseteq \Delta$ is the set containing all the objects from Δ that v assigns to the free variables of φ. Observe that $D_{\varphi,v}$ is always finite, since so is the set of φ's free variables.

Theorem 8 *(Theorem 5.6.3 of [60]). For every FO formula φ, one can effectively compute a formula φ', with quantified variables ranging only over the active domain, such that for any interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ with infinite domain Δ, and any valuation v, we have that $\mathcal{I}, v \models \varphi$ if and only if $\tilde{\mathcal{I}}^{\varphi,v}, v \models \varphi'$.*

This result essentially says that checking whether $\mathcal{I}, v \models \varphi'$ requires knowing only the interpretation function $\cdot^{\mathcal{I}}$ of $\mathcal{I}$, while the interpretation domain Δ can be disregarded. In other words φ' is a *domain-independent* formula [1]. One way to obtain domain-independent formulas is to avoid the use of negation and instead use logical difference with respect to the active domain. The above theorem says that it is always possible to transform a FO formula to be evaluated over an infinite domain to a domain-independent one to be evaluated over the active domain only, suitably augmented with the objects assigned to the free variables of the formula (and actually its proof gives an effective procedure to do so).

An immediate consequence of Theorem 8 is that if $adom(\mathcal{I})$ is finite, then checking whether $\mathcal{I}, v \models \varphi$ is decidable, no matter whether the interpretation domain of $\mathcal{I}$ is finite or infinite. Indeed, in the former case, decidability is obvious, while in the latter, one can simply check whether $\tilde{\mathcal{I}}^{\varphi,v}, v \models \varphi'$, which requires only lookups on the *finite* extensions of fluents and, in the presence of quantified variables, iterating over the *finitely many* elements of the active domain union the objects assigned by v to the free variables of φ. Thus, we have the following result.

Theorem 9. *Given a possibly open FO formula φ and an interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ with infinite Δ, if $adom(\mathcal{I})$ is finite, then, for any valuation v, checking whether $\mathcal{I}, v \models \varphi$ is decidable.*

Proof. See discussion above. □

Theorem 9 can be lifted to computing all the valuations v such that $\mathcal{I}, v \models \varphi$. Let φ be a FO formula with free variables $\vec{x}$, and $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ a FO interpretation. Then, the *answer on $\mathcal{I}$ to φ* is the relation $\varphi^{\mathcal{I}} \doteq \{\vec{d} \in \vec{\Delta} \mid \mathcal{I}, v \models \varphi, \text{ for } v(\vec{x}) = \vec{d}\}$. Sometimes, it is useful to fix the valuation of some variables $\vec{x}_{in} \subseteq \vec{x}$, say $v(\vec{x}_{in}) = \vec{d}_{in}$, and then consider the answer to φ under this partial assignment, that is, the relation $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}} \doteq \{\vec{d}_{out} \in \vec{\Delta} \mid \mathcal{I}, v \models \varphi, \text{ for } v(\vec{x}_{in}) = \vec{d}_{in} \text{ and } v(\vec{x} \setminus \vec{x}_{in}) = \vec{d}_{out}\}$.[10] The following theorem says that if $\mathcal{I}$ has an infinite domain Δ but a finite active domain, and the answer $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}}$ is finite, then the objects occurring in the answer come necessarily from either the active domain, or the values assigned to $\vec{x}_{in}$ by v.

Theorem 10. *Consider a FO formula φ with free variables $\vec{x}$. Let $\mathcal{I}$ be an interpretation with infinite Δ and finite active domain. If $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}}$ is finite, then $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}} \subseteq (adom(\mathcal{I}) \cup \vec{d}_{in})^n$, where $n = |\vec{x} \setminus \vec{x}_{in}|$.*

Proof. By contradiction. It can be easily proven that if $\mathcal{I}, v \models \varphi$, for $v(x_i) = d_i \notin (adom(\mathcal{I}) \cup \vec{d}_{in})$ and $x_i \in \vec{x} \setminus \vec{x}_{in}$, then for any other valuation $v' = v[x_i/d'_i]$ such that $d'_i \in \Delta \setminus (adom(\mathcal{I}) \cup \vec{d}_{in})$, we have that $\mathcal{I}, v' \models \varphi$. Since Δ is infinite and $adom(\mathcal{I})$ is finite, such d'_i are infinitely many, thus $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}}$ is infinite. Contradiction. □

In other words, any "new" object, with respect to those in $adom(\mathcal{I})$, occurring in the answer, must come from $\vec{d}_{in}$. A direct consequence of Theorems 9 and 10 is that one can actually compute the answer on $\mathcal{I}$ to φ.

[10] $\vec{x} \setminus \vec{x}_{in}$ denotes the tuple obtained from $\vec{x}$ by projecting out the components of $\vec{x}_{in}$.

Theorem 11. *Consider a FO formula φ with free variables $\vec{x}$. Let $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ be an interpretation with infinite Δ and finite active domain. If, for some valuation v such that $v(\vec{x}_{in}) = \vec{d}_{in}$, $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}}$ is finite, then $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}}$ is effectively computable.*

Proof. It suffices to record in $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}}$ all those tuples $\vec{d}_{out}$ such that for some v with $v(\vec{x}_{in}) = \vec{d}_{in}$ and $v(\vec{x} \setminus \vec{x}_{in}) = \vec{d}_{out}$, it is the case that $\mathcal{I}, v \models \varphi$. Since by Theorem 10 such $\vec{d}_{out}$ are finitely many and can be obtained using values from $adom(\mathcal{I}) \cup \vec{d}_{in}$, which is finite, and, by Theorem 9, checking whether $\mathcal{I}, v \models \varphi$ is decidable, it follows that $\varphi^{\mathcal{I}}_{\vec{x}_{in}/\vec{d}_{in}}$ is computable. □

These results find immediate application to the case of bounded action theories. Indeed, bounded action theories guarantee that $\mathcal{I}_{\mathcal{M}}(s)$, in Theorem 7, is finite (for s executable). Thus, by Theorem 9, for φ and v as above, we have that checking whether $\mathcal{I}_{\mathcal{M}}(s), v \models \varphi$ is decidable. A useful implication of this is that it is decidable to check whether an action $A^{\mathcal{M}}(\vec{o})$ is executable in a given situation s. Indeed, this requires checking whether $\mathcal{M}, v \models Poss(A(\vec{x}), s)$, with $v(\vec{x}) = \vec{o}$, which, by Theorem 7, is equivalent to $\mathcal{I}_{\mathcal{M}}(s), v \models \phi_A(\vec{x})$, with $\phi_A(\vec{x}, s)$ the RHS of the precondition axiom of A, which, in turn, is decidable. Moreover, Theorem 11 can be used to show that for a bounded action theory, the effects of executing an action at a given situation, as determined by the successor-state axioms, are computable and depend only on $\mathcal{I}_{\mathcal{M}}(s)$ (and the action). Indeed, we can exploit these results to get a sort of one-step regression theorem in our setting [67,75].

Theorem 12. *Let $\mathcal{M}$ be a model of a bounded action theory $\mathcal{D}$, s an executable situation, and $a = A^{\mathcal{M}}(\vec{o})$ an action, with action type $A(\vec{y})$. Then, for any fluent F, there exists a situation-suppressed action-term-free formula $\phi = \phi(\vec{x}, \vec{y})$ such that $F^{\mathcal{I}_{\mathcal{M}}(do^{\mathcal{M}}(a,s))} = \phi^{\mathcal{I}_{\mathcal{M}}(s)}_{\vec{y}/\vec{o}}$, and hence $F^{\mathcal{I}_{\mathcal{M}}(do^{\mathcal{M}}(a,s))}$ is effectively computable.*

Proof. Let $F(\vec{x}, do(a, s)) \equiv \phi_F(\vec{x}, a, s)$ be the successor-state axiom for fluent F. For the extension of F at situation $s' = do^{\mathcal{M}}(a, s)$, we have that $\langle \vec{p}, s' \rangle \in F$ iff $\mathcal{M}, v \models \phi_F(\vec{x}, A(\vec{y}), s)$, for some v such that $v(\vec{x}) = \vec{p}$ and $v(\vec{y}) = \vec{o}$. Notice that ϕ_F contains, in general, action and situation terms, and is uniform in s. However, by Theorem 5, it can be rewritten as an equivalent action-term-free formula $\phi^A_F(\vec{x}, \vec{y}, s)$. Then, by suppressing the situation argument, we obtain: $\vec{p} \in F^{\mathcal{I}_{\mathcal{M}}(s')}$ iff $\mathcal{I}_{\mathcal{M}}(s), v \models \phi^A_F(\vec{x}, \vec{y})$, for some v such that $v(\vec{x}) = \vec{p}$ and $v(\vec{y}) = \vec{o}$. That is, for $\phi = \phi^A_F$, $F^{\mathcal{I}_{\mathcal{M}}(s')} = \phi^{\mathcal{I}_{\mathcal{M}}(s)}_{\vec{y}/\vec{o}}$. Thus, since by boundedness of $\mathcal{D}$, $F^{\mathcal{I}_{\mathcal{M}}(s')}$ is finite, Theorem 11 implies the thesis. □

This result implies that, given $\mathcal{I}_{\mathcal{M}}(s)$ and an action $a = A^{\mathcal{M}}(\vec{o})$, we can obtain the interpretation of each F at $do^{\mathcal{M}}(a, s)$ by simply "querying" $\mathcal{I}_{\mathcal{M}}(s)$. Hence, by taking the same interpretation of constants as in $\mathcal{M}$, we can construct $\mathcal{I}_{\mathcal{M}}(do^{\mathcal{M}}(a, s))$, from $\mathcal{I}_{\mathcal{M}}(s)$ and the successor-state axioms of $\mathcal{D}$.

6.3. $\mu\mathcal{L}_p$ over transition systems

The results presented in Sections 6.1 and 6.2 suggest that, for the purpose of verification of $\mu\mathcal{L}_p$ formulas, one can operate on simpler structures than the models of situation calculus action theories. Indeed, as we saw, both actions and situations can be essentially disregarded. In this section, we introduce such simpler structures, namely *transition systems* (TS), show how $\mu\mathcal{L}_p$ formulas are evaluated over them, and present some important results that allow us to perform the verification on TSs instead of on the original model. The connection between models of situation calculus theories and transition systems will be discussed in Section 6.4. By Theorem 5, we can focus, without loss of generality, on a variant of $\mu\mathcal{L}_p$ where action terms do not occur.

By $Int^{\mathcal{F},C}_{\Delta}$, we denote the set of all possible interpretations of the situation suppressed fluents in $\mathcal{F}$ and constants in C, over the object domain Δ. A *transition system* (TS) (over the situation-suppressed fluents $\mathcal{F}$, constants C, and object domain Δ) is a tuple $T = \langle \Delta, Q, q_0, \rightarrow, \mathcal{I} \rangle$, where:

- Δ is an *object domain*;
- Q is a *set of states*;
- $q_0 \in Q$ is the *initial state*;
- $\rightarrow \subseteq Q \times Q$ is a *transition relation* between states; and
- $\mathcal{I} : Q \mapsto Int^{\mathcal{F},C}_{\Delta}$ is a *labeling function* associating each state q with an interpretation $\mathcal{I}(q) = \langle \Delta, \cdot^{\mathcal{I}(q)} \rangle$ such that the constants in C are interpreted in the same way in all the states over which $\mathcal{I}$ is defined.

To interpret a $\mu\mathcal{L}_p$ formula over a TS $T = \langle \Delta, Q, q_0, \rightarrow, \mathcal{I} \rangle$, we use valuations (v, V) formed by an individual variable valuation v and a parameterized predicate variable valuation V, as in Section 5. We define the *extension function* $(\cdot)^T_{(v,V)}$, which maps $\mu\mathcal{L}_p$ formulas to subsets of Q, as follows:

$$
\begin{aligned}
(\varphi)^T_{(v,V)} &= \{q \in Q \mid \mathcal{I}(q), v \models \varphi\}\\
(\neg\Phi)^T_{(v,V)} &= Q - (\Phi)^T_{(v,V)}\\
(\Phi_1 \wedge \Phi_2)^T_{(v,V)} &= (\Phi_1)^T_{(v,V)} \cap (\Phi_2)^T_{(v,V)}\\
(\exists x.\,\textsc{live}(x) \wedge \Phi)^T_{(v,V)} &= \{q \in Q \mid \exists d \in adom(\mathcal{I}(q)).\, q \in (\Phi)^T_{(v,V)[x/d]}\}\\
(\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi)^T_{(v,V)} &= \{q \in Q \mid \vec{x}/\vec{d} \in v \text{ and } \vec{d} \subseteq adom(\mathcal{I}(q)) \text{ and}\\
&\qquad \exists q'.\, q \to q' \text{ and } q' \in (\Phi)^T_{(v,V)}\}\\
(\textsc{live}(\vec{x}) \wedge [-] \Phi)^T_{(v,V)} &= \{q \in Q \mid \vec{x}/\vec{d} \in v \text{ and } \vec{d} \subseteq adom(\mathcal{I}(q)) \text{ and}\\
&\qquad \forall q'.\, q \to q' \text{ implies } q' \in (\Phi)^T_{(v,V)}\}\\
(Z)^T_{(v,V)} &= V(Z)\\
(\mu Z.\Phi)^T_{(v,V)} &= \bigcap \{\mathcal{E} \subseteq Q \mid (\Phi)^T_{(v,V)[Z/\mathcal{E}]} \subseteq \mathcal{E}\}
\end{aligned}
$$

Given a $\mu\mathcal{L}_p$ formula Φ, we say that a transition system T *satisfies* Φ *at state* q*, under* v *and* V, written $T, q, (v, V) \models \Phi$, if $q \in (\Phi)^T_{(v,V)}$. When Φ is closed on predicate variables, we omit V, as irrelevant, and write $T, q, v \models \Phi$. If Φ is closed on both individual and predicate variables we simply write $T, q \models \Phi$. For closed formulas, we say that T *satisfies* Φ, written $T \models \Phi$, if $T, q_0 \models \Phi$.

For our TSs we can prove a suitable version of the classical *bisimulation invariance result* for the μ-calculus, which states that bisimilar TSs satisfy exactly the same μ-calculus formulas, see e.g., [15]. Obviously, the notion of bisimulation needed here is not the classical one, but one that takes into account the FO interpretations labeling the states of the transition systems, as well as the controlled form of quantification across states allowed in $\mu\mathcal{L}_p$.

We first recall the standard notions of *isomorphism* and *isomorphic interpretations*. Two FO interpretations $\mathcal{I}_1 = \langle \Delta_1, \cdot^{\mathcal{I}_1} \rangle$ and $\mathcal{I}_2 = \langle \Delta_2, \cdot^{\mathcal{I}_2} \rangle$, over the same fluents $\mathcal{F}$ and constants C, are said to be *isomorphic*, written $\mathcal{I}_1 \sim \mathcal{I}_2$, if there exists a bijection (called *isomorphism*) $h : \Delta_1 \mapsto \Delta_2$ such that: *(i)* for every $F \in \mathcal{F}$, it is the case that $\vec{x} \in F^{\mathcal{I}_1}$ if and only if $h(\vec{x}) \in F^{\mathcal{I}_2}$; and *(ii)* for every $c \in C$, we have that $c^{\mathcal{I}_2} = h(c^{\mathcal{I}_1})$. It is immediate to see that if h is an isomorphism, then so is h^{-1}, and that $\sim$ is an equivalence relation. Intuitively, in order for two interpretations to be isomorphic, it is required that one can be obtained from the other by renaming the individuals in the interpretation domain. Notice that, necessarily, the interpretation domains of isomorphic interpretations have the same cardinality. When needed, to make it explicit that h is an isomorphism between $\mathcal{I}_1$ and $\mathcal{I}_2$, we write $\mathcal{I}_1 \sim_h \mathcal{I}_2$. We denote by $h|_{D_1}$ the *restriction* of h to D_1, i.e., the mapping $h|_{D_1} : D_1 \mapsto h(D_1)$, such that $h|_{D_1}(d) = h(d)$, for every $d \in D_1$. In addition, recall that $\tilde{\mathcal{I}} = \langle adom(\mathcal{I}), \cdot^{\mathcal{I}} \rangle$ denotes the *restriction of an interpretation* $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ to its active domain.

The bisimulation relation that captures $\mu\mathcal{L}_p$ can be defined as follows. Let $T_1 = \langle \Delta_1, Q_1, q_{10}, \to_1, \mathcal{I}_1 \rangle$ and $T_2 = \langle \Delta_2, Q_2, q_{20}, \to_2, \mathcal{I}_2 \rangle$ be two transition systems (over the situation-suppressed fluents and constants of an action theory $\mathcal{D}$), and let H be the set of all possible bijections $h : D_1 \mapsto D_2$, for $D_1 \subseteq \Delta_1$ and $D_2 \subseteq \Delta_2$. A relation $B \subseteq Q_1 \times H \times Q_2$ is a *persistence-preserving bisimulation* between T_1 and T_2, if $\langle q_1, h, q_2 \rangle \in B$ implies that:

1. $\tilde{\mathcal{I}}_1(q_1) \sim_h \tilde{\mathcal{I}}_2(q_2)$;
2. for each $q_1' \in Q_1$, if $q_1 \to_1 q_1'$ then there exists $q_2' \in Q_2$ such that:
   (a) $q_2 \to_2 q_2'$,
   (b) there exists a bijection $h' : adom(\mathcal{I}_1(q_1)) \cup adom(\mathcal{I}_1(q_1')) \mapsto adom(\mathcal{I}_2(q_2)) \cup adom(\mathcal{I}_2(q_2'))$ such that its restriction $h'|_{adom(\mathcal{I}_1(q_1))}$ coincides with h and its restriction $h'|_{adom(\mathcal{I}_1(q_1'))}$ is such that $\langle q_1', h'|_{adom(\mathcal{I}_1(q_1'))}, q_2' \rangle \in B$;
3. for each $q_2' \in Q_2$, if $q_2 \to_2 q_2'$ then there exists $q_1' \in Q_1$ such that:
   (a) $q_1 \to_2 q_1'$,
   (b) there exists a bijection $h' : adom(\mathcal{I}_1(q_1)) \cup adom(\mathcal{I}_1(q_1')) \mapsto adom(\mathcal{I}_2(q_2)) \cup adom(\mathcal{I}_2(q_2'))$ such that its restriction $h'|_{adom(\mathcal{I}_1(q_1))}$ coincides with h and its restriction $h'|_{adom(\mathcal{I}_1(q_1'))}$ is such that $\langle q_1', h'|_{adom(\mathcal{I}_1(q_1'))}, q_2' \rangle \in B$.

Notice that requirements 2b and 3b impose the existence of a bijection h' that preserves the bijection h (in fact, the isomorphism) between the objects in $adom(\mathcal{I}_1(q_1))$ and those in $adom(\mathcal{I}_2(q_2))$; this essentially means that the "identity" of such objects is preserved along the transition. Moreover, h' is required to induce an isomorphism between $adom(\mathcal{I}_1(q_1'))$ and $adom(\mathcal{I}_2(q_2'))$, when restricted to $adom(\mathcal{I}_1(q_1'))$, such that $\langle q_1', h'|_{adom(\mathcal{I}_1(q_1'))}, q_2' \rangle \in B$.

We say that a state $q_1 \in Q_1$ is *(persistence-preserving) bisimilar* to $q_2 \in Q_2$, written $q_1 \approx q_2$, if there exists a persistence-preserving bisimulation B between T_1 and T_2 such that $\langle q_1, h, q_2 \rangle \in B$, for some h; when needed, we also write $q_1 \approx_h q_2$, to explicitly name h. Finally, a transition system T_1 is said to be *persistence-preserving bisimilar* to T_2, written $T_1 \approx T_2$, if $q_{10} \approx q_{20}$. It is immediate to see that bisimilarity between states and transition systems, i.e., the (overloaded) relation $\approx$, is an equivalence relation.

Next, we prove a result (Theorem 13) saying that $\mu\mathcal{L}_p$ enjoys invariance under this notion of bisimulation. To this end, we first show the result for the simpler logic $\mathcal{L}_P$, obtained from $\mu\mathcal{L}_p$ by dropping the fixpoint construct. Namely, $\mathcal{L}_P$ is defined as:

$$\Phi ::= \varphi \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \exists x.\textsc{live}(x) \wedge \Phi \mid \textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi \mid \textsc{live}(\vec{x}) \wedge [-]\Phi$$

Such a logic corresponds to a first-order variant of the Hennessy–Milner Logic [50]. Note that its semantics is completely independent from the second-order valuation.

Given an individual variable valuation v we denote by $\textsc{im}(v)$ its image on the object domain.

Lemma 1. *Consider two transition systems $T_1 = \langle \Delta_1, Q_1, q_{10}, \rightarrow_1, \mathcal{I}_1 \rangle$ and $T_2 = \langle \Delta_2, Q_2, q_{20}, \rightarrow_2, \mathcal{I}_2 \rangle$, two states $q_1 \in Q_1$, $q_2 \in Q_2$, such that $q_1 \approx_h q_2$, and two individual variable valuations v_1 and v_2 mapping variables to Δ_1 and Δ_2, respectively. If there exists a bijection $\hat{h}$ between $adom(\mathcal{I}_1(q_1)) \cup \textsc{im}(v_1)$ and $adom(\mathcal{I}_2(q_2)) \cup \textsc{im}(v_2)$ whose restriction $\hat{h}|_{adom(\mathcal{I}_1(q_1))}$ coincides with h and such that for each individual variable x, $\hat{h}(v_1(x)) = v_2(x)$, then for every formula Φ of $\mathcal{L}_P$, possibly open on individual variables, we have that:*

$$T_1, q_1, v_1 \models \Phi \text{ if and only if } T_2, q_2, v_2 \models \Phi.$$

Proof. We proceed by induction on the structure of Φ. For $\Phi = \varphi$, we observe that, by Theorem 8, $\mathcal{I}_i(q_i), v_i \models \varphi$ if and only if $\tilde{\mathcal{I}}_i^{\varphi, v_i}(q_i), v_i \models \varphi'$ $(i = 1, 2)$, for φ' the rewriting of φ as its domain-independent version. Further, since $\tilde{\mathcal{I}}_1(q_1) \sim_h \tilde{\mathcal{I}}_2(q_2)$, and there is a bijection $\hat{h}$ between the objects assigned to variables by v_1 and v_2 (even if they are not in $adom(\mathcal{I}_1(q_1))$ or $adom(\mathcal{I}_2(q_2))$), it follows that $\tilde{\mathcal{I}}_1^{\varphi, v_1}(q_1) \sim \tilde{\mathcal{I}}_2^{\varphi, v_2}(q_2)$, thus, by the invariance of FOL wrt isomorphic interpretations, it follows that $\tilde{\mathcal{I}}_1^{\varphi, v_1}(q_1), v_1 \models \varphi'$ if and only if $\tilde{\mathcal{I}}_2^{\varphi, v_2}(q_2), v_2 \models \varphi'$. These two facts easily imply the thesis. The cases for Boolean connectives are obtained by straightforward induction using the same individual valuations v_1 and v_2 and the same bijection $\hat{h}$.

For $\Phi = \exists y.\textsc{live}(y) \wedge \Phi'$. Suppose that $T_1, q_1, v_1 \models \Phi$. Then, for some d_1, it is the case that $T_1, q_1, v_1[y/d_1] \models \textsc{live}(y) \wedge \Phi'$. Notice that this implies $d_1 \in adom(\mathcal{I}_1(q_1))$, then $\hat{h}(d_1) = h(d_1) = d_2$, for some $d_2 \in adom(\mathcal{I}_2(q_2))$, as $\hat{h}$ coincides with h on $adom(\mathcal{I}_1(q_1))$. Consider the individual valuation $v_2[y/d_2]$. For every variable x we have $\hat{h}(v_1[y/d_1](x)) = v_2[y/d_2](x)$ (for y we have $v_2[y/d_2](y) = d_2 = \hat{h}(d_1) = \hat{h}(v_1[y/d_1](y))$). Hence, using these new valuations and the same bijection $\hat{h}$, now restricted to $\textsc{im}(v_1[y/d_1])$ and $\textsc{im}(v_2[y/d_2])$ (to take into account the assignments to y), we can apply the induction hypothesis, and conclude that $T_2, q_2, v_2[y/d_2] \models \textsc{live}(y) \wedge \Phi'$, which implies $T_2, q_2, v_2 \models \Phi$. The other direction is proven symmetrically.

For $\Phi = \textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi'$. Suppose that $T_1, q_1, v_1 \models (\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi')$. By definition, this implies that $v_1(x_i) \in adom(\mathcal{I}_1(q_1))$ for each $x_i \in \vec{x}$, and there exists a transition $q_1 \rightarrow_1 q_1'$ such that $T_1, q_1', v_1 \models \Phi'$. Since $q_1 \approx_h q_2$, there exist: *(i)* a transition $q_2 \rightarrow_2 q_2'$, and *(ii)* a bijection $h' : adom(\mathcal{I}_1(q_1)) \cup adom(\mathcal{I}_1(q_1')) \mapsto adom(\mathcal{I}_2(q_2)) \cup adom(\mathcal{I}_2(q_2'))$ such that its restriction $h'|_{adom(\mathcal{I}_1(q_1))}$ coincides with h, its restriction $h'|_{adom(\mathcal{I}_1(q_1'))}$ is an isomorphism such that $\tilde{\mathcal{I}}_1(q_1') \sim_{h'|_{adom(\mathcal{I}_1(q_1'))}} \tilde{\mathcal{I}}_2(q_2')$, and $q_1' \approx_{h'|_{adom(\mathcal{I}_1(q_1'))}} q_2'$. Now consider two new variable valuations v_1' and v_2', defined as follows:

- for $x_i \in \vec{x}$ (for which we have that $v_1(x_i) \in adom(\mathcal{I}_1(q_1))$), let $v_1'(x_i) = v_1(x_i)$ and $v_2'(x_i) = v_2(x_i)$;
- choose $d_1 \in \Delta_1$ and, for all $y \notin \vec{x}$, let $v_1'(y) = d_1$, then: if $d_1 \in adom(\mathcal{I}_1(q_1)) \cup adom(\mathcal{I}_1(q_1'))$, for all $y \notin \vec{x}$, let $v_2'(y) = h'(d_1)$; else, choose $d_2 \notin adom(\mathcal{I}_2(q_2)) \cup adom(\mathcal{I}_2(q_2'))$, let, for all $y \notin \vec{x}$, $v_2'(y) = d_2$, and contextually extend h' so that $h'(d_1) = d_2$.

As a result, for all variables x, we have $h'(v_1'(x)) = v_2'(x)$ (for h' possibly extended as above). Consider the bijection $\hat{h}' = h'|_{adom(\mathcal{I}_1(q_1')) \cup \textsc{im}(v_1')}$. With this new bijection and the valuations v_1' and v_2', we can apply the induction hypothesis, and obtain that $T_1, q_1', v_1 \models \Phi'$ implies $T_2, q_2', v_2' \models \Phi'$, and since $q_2 \rightarrow_2 q_2'$, we have that $T_2, q_2, v_2' \models (\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi')$. Now, observe that the only free variables of $(\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi')$ are $x_i \in \vec{x}$, and that, for these, we have $v_1'(x_i) = v_1(x_i)$ and $v_2'(x_i) = v_2(x_i)$. Therefore, we can conclude that $T_2, q_2, v_2 \models (\textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi')$. The other direction can be proven in a symmetric way.

For $\Phi = \textsc{live}(\vec{x}) \wedge [-] \Phi''$: we observe that we can rewrite Φ as $\neg(\textsc{live}(\vec{x}) \supset \langle - \rangle \Phi')$, with $\Phi' = \neg \Phi''$. Then, assume that $T_1, q_1, v_1 \models (\textsc{live}(\vec{x}) \supset \langle - \rangle \Phi')$. By definition, this implies that: *(i)* either for some $x_i \in \vec{x}$ we have $v_1(x_i) \notin adom(\mathcal{I}_1(q_1))$; or *(ii)* for all $x_i \in \vec{x}$ we have $v_1(x_i) \in adom(\mathcal{I}_1(q_1))$ and there exists a transition $q_1 \rightarrow_1 q_1'$ such that $T_1, q_1', v_1 \models \Phi'$. We distinguish the two cases:

- If for some $x_i \in \vec{x}$, $v_1(x_i) \notin adom(\mathcal{I}_1(q_1))$, then we have that $v_2(x_i) \notin adom(\mathcal{I}_2(q_2))$. Indeed, assume toward contradiction that $v_2(x_i) \in adom(\mathcal{I}_2(q_2))$. Since $\tilde{\mathcal{I}}_1(q_1) \sim_h \tilde{\mathcal{I}}_2(q_2)$ it follows that the inverse h^{-1} of h is unique, hence $h^{-1}(v_2(x_i)) = v_1(x_i)$ and $v_1(x_i) \in adom(\mathcal{I}_1(q_1))$, getting a contradiction. Thus, we have that $T_2, q_2, v_2 \not\models \textsc{live}(\vec{x})$ and so $T_2, q_2, v_2 \models (\textsc{live}(\vec{x}) \supset \langle - \rangle \Phi')$.
- If for all $x_i \in \vec{x}$, $v_i(x_i) \in adom(\mathcal{I}_1(q_1))$, we can proceed in the same way as for the case of $\Phi = \textsc{live}(\vec{x}) \wedge \langle - \rangle \Phi'$.

The other direction is proven symmetrically. □

We can now extend the result to the whole $\mu\mathcal{L}_p$.

Lemma 2. *Consider two transition systems $T_1 = \langle \Delta_1, Q_1, q_{10}, \rightarrow_1, \mathcal{I}_1 \rangle$ and $T_2 = \langle \Delta_2, Q_2, q_{20}, \rightarrow_2, \mathcal{I}_2 \rangle$, two states $q_1 \in Q_1$, $q_2 \in Q_2$, such that $q_1 \approx_h q_2$, and two individual variable valuations v_1 and v_2 mapping variables to Δ_1 and Δ_2, respectively. If there exists a bijection $\hat{h}$ between $adom(\mathcal{I}_1(q_1)) \cup \text{IM}(v_1)$ and $adom(\mathcal{I}_2(q_2)) \cup \text{IM}(v_2)$ whose restriction $\hat{h}|_{adom(\mathcal{I}_1(q_1))}$ coincides with h and such that for each individual variable x, $\hat{h}(v_1(x)) = v_2(x)$, then for every formula Φ of $\mu\mathcal{L}_p$, closed on the predicate variables but possibly open on the individual variables, we have:*

$$T_1, q_1, v_1 \models \Phi \text{ if and only if } T_2, q_2, v_2 \models \Phi.$$

Proof. We prove the theorem in two steps. First, we show that Lemma 1 can be extended to the infinitary version of $\mathcal{L}_P$ that supports arbitrary infinite disjunction of formulas sharing the same free variables [88]. Then, we recall that fixpoints can be translated into this infinitary logic, thus guaranteeing invariance for the whole $\mu\mathcal{L}_p$ logic. Let Ψ be a possibly infinite set of open $\mathcal{L}_P$ formulas. Given a transition system $T = \langle \Delta, Q, q_0, \rightarrow, \mathcal{I} \rangle$, the semantics of $\bigvee \Psi$ is $(\bigvee \Psi)^T_{(v,V)} = \bigcup_{\psi \in \Psi} (\psi)^T_{(v,V)}$. Therefore, given a state q of T and a variable valuation v, we have $T, q, v \models \Psi$ if and only if $T, q, v \models \psi$ for some $\psi \in \Psi$. Arbitrary infinite conjunction is obtained for free through negation. Lemma 1 extends to this arbitrary infinite disjunction. By the induction hypothesis, under the assumption of the Lemma, we can assume that for every formula $\psi \in \Psi$, we have $T_1, q_{10}, v_1 \models \psi$ if and only if $T_2, q_{20}, v_2 \models \psi$. Given the semantics of $\bigvee \Psi$ above, this implies that $T_1, q_{10}, v_1 \models \bigvee \Psi$ if and only if $T_2, q_{20}, v_2 \models \bigvee \Psi$.

In order to extend the result to the whole $\mu\mathcal{L}_p$, we translate μ-calculus *approximates* into the infinitary $\mathcal{L}_P$ by (see [15, 88]), where the approximant of index α is denoted by $\mu^\alpha Z.\Phi$ for least fixpoint formulas $\mu Z.\Phi$ and $\nu^\alpha Z.\Phi$ for greatest fixpoint formulas $\nu Z.\Phi$. This is a standard result that holds also for $\mu\mathcal{L}_p$. In particular, such approximates are as follows:

$$\begin{array}{rcl@{\qquad}rcl} \mu^0 Z.\Phi & = & \texttt{false} & \nu^0 Z.\Phi & = & \texttt{true} \\ \mu^{\beta+1} Z.\Phi & = & \Phi[Z/\mu^\beta Z.\Phi] & \nu^{\beta+1} Z.\Phi & = & \Phi[Z/\nu^\beta Z.\Phi] \\ \mu^\lambda Z.\Phi & = & \bigvee_{\beta<\lambda} \mu^\beta Z.\Phi & \nu^\lambda Z.\Phi & = & \bigwedge_{\beta<\lambda} \nu^\beta Z.\Phi \end{array}$$

where λ is a limit ordinal, and the notation $\Phi[Z/\nu^\beta Z.\Phi]$ denotes the formula obtained from Φ by replacing each occurrence of Z by $\nu^\beta Z.\Phi$. By Tarski and Knaster Theorem [85], the fixpoints and their approximates are connected by the following properties: given a transition system T and a state q of T,

- $q \in (\mu Z.\Phi)^T_{(v,V)}$ if and only if there exists an ordinal α such that $s \in (\mu^\alpha Z.\Phi)^T_{(v,V)}$ and, for every $\beta < \alpha$, it holds that $s \notin (\mu^\beta Z.\Phi)^{\mathcal{M}}_{(v,V)}$;
- $q \notin (\nu Z.\Phi)^T_{(v,V)}$ if and only if there exists an ordinal α such that $s \notin (\nu^\alpha Z.\Phi)^T_{(v,V)}$ and, for every $\beta < \alpha$, it holds that $q \in (\nu^\beta Z.\Phi)^{\mathcal{M}}_{(v,V)}$.

Since each approximate, including the ones corresponding exactly to the least and greatest fixpoints, can be written as an infinitary $\mathcal{L}_P$ formula, we get the thesis. □

With this lemma in place we can prove the invariance result.

Theorem 13. *Consider two transition systems $T_1 = \langle \Delta_1, Q_1, q_{10}, \rightarrow_1, \mathcal{I}_1 \rangle$ and $T_2 = \langle \Delta_2, Q_2, q_{20}, \rightarrow_2, \mathcal{I}_2 \rangle$. If $T_1 \approx T_2$, then, for every $\mu\mathcal{L}_p$ closed formula Φ*

$$T_1 \models \Phi \text{ if and only if } T_2 \models \Phi.$$

Proof. If $T_1 \approx T_2$ then for some bijection h we have $q_{10} \approx_h q_{20}$. This implies that $\tilde{\mathcal{I}}_1(q_{10}) \sim_h \tilde{\mathcal{I}}_2(q_{20})$. Now consider the variable valuations v_1 and v_2 defined as follows (notice that since Φ is closed such individual valuations are irrelevant in evaluating it): choose an arbitrary $d_1 \in \Delta_1$ and let, for all variables x, $v_1(x) = d_1$; if $d_1 \in adom(\mathcal{I}_1(q_1))$, let, for all x, $v_2(x) = h(d_1)$; else, choose $d_2 \notin adom(\mathcal{I}_2(q_2))$ and let, for all x, $v'_2(x) = d_2$.

Now, define a bijection h' such that for all $d \in adom(\mathcal{I}(q_1))$, $h'(d) = h(d)$, and if $d_1 \notin adom(\mathcal{I}_1(q_1))$, $h'(d_1) = d_2$. It can be seen that h' is a bijection between $adom(\mathcal{I}_1(q_1) \cup \text{IM}(v_1)$ and $adom(\mathcal{I}_2(q_2) \cup \text{IM}(v_2)$ such that $\tilde{\mathcal{I}}_1(q_1) \sim_{h'|_{adom(\mathcal{I}_1(q_1))}} \tilde{\mathcal{I}}_2(q_2)$ and for all variables x, $h'(v_1(x)) = v_2(x)$. Hence, by Lemma 2, we obtain the thesis. □

Thus, to check whether a transition system T satisfies a $\mu\mathcal{L}_p$ formula Φ, one can perform the check on any transition system T' that is bisimilar to T. This is particularly useful in those cases where T is infinite-state but admits some finite-state bisimilar transition system. We exploit this result later on.

6.4. Transition systems induced by a situation calculus theory

Among the various TSs, we are interested in those *induced* by models of the situation calculus action theory $\mathcal{D}$. Consider a model $\mathcal{M}$ of $\mathcal{D}$ with object domain Δ and situation domain $\mathcal{S}$. The TS *induced by* $\mathcal{M}$ is the labeled TS $T_{\mathcal{M}} = \langle \Delta, Q, q_0, \mathcal{I}, \rightarrow \rangle$, such that:

- $Q = \mathcal{S}$ is the set of *possible states*, each corresponding to a distinct executable situation in $\mathcal{S}$;
- $q_0 = S_0^{\mathcal{M}} \in Q$ is the *initial state*, with $S_0^{\mathcal{M}}$ the initial situation of $\mathcal{D}$;
- $\rightarrow \subseteq Q \times Q$ is the *transition relation* such that $q \rightarrow q'$ iff there exists some action a such that $\langle a, q\rangle \in Poss^{\mathcal{M}}$ and $q' = do^{\mathcal{M}}(a, q)$;
- $\mathcal{I} : Q \mapsto Int_{\Delta}^{\mathcal{F},C}$ is the *labeling function* associating each state (situation) q with the interpretation $\mathcal{I}(q) = \mathcal{I}_{\mathcal{M}}(q)$.

As it can be seen, the TS induced by a model $\mathcal{M}$ is essentially the tree of executable situations, with each situation labeled by an interpretation of fluents (and constants), corresponding to the interpretation associated by $\mathcal{M}$ to that situation. Notice that transitions do not carry any information about the corresponding triggering action.

We can now show that the semantics of $\mu\mathcal{L}_p$ on a model can alternatively be given in terms of the corresponding induced TS.

Theorem 14. *Let $\mathcal{D}$ be an action theory, $\mathcal{M}$ a model of $\mathcal{D}$ with (infinite) object domain Δ and situation domain $\mathcal{S}$, and $T_{\mathcal{M}}$ the corresponding induced TS. Then for every $\mu\mathcal{L}_p$ formula Φ (with no occurrence of action terms) we have that:*

$$(\Phi)^{\mathcal{M}}_{(v,V)} = (\Phi)^{T_{\mathcal{M}}}_{(v,V)}$$

Proof. By induction on the structure of Φ. For the base case of an open uniform situation-suppressed situation calculus formula φ, we need to prove that

$$(\varphi)^{\mathcal{M}}_{(v,V)} = \{s \in \mathcal{S} \mid \mathcal{M}, v \models \varphi[s]\} = (\varphi)^{T_{\mathcal{M}}}_{(v,V)} = \{s \in \mathcal{S} \mid \mathcal{I}(s), v \models \varphi\}.$$

This is indeed the case: since no action terms occur in φ and φ is uniform in s, the evaluation of φ depends only on the interpretation of each fluent (and constant) at s, i.e., on $\mathcal{I}_{\mathcal{M}}(s)$. Once this base case is settled, the inductive cases are straightforward. □

6.5. Abstract finite-state transition system

As shown above, satisfaction of $\mu\mathcal{L}_p$ formulas is preserved by persistence-preserving bisimulations. This holds even between an infinite- and a finite-state TS. When this is the case, the verification can be performed on the finite TS using standard μ-calculus model checking techniques, which essentially perform fixpoint computations on a finite state space. We next show how, for the case of bounded theories, one can construct a finite TS T_F that is bisimilar to the TS $T_{\mathcal{M}}$ induced by $\mathcal{M}$.

We construct T_F using Procedure 1. The procedure takes as input an action theory $\mathcal{D}$ (with complete information on the initial situation) bounded by b and a model $\mathcal{M}$ of $\mathcal{D}$ with infinite object domain Δ,[11] and returns a finite-state TS T_F bisimilar to $T_{\mathcal{M}}$. T_F is built incrementally, through iterative refinements of the set of states Q, the interpretation function $\mathcal{I}$, and the transition relation $\rightarrow$. Initially, Q contains only the initial state q_0 (line 2); $\mathcal{I}(q_0)$ interprets constants and fluents in the same way as $\mathcal{M}$ at the initial situation (line 3); and $\rightarrow$ is empty (line 4). The set Q_{te} contains the states of T_F to be "expanded" (initially q_0 only, line 5); this is done at each iteration of the **while** loop (lines 6–20), as explained next.

Firstly, a state q is extracted from Q_{te} (lines 7 and 8). Then, a finite subset O of objects from Δ is defined (line 9). The values from O, together with those from $adom(\mathcal{I}(q))$, are used, in combination with the action types, to generate actions executable on the interpretation $\mathcal{I}(q)$[12] (lines 10, 11). The particular choice of O guarantees that the set of generated actions, while finite, is fully representative, for the purpose of verification, of all the (possibly infinitely many) actions executable on $\mathcal{I}(q)$ (see Theorem 16). Moreover, the objects are chosen so as to maximize *reuse* of the objects occurring in the interpretation of the states already in Q.

The actual expansion step consists in computing, for each generated action, the interpretation $\mathcal{I}'$ obtained by executing the action on (a situation with interpretation) $\mathcal{I}(q)$. This is done by computing, on $\mathcal{I}(q)$, the answers to the right-hand side $\phi(a, \vec{y})$ of the (situation-suppressed) successor state axiom of each fluent F, with a set to the current action (line 12). Once $\mathcal{I}'$ has been computed, two cases are possible: either it is isomorphic to some interpretation $\mathcal{I}(q')$ labeling an existing state $q' \in Q$ (line 13), under some isomorphism that preserves $\mathcal{I}(q)$, or it is not (line 15). In the former case, the transition relation is simply updated with a transition from q to q' (line 14) and no new state is generated. We stress that, in this case, the isomorphism is defined over the whole Δ, not only over the active domains of the interpretations. In the latter case, a fresh state q' with labeling $\mathcal{I}(q')$ is added to Q, and the transition relation is updated with $q \rightarrow q'$ (lines 16). Further, q' is also added to Q_{te}, so as to be expanded in future iterations. The procedure iterates over the expansion step until the set Q_{te} is empty, i.e., until there are no more states to expand.

[11] In fact, given the object domain Δ, the model $\mathcal{M}$ is fully determined by $\mathcal{D}$ modulo object renaming.

[12] Notice that since $Poss(a, s)$ is uniform in s, the situation does not play any role in establishing whether, for given a and s, $Poss(a, s)$ holds. In fact, only the interpretation of fluents (and constants) at s matters. Consequently, one can take such an interpretation and safely suppress the situation argument.

Procedure 1 Computation of a finite-state TS persistence-preserving bisimilar to $T_{\mathcal{M}}$.

Input: A basic action theory $\mathcal{D}$ bounded by b, with complete information on S_0, and a model $\mathcal{M}$ of $\mathcal{D}$ with infinite object domain Δ
Output: A finite-state TS $T_F = \langle \Delta, Q, q_0, \mathcal{I}, \rightarrow \rangle$ persistence-preserving bisimilar to $T_{\mathcal{M}}$

1: **let** $\mathcal{F}$ the set of fluents of $\mathcal{D}$, C the set of constants explicitly mentioned in $\mathcal{D}$;
2: **let** $Q := \{q_0\}$, for q_0 a fresh state;
3: **let** $\mathcal{I}(q_0) = \mathcal{I}_{\mathcal{M}}(S_0^{\mathcal{M}})$;
4: **let** $\rightarrow := \emptyset$;
5: **let** $Q_{te} := \{q_0\}$;
6: **while** $(Q_{te} \neq \emptyset)$ **do**
7: **pick** $q \in Q_{te}$;
8: **let** $Q_{te} := Q_{te} - \{q\}$;
9: **let** $O \subseteq \Delta$ be any (finite) set of objects such that:
(i) $|O| = \max\{|\vec{x}| \mid A(\vec{x}) \in \mathcal{A}\}$;
(ii) $O \cap adom(\mathcal{I}(q)) = \emptyset$;
(iii) $|O \cap \bigcup_{q \in Q} adom(\mathcal{I}(q))|$ is maximal (subject to *(i)* and *(ii)*).
10: **for all** action types $A(\vec{x})$ of $\mathcal{D}$ **do**
11: **for all** valuations v such that $v(\vec{x}) \in (adom(\mathcal{I}(q)) \cup O)^{|\vec{x}|}$ and $\mathcal{I}(q), v \models Poss(A(\vec{x}))$ **do**
12: **let** $\mathcal{I}' = \langle \Delta, \cdot^{\mathcal{I}'} \rangle$ be an interpretation such that: *(i)* $c^{\mathcal{I}} = c^{\mathcal{M}}$, for all constants in C; *(ii)* $F^{\mathcal{I}'} = \{\vec{d} \mid \mathcal{I}(q), v[\vec{y}/\vec{d}] \models \phi_F(A(\vec{x}), \vec{y})\}$, for $\phi_F(a, \vec{y})$ the (situation-suppressed) RHS of the SSA of fluent F.
13: **if** (there exists $q' \in Q$ and an isomorphism h between $\mathcal{I}'$ and $\mathcal{I}(q')$ that is the identity on $adom(\mathcal{I}(q))$) **then**
14: $\rightarrow := \rightarrow \cup \{q \rightarrow q'\}$;
15: **else**
16: **let** $Q := Q \uplus \{q'\}$, for q' a fresh state;
$\mathcal{I}(q') := \mathcal{I}'$;
$\rightarrow := \rightarrow \cup \{q \rightarrow q'\}$;
$Q_{te} := Q_{te} \uplus \{q'\}$;
17: **end if**
18: **end for**
19: **end for**
20: **end while**
21: **return** $T_F = \langle \Delta, Q, q_0, \mathcal{I}, \rightarrow \rangle$

We observe that the choice of q' at line 14 guarantees the existence of an isomorphism h' between $\mathcal{I}'$ and $\mathcal{I}(q')$ that is the identity on $adom(\mathcal{I}(q))$. That is, any object occurring in $\mathcal{I}'$ that comes from $\mathcal{I}(q)$ must be mapped into itself. The purpose of this choice is to avoid adding a fresh state q'' (with interpretation $\mathcal{I}'$) to Q but *reuse* any state q' already in Q, if bisimilar to the candidate q''. This is a key step for the procedure to construct a transition system that is both finite and persistence-preserving bisimilar to $T_{\mathcal{M}}$.

We can now show that Procedure 1 terminates and returns a TS persistence-preserving bisimilar to $T_{\mathcal{M}}$. This result is split into two main results: Theorem 15, which shows that the procedure terminates, returning a finite TS, and Theorem 16, which shows that the obtained TS is indeed persistence-preserving bisimilar to $T_{\mathcal{M}}$.

To prove termination, we first derive a bound on the active domain of the interpretations labeling the states in Q.

Lemma 3. *There exists a value $b' = \sum_{F \in \mathcal{F}} b \cdot a_F + |C|$ such that, at any iteration of Procedure 1 and for any $q \in Q$, $|adom(\mathcal{I}(q))| \leq b'$, where b is the value bounding $\mathcal{D}$, a_F the arity of fluent F, and C the set of constants explicitly mentioned in $\mathcal{D}$.*

Proof. We first show that: (†) for every $q \in Q$, there exists a situation s executable in $\mathcal{D}$ such that $\mathcal{I}(q) = \mathcal{I}_{\mathcal{M}}(s)$. This intuitively means that, modulo situation suppression, every state of T_F is labeled by an interpretation that matches that of $\mathcal{M}$ on constants and fluents at some executable situation s.

The proof is by induction on Q. For q_0, the thesis follows by the definition of $\mathcal{I}(q_0)$ at line 3, as $S_0^{\mathcal{M}}$ is executable. For the induction step, consider $q \in Q$ and assume, by the induction hypothesis, that $\mathcal{I}(q)$ is as above, for an executable situation s. Then, for any valuation (of object variables) v, we have that $\mathcal{I}(q), v \models Poss(A(\vec{x}))$ if and only if $\mathcal{I}_{\mathcal{M}}(s), v \models Poss(A(\vec{x}))$, that is, by Theorem 7, $\mathcal{M}, v' \models Poss(A(\vec{x}), \sigma)$, for σ a situation variable and v' a situation calculus variable assignment analogous to v on all individual variables and such that $v'(\sigma) = s$. Thus, by line 11, $A(\vec{x})$ is executable at s (with respect to $\mathcal{M}$ and v). Similarly, for any fluent F and valuation v, we have that $\mathcal{I}(q), v \models \phi_F(A(\vec{x}), \vec{y})$ iff $\mathcal{M}, v' \models \phi_F(A(\vec{x}), \vec{y}, \sigma)$, that is, since $F(\vec{y}, do(a, \sigma)) \equiv \phi_F(a, \vec{y}, \sigma)$ (by definition of successor-state axiom), $\mathcal{I}(q), v \models \phi_F(A(\vec{x}), \vec{y})$ iff $\mathcal{M}, v' \models F(\vec{y}, do(A(\vec{x}), \sigma))$. But then, since by line 12, $F^{\mathcal{I}'} = \{\vec{d} \in \vec{\Delta} \mid \mathcal{I}(q), v[\vec{y}/\vec{d}] \models \phi_F(A(\vec{x}), \vec{y})\}$, it follows that $\mathcal{I}', v \models F(\vec{y})$ iff $\mathcal{M}, v'[\vec{y}/d] \models F(\vec{y}, do(A(\vec{x}), \sigma))$. Thus, $F^{\mathcal{I}'} = \{\vec{d} \in \Delta \mid \mathcal{M}, v'[\vec{y}/\vec{d}] \models F(x, do(A(\vec{x}), \sigma))\}$. Therefore, when a state q' is added to Q (line 16), its labeling $\mathcal{I}(q') = \mathcal{I}'$ is such that $\mathcal{I}(q') = \mathcal{I}_{\mathcal{M}}(do^{\mathcal{M}}(A^{\mathcal{M}}(v(\vec{x})), s))$. This proves (†).

Observe that (†) and the boundedness of $\mathcal{D}$ imply, together, that $|adom(\mathcal{I}(q))|$ is bounded, for any $q \in Q$. We denote by b' the bound on $|adom^{\mathcal{M}}(s)|$, for any executable situation s of $\mathcal{D}$, and on $|adom(\mathcal{I}(q))|$, for $q \in Q$. Notice that, in general, b' is different than b, in that the former bounds the number of *objects* occurring in the interpretations, while the latter bounds the number of *tuples* in the interpretation of fluents. To obtain b', observe that if the theory is bounded by b, then, for any model, the extension of each fluent $F \in \mathcal{F}$ at any executable situation contains at most b distinct tuples. Thus, the extension of the generic fluent F cannot contain, at any executable situation, more than $a_F \cdot b$ distinct objects, where a_F is the arity of F (the maximum number of tuples, each with distinct objects, distinct also from all others in the extension). As a result,

the extensions cannot contain, overall, more than $\sum_{F\in\mathcal{F}} a_F \cdot b$ distinct objects. Hence, considering that $\mathcal{I}(q)$ interprets both the fluents in $\mathcal{F}$ and the constants in C, it follows that $|adom(\mathcal{I}(q))| \leq \sum_{F\in\mathcal{F}} a_F \cdot b + |C| \doteq b'$. □

Then, we use the obtained bound to show that also the set of *all* objects occurring in the labelings of some state in Q, denoted $adom(Q)$, is bounded.

Lemma 4. *Let $adom(Q) = \bigcup_{q\in Q} adom(\mathcal{I}(q))$. At any iteration of Procedure 1, we have that $|adom(Q)| \leq 2b' + N$, for b' the bound on $|adom(\mathcal{I}(q))|$ defined as in Lemma 3, and N the maximum number of parameters of the action types in $\mathcal{D}$.*

Proof. By induction on the size of Q. For $Q = \{q_0\}$, we have that $adom(Q) = adom(\mathcal{I}(q_0))$, thus the thesis follows as, by Lemma 3, $|adom(\mathcal{I}(q_0))| < b'$. For $Q = \{q_0, \ldots, q_n\}$, assume, by induction hypothesis, that $|adom(Q)| < 2b' + N$. Since, by Lemma 3, the state $q \in Q_{te} \subseteq Q$ picked at line 7 is such that $|adom(\mathcal{I}(q))| \leq b'$ and Δ is infinite, then, by Theorem 10 (after applying Theorem 5, if action terms have to be suppressed in ϕ_F), $\mathcal{I}'$ (line 12) is such that $adom(\mathcal{I}') \subseteq adom(\mathcal{I}(q)) \cup v(\vec{x})$.[13] Now, observe that $v(\vec{x})$ may take values from O and that the constraints on the choice of O (line 9) require that the reuse of objects from $adom(Q)$ be maximized. That is, including fresh objects (with respect to $adom(Q)$) in O is allowed (in fact, required) *only* if needed to guarantee that $|O| = |\vec{x}|$ (while $O \cap adom(\mathcal{I}(q)) = \emptyset$). Thus, two cases are possible: either $|adom(Q) \setminus adom(\mathcal{I}(q))| < |\vec{x}|$ (in which case fresh objects must be added to O), or not. In the first case, because $|\vec{x}| \leq N$ and $adom(\mathcal{I}(q)) \subseteq adom(Q)$, it follows that $|adom(Q)| - |adom(\mathcal{I}(q))| < N$. Thus, since $|adom(\mathcal{I}(q))| \leq b'$, we have that $|adom(Q)| < N + b'$. From this, observing that $|adom(\mathcal{I}(q'))| \leq b'$, we obtain $|adom(Q \cup \{q'\})| \leq 2b' + N$. In the second case, O contains no fresh objects, thus $|adom(Q \cup \{q'\})| = |adom(Q)| \leq 2b' + N$. □

Exploiting this result, we can prove termination.

Theorem 15. *Procedure 1 terminates and returns a finite-state transition system T_F.*

Proof. Firstly, observe that, as a consequence of Lemma 4: *(i)* checking whether $\mathcal{I}(q), v \models Poss(A(\vec{x}))$ (line 11) is decidable, and *(ii)* $F^{\mathcal{I}'}$ (line 12) is computable. These, indeed, are implied by the fact that $|adom(\mathcal{I}(q))|$ is bounded, thus finite, and by Theorems 9 and 11, respectively. To apply these theorems, however, one needs to suppress action terms first, if present, in formulas $\phi_F(A(\vec{x}), \vec{y})$ and $\phi_A(\vec{x})$. To this end, Theorem 5 can be used. Notice also that computability of $F^{\mathcal{I}(q_0)}$ (line 3) is a direct consequence of the fact that $\mathcal{D}$ has complete information and is bounded, therefore the extension of all fluents at S_0 is finite. Items *(i)* and *(ii)* above guarantee that all the atomic steps of Procedure 1 can be completed in finite time.

Next, we prove that eventually $Q_{te} = \emptyset$. Observe that, since $\mathcal{A}$ (i.e., the set of action types of $\mathcal{D}$), Q, O, $adom^{\mathcal{M}}(S_0)$, and $adom(\mathcal{I}(q))$ are finite, it follows that, at every iteration of the while-loop (lines 6–20), the nested loops (lines 10–19) terminate; thus, proving that Q_{te} becomes empty in a finite number of steps is sufficient to prove that only a finite number of iterations are executed and, hence, the procedure terminates. Obviously, this also implies that the returned Q, thus T_F, is finite.

To see that eventually $Q_{te} = \emptyset$, notice that Q is *inflationary*, i.e., states, once added, are never removed. Consequently, objects can be added to $adom(Q)$ (when a fresh q' is added) but not removed. This, together with the fact that, by Lemma 4, $|adom(Q)|$ is bounded, implies that, from some iteration i on, $adom(Q)$ remains unchanged. Let AQ_i be $adom(Q)$ at iteration i (and at subsequent steps). Obviously, after that point, if a fresh state q' is added, it must be such that $adom(\mathcal{I}(q')) \subseteq AQ_i$. Notice that, even though $adom(Q)$ cannot change, this is not the case for Q. Indeed, new states q' could still be added, as long as $\mathcal{I}(q') = \mathcal{I}'$ contains only objects from AQ_i. However, since $|adom(Q)|$, thus $|AQ_i|$, is bounded, only finitely many interpretations $\mathcal{I}'$ can be built using values from AQ_i. Consequently, if new states keep being introduced after i, it follows that, from some step i' on, the interpretation $\mathcal{I}'$ generated at line 12 matches the interpretation $\mathcal{I}(q')$ of some q' already in Q. Hence, from i' on, the condition at line 13 is always satisfied (with h the identity function), and no fresh state q' can be added to Q any more. Therefore, no new state is added to Q_{te} (line 16), which becomes eventually empty, as at every iteration one state is extracted from it (line 7). This completes the proof. □

Finally, we show that the returned T_F retains all the information needed to check whether $\mathcal{M} \models \Phi$. That is, by Theorem 13, we show that T_F is persistence-preserving bisimilar to $T_{\mathcal{M}}$.

Theorem 16. *The TS T_F computed by Procedure 1, on a basic action theory $\mathcal{D}$ (with complete information) bounded by b and a model $\mathcal{M}$ for $\mathcal{D}$, is persistence-preserving bisimilar to the TS $T_{\mathcal{M}}$ induced by $\mathcal{M}$.*

Proof. Let $T_F = \langle \Delta, Q, q_0, \mathcal{I}_F, \rightarrow_F \rangle$ and $T_{\mathcal{M}} = \langle \Delta, R, r_0, \mathcal{I}_{\mathcal{M}}, \rightarrow_{\mathcal{M}} \rangle$, and define the relation $B \subseteq Q \times H \times R$ such that $\langle q, h, r \rangle \in B$ if and only if $\tilde{\mathcal{I}}_F(q) \sim_h \tilde{\mathcal{I}}_{\mathcal{M}}(r)$ (for any h). Notice that, since T_F and $T_{\mathcal{M}}$ have the same object domain Δ, h can

[13] To simplify the notation, we use $v(\vec{x})$ for the set $\{v(x_1), \ldots, v(x_n)\}$.

always be extended to a standard isomorphism $\hat{h}$ between $\mathcal{I}_F(q)$ and $\mathcal{I}_{\mathcal{M}}(r)$: namely, one can take any bijection $\hat{h} : \Delta \mapsto \Delta$ such that $\hat{h}|_{adom(\mathcal{I}_F(q))} = h$.

We show that B is a persistence-preserving bisimulation between T_F and $T_{\mathcal{M}}$ (page 187). Consider a tuple $\langle q, h, r\rangle \in B$. Requirement 1 of the definition is trivially satisfied by the definition of B. As to requirement 2, let $q' \in Q$ be such that $q \to_F q'$. As shown in the proof of Theorem 15, there exists an executable situation s such that $\mathcal{I}_F(q) = \mathcal{I}_{\mathcal{M}}(s)$. Moreover, by the definition of $T_{\mathcal{M}}$, r is a situation such that $\mathcal{I}_{\mathcal{M}}(r)$ matches the interpretation given by $\mathcal{M}$ to fluents at r. Because $q \to_F q'$, by the construction of T_F in Procedure 1 (line 11), we have that, for some valuation v and action type A, $\mathcal{I}_F(q), v \models Poss(A(\vec{x}))$, that is, by the existence of s as above, $\mathcal{M}, v \models Poss(A(\vec{x}), s)$. Then, by extending h to an isomorphism $\hat{h}$ between $\mathcal{I}_F(q)$ and $\mathcal{I}_{\mathcal{M}}(r)$, as discussed above, we can see that $\mathcal{I}_{\mathcal{M}}(r), v' \models Poss(A(\vec{x}), r)$, for $v' = \hat{h} \circ v$, which implies that $\mathcal{M}, v' \models Poss(A(\vec{x}), r)$. Therefore, by the definition of $T_{\mathcal{M}}$, for $r' = do^{\mathcal{M}}(A^{\mathcal{M}}(\hat{h}(v(\vec{x})), r) \in R$, we have that $r \to'_{\mathcal{M}} r$. Thus requirement 2a is fulfilled.

Next, we show the existence of an isomorphism $\hat{h}'$ between $\mathcal{I}_F(q')$ and $\mathcal{I}_{\mathcal{M}}(r')$ that extends h. Once proven, this implies the existence of a bijection $h' : adom(q) \cup adom(q') \mapsto adom(r) \cup adom(r')$ such that $h'|_{adom(\mathcal{I}_F(q))} = h$ and $\tilde{\mathcal{I}}_F(q') \sim_{h'|_{adom(\mathcal{I}_F(q'))}} \tilde{\mathcal{I}}_{\mathcal{M}}(r')$. Indeed, it is sufficient to take $h' = \hat{h}'|_{adom(q) \cup adom(q')}$. Thus, the existence of $\hat{h}'$ implies requirement 2b.

To prove that such an $\hat{h}'$ exists, we distinguish two cases: *(i)* when the transition $q \to_F q'$ is added at line 16 (i.e., q' is a fresh state), and *(ii)* when it is added at line 14 (i.e., q' is already in Q). For case *(i)*, observe that $\mathcal{I}_{\mathcal{M}}(r')$ can be obtained by applying the right-hand side of the successor-state axiom of each fluent F to $\mathcal{I}_{\mathcal{M}}(r)$ (see Theorem 12), which is also the way to obtain $\mathcal{I}_F(q')$ from $\mathcal{I}_F(q)$, according to Procedure 1. Then, since $\hat{h}$ is an isomorphism between $\mathcal{I}_F(q)$ and $\mathcal{I}_{\mathcal{M}}(r)$, we have that $\mathcal{I}_{\mathcal{M}}(r) = \hat{h}(\mathcal{I}_F(q))$, where $\hat{h}(\mathcal{I}_F(q))$ denotes the interpretation obtained from $\mathcal{I}_F(q)$ by renaming its objects according to $\hat{h}$. Because $v' = \hat{h} \circ v$, it can be checked that $\mathcal{I}_{\mathcal{M}}(r') = \hat{h}(\mathcal{I}_F(q'))$, thus $\hat{h}' = \hat{h}$ is an isomorphism between $\mathcal{I}_F(q')$ and $\mathcal{I}_{\mathcal{M}}(r')$, which obviously extends h. For case *(ii)*, let $\mathcal{I}'$ be the interpretation obtained by applying the successor-state axioms to $\mathcal{I}_F(q)$. By the discussion above, we have that $\mathcal{I}_{\mathcal{M}}(r') = \hat{h}(\mathcal{I}')$, while, in general, $\mathcal{I}' \neq \mathcal{I}_F(q')$. However, the condition at line 13 guarantees the existence of an isomorphism g such that $\mathcal{I}' = g(\mathcal{I}_F(q'))$, that is the identity on $adom(\mathcal{I}_F(q))$. Now, consider $\hat{h}' = \hat{h} \circ g$. Being a composition of isomorphisms, $\hat{h}'$ is an isomorphism itself, in particular such that $\mathcal{I}_{\mathcal{M}}(r') = \hat{h}'(\mathcal{I}_F(q'))$. Moreover, $\hat{h}'$ extends $h|_{adom(\mathcal{I}_F(q))}$. This is a straightforward consequence of the facts that $\hat{h}$ extends h and g is the identity on $adom(\mathcal{I}_F(q))$, which imply that $\hat{h}'$ matches h on $adom(\mathcal{I}_F(q))$. Thus, requirement 2 is fulfilled. The proof for requirement 3 follows the same argument, with h replaced by its inverse h^{-1}.

Since B is a persistence-preserving bisimulation, the fact that $\langle q_0, h_0, r_0\rangle \in B$, for h_0 the identity, completes the proof. □

Next we prove that checking whether T_F satisfies a $\mu\mathcal{L}_p$ formula is decidable.

Theorem 17. *Given a transition system $T = \langle \Delta, Q, q_0, \mathcal{I}, \to\rangle$, if Q is finite and, for every $q \in Q$, $adom(\mathcal{I}(q))$ is finite, then for every $\mu\mathcal{L}_p$ formula Φ, checking whether $T \models \Phi$ is decidable.*

Proof. Firstly, by applying Theorem 6 followed by Theorem 8 to the FO components of Φ, we rewrite Φ as an equivalent $\mu\mathcal{L}_p$ (closed) formula Φ' where no action terms occur and whose FO components are domain-independent. Once done so, the theorem is a consequence of the finiteness of Q and $adom(q)$, for $q \in Q$. Under these assumptions, $(\Phi')^T_{(v,V)}$ is easily computable by recursive applications of the definition of $(\cdot)^T_{(v,V)}$ (page 186). In particular, for the base case of Φ' a FO formula φ', since φ' is action-term-free and domain-independent, one can apply Theorem 9. As to quantified variables (outside the FO components), they can be easily dealt with, by the finiteness of $adom(q)$. The other cases are straightforward. □

Finally, putting all the above results together, we obtain Theorem 4, by observing that one can compute T_F using Procedure 1 and then check whether $T_F \models \Phi$ by Theorem 17. Termination and correctness of this construction are guaranteed by Theorems 13, 15, and 16.[14]

7. Dealing with incomplete information

In this section, we address the case of partial information on the initial situation, by assuming that $\mathcal{D}_0$ is a set of axioms characterizing a possibly infinite set of bounded initial databases. Also in this case, we focus on theories whose models have infinite object domains (as we have infinitely many distinct constants).

We first prove that whenever two models interpret their respective initial situations in isomorphic ways, they are persistence-preserving bisimilar. We observe that this result holds independently of the cardinalities of the object domains of the models.

[14] Notice that no assumption is made on the object domain Δ of $\mathcal{M}$ except for it to be infinite. Hence, these results hold also if we assume standard names for object domains, as done in [27]: in that case the object domain is infinite but numerable and coincides with the set of constants $\mathcal{N}$ (this requires a second-order domain closure axiom).

Theorem 18. *Let $\mathcal{D}$ be a bounded basic action theory. For every two models $\mathcal{M}$ and $\mathcal{M}'$ of $\mathcal{D}$, with possibly different infinite object domains Δ and Δ', respectively, if $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \tilde{\mathcal{I}}_{\mathcal{M}'}(S_0^{\mathcal{M}'})$, then $T_{\mathcal{M}} \approx T_{\mathcal{M}'}$.*

Proof. Let $T_{\mathcal{M}} = \langle \Delta, Q, q_0, \rightarrow, \mathcal{I} \rangle$ and $T_{\mathcal{M}'} = \langle \Delta', Q', q_0', \rightarrow', \mathcal{I}' \rangle$. We prove a stronger claim, i.e., that the relation $B \subseteq Q \times H \times Q'$ such that $\langle q_1, h, q_2 \rangle \in B$ if and only if $\tilde{\mathcal{I}}(q_1) \sim_h \tilde{\mathcal{I}}'(q_2)$ (for any h), is a persistence-preserving bisimulation relation between $T_{\mathcal{M}}$ and $T_{\mathcal{M}'}$. This result, once proven, implies the thesis; indeed, by $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \tilde{\mathcal{I}}_{\mathcal{M}'}(S_0^{\mathcal{M}'})$, we have that there exists $\bar{h}$ such that $\tilde{\mathcal{I}}(S_0^{\mathcal{M}}) \sim_{\bar{h}} \tilde{\mathcal{I}}(S_0^{\mathcal{M}'})$, thus, by the definition of B, $\langle S_0^{\mathcal{M}}, \bar{h}, S_0^{\mathcal{M}'} \rangle \in B$, that is, $\langle q_0, \bar{h}, q_0' \rangle \in B$, as $q_0 = S_0^{\mathcal{M}}$ and $q_0' = S_0^{\mathcal{M}'}$.

Let $\langle q_1, h, q_2 \rangle \in B$. Requirement 1 of the definition of bisimulation (page 187) is clearly satisfied. For requirement 2, first recall that, by definition of induced transition system (page 189), $\mathcal{I}(q_1) = \mathcal{I}_{\mathcal{M}}(q_1)$ and $\mathcal{I}'(q_2) = \mathcal{I}_{\mathcal{M}'}(q_2)$, thus $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1) \sim_h \tilde{\mathcal{I}}_{\mathcal{M}'}(q_2)$. Assume that there exists $q_1' \in Q$ such that $q_1 \rightarrow q_1'$. By definition of transition system induced by $\mathcal{M}$ (page 189), there exist an action type A and a valuation v such that $\mathcal{M}, v \models \phi_A(\vec{x}, q_1)$, for $Poss(A(\vec{x}), s) \equiv \phi_A(\vec{x}, s)$ the precondition axiom of A. This is equivalent to $\mathcal{I}_{\mathcal{M}}(q_1), v \models \phi_A(\vec{x})$, for $\phi_A(\vec{x})$ the situation-suppressed version of $\phi_A(\vec{x}, s)$. Now, let $\phi_A'(\vec{x})$ be the domain-independent version of $\phi_A(\vec{x})$. By Theorem 8, we have that $\mathcal{I}_{\mathcal{M}}(q_1), v \models \phi_A(\vec{x})$ if and only if $\tilde{\mathcal{I}}_{\mathcal{M}}^{\phi_A, v}(q_1), v \models \phi_A'(\vec{x})$. If we extend h to $v(\vec{x})$ in a way such that we obtain a bijection $\hat{h}$ (by a cardinality argument, this is always possible), then, because $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1) \sim_h \tilde{\mathcal{I}}_{\mathcal{M}'}(q_2)$, we have that $\tilde{\mathcal{I}}_{\mathcal{M}}^{\varphi, v}(q_1), v \models \phi_A'(\vec{x})$ if and only if $\tilde{\mathcal{I}}_{\mathcal{M}'}^{\varphi, \hat{h} \circ v}(q_2), \hat{h} \circ v \models \phi_A'(\vec{x})$. But then, again by Theorem 8, $\mathcal{I}_{\mathcal{M}'}(q_2), \hat{h} \circ v \models \phi_A(\vec{x})$. Thus, by reintroducing the situation argument in ϕ_A, we have that $\mathcal{M}', v' \models \phi_A(\vec{x}, q_2)$, that is, there exists an action $a' = A^{\mathcal{M}'}(\hat{h}(v(\vec{x})))$ such that $\langle a', q_2 \rangle \in Poss^{\mathcal{M}'}$. Therefore, by the definition of $T_{\mathcal{M}'}$, it follows that $q_2 \rightarrow q_2'$, for $q_2' = do^{\mathcal{M}}(a', q_2)$. This proves requirement 2a.

For requirement 2b, we first show that $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1')$ can be obtained from $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1)$, through the successor-state axioms. To this end, notice that $\mathcal{I}_{\mathcal{M}}(q_1')$ can be obtained by taking, for each fluent F, the right-hand side $\phi(\vec{x}, a, s)$ of the corresponding successor-state axiom (the subscript F is removed to simplify the notation), then deriving the equivalent action-term-free formula $\phi(\vec{y}, \vec{x})$, as shown in Theorem 12, for action $a = A^{\mathcal{M}}(v(\vec{x}))$, and finally letting $F^{\mathcal{I}_{\mathcal{M}}(q_1')} = \phi_{\vec{x}/v(\vec{x})}^{\mathcal{I}_{\mathcal{M}}(q_1)}$, that is, by interpreting each F as the answer to the corresponding query ϕ on the interpretation $\mathcal{I}_{\mathcal{M}}(q_1)$, under the partial assignment $\vec{x}/v(\vec{x})$ (constants are always interpreted as in $\mathcal{M}$). Now observe that, since the action theory is bounded, so is the extension of each fluent F at q_1 and q_1'. Thus, by Theorem 10, the extension of each fluent at q_1' contains only values from $adom(\mathcal{I}_{\mathcal{M}}(q_1)) \cup v(\vec{x})$, that is $adom(\mathcal{I}_{\mathcal{M}}(q_1')) \subseteq adom(\mathcal{I}_{\mathcal{M}}(q_1)) \cup v(\vec{x})$. Hence, if we denote (for each F) the domain-independent rewriting of $\phi(\vec{y}, \vec{x})$ as $\phi'(\vec{y}, \vec{x})$, by Theorem 8, we have that $F^{\mathcal{I}_{\mathcal{M}}(q_1')} = \phi_{\vec{x}/v(\vec{x})}^{\mathcal{I}_{\mathcal{M}}(q_1)} = \phi'^{\tilde{\mathcal{I}}_{\mathcal{M}}(q_1)}_{\vec{x}/v(\vec{x})}$, that is, by answering ϕ' on $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1)$, we obtain the extension of F at q_1'. Obviously, by doing so for every fluent F, we can obtain $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1')$ from $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1)$. By an analogous argument, it can be shown that $\tilde{\mathcal{I}}_{\mathcal{M}'}(q_2')$ can be obtained from $\tilde{\mathcal{I}}_{\mathcal{M}'}(q_2)$, for action $a' = A^{\mathcal{M}'}(\hat{h}(v(\vec{x})))$.

Next, consider again the bijection $\hat{h}$ defined above, and recall that $\hat{h}$ extends h on $v(\vec{x})$, and that $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1) \sim_h \tilde{\mathcal{I}}_{\mathcal{M}'}(q_2)$. By the invariance of FO under isomorphic interpretations, we have that, for each fluent F, the answers to ϕ' on $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1)$ and $\tilde{\mathcal{I}}_{\mathcal{M}'}(q_2)$, under the partial assignments, respectively, $\vec{x}/v(\vec{x})$ and $\vec{x}/\hat{h}(v(\vec{x}))$, coincide, modulo the object renaming induced by $\hat{h}$. But then, it is immediate to check that $h' = \hat{h}|_{adom(\mathcal{I}_{\mathcal{M}}(q_1)) \cup adom(\mathcal{I}_{\mathcal{M}}(q_1'))}$ is a bijection such that $\tilde{\mathcal{I}}_{\mathcal{M}}(q_1') \sim_{h'|_{adom(\mathcal{I}_{\mathcal{M}}(q_1'))}} \tilde{\mathcal{I}}_{\mathcal{M}'}(q_2')$ and, hence, by the definition of B, $\langle q_1', h'|_{adom(\mathcal{I}_{\mathcal{M}}(q_1'))}, q_2' \rangle \in B$. This proves requirement 2b. The proof of requirement 3b is analogous. □

Now, consider a set *Mod* of models of $\mathcal{D}$ having isomorphic interpretations at S_0. By Theorem 18, all such models have induced TSs that are persistence-preserving bisimilar to each other. Thus, by Theorem 13, to check whether a $\mu\mathcal{L}_p$ formula ϕ holds in all models of *Mod*, one can perform the check on any arbitrary model of *Mod*, using, e.g., the technique discussed for the case of complete information. This result, together with the assumption of boundedness, will be exploited next, to prove our main theorem.

Theorem 19. *Let $\mathcal{D}$ be an action theory bounded by b with incomplete information on the initial situation, and let Φ be a $\mu\mathcal{L}_p$ closed formula. Then, checking whether $\mathcal{D} \models \Phi$ is decidable.*

Proof. Let $Mod_{\mathcal{D}}$ be the set of all models of $\mathcal{D}$, and consider a partition of it such that each cell contains only models whose interpretations at S_0 match, modulo object renaming. Formally, we define $Mod_{\mathcal{D}} = (Mod_{\mathcal{D}}^1, Mod_{\mathcal{D}}^2, \ldots)$ such that, for every two models $\mathcal{M}$ and $\mathcal{M}'$ in $Mod_{\mathcal{D}}^i$, $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \tilde{\mathcal{I}}_{\mathcal{M}'}(S_0^{\mathcal{M}'})$. As a consequence of the boundedness of $\mathcal{D}$, the number of cells in the partition is finite. Indeed, a bounded number of objects yields, up to object renaming, only a bounded number of possible interpretations (of finitely many fluents and constants) at S_0. Thus, for some finite n depending on the theory $\mathcal{D}$ and the bound b, we have that $Mod_{\mathcal{D}} = (Mod_{\mathcal{D}}^1, Mod_{\mathcal{D}}^2, \ldots, Mod_{\mathcal{D}}^n)$.

Since, by Theorem 18, any two models $\mathcal{M}$ and $\mathcal{M}'$ of the generic cell $Mod_{\mathcal{D}}^i$ induce persistence-preserving bisimilar transition systems, then, by Theorem 13, we have that all the models of $Mod_{\mathcal{D}}^i$ satisfy Φ if and only if some model $\mathcal{M}$ of

$Mod^i_{\mathcal{D}}$ satisfies Φ. Thus, to check whether $\mathcal{D} \models \Phi$, we can simply choose one model $\mathcal{M}_i$ per cell $Mod^i_{\mathcal{D}}$, and then check whether, for all $i = 1, \ldots, n$, $\mathcal{M}_i \models \Phi$; if this is the case, then, and only then, we can conclude that $\mathcal{D} \models \Phi$. Obviously, for this approach to be effective, we need a model $\mathcal{M}_i$ per cell $Mod^i_{\mathcal{D}}$ and a way to perform the check. The rest of the proof addresses these two points.

Let $\mathcal{F}$ be the set of situation-suppressed fluents of $\mathcal{D}$, and C the (finite) set of constant symbols explicitly mentioned in $\mathcal{D}$ (beyond $\mathcal{D}_{uno}$). We observe that each cell $Mod^i_{\mathcal{D}}$ of the partition $Mod_{\mathcal{D}} = (Mod^1_{\mathcal{D}}, \ldots, Mod^n_{\mathcal{D}})$ can be uniquely identified by an interpretation $\mathcal{I}_i$ of $\mathcal{F}$ and C over some infinite object domain Δ. Indeed, by transitivity of $\sim$, any two models $\mathcal{M}, \mathcal{M}'$ of $\mathcal{D}$ such that $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \mathcal{I}_i$ and $\tilde{\mathcal{I}}'_{\mathcal{M}}(S_0^{\mathcal{M}'}) \sim \mathcal{I}_i$ are also such that $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \tilde{\mathcal{I}}'_{\mathcal{M}}(S_0^{\mathcal{M}'})$. Notice that $\mathcal{I}_i$ certainly exists, as one can simply take $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}})$, for some model $\mathcal{M} \in Mod^i_{\mathcal{D}}$. Clearly, each $\mathcal{I}_i$ contains only a bounded number of objects in the active domain and satisfies $\mathcal{D}_0$, i.e., $\mathcal{I}_i \models \mathcal{D}_0$.

Now, assume given one interpretation $\mathcal{I}_i$ per cell $Mod^i_{\mathcal{D}}$ (we show below how to obtain them) and observe that, from $\mathcal{I}_i$, we can extract a complete initial situation description as a database $\mathcal{D}^i_0$. This can be easily done, as $\mathcal{I}_i$ is finite. Consider the theory $\mathcal{D}^i = (\mathcal{D} \setminus \mathcal{D}_0) \cup \mathcal{D}^i_0$, obtained by replacing $\mathcal{D}_0$ with $\mathcal{D}^i_0$, and assume the same interpretation of constants in C as that defined by $\mathcal{I}_i$. Under this assumption, $\mathcal{D}^i$ defines a family of models that differ only in the object domain and in the interpretation of constants outside C (which, however, must satisfy $\mathcal{D}_{uno}$). In particular, the interpretation of fluents in $\mathcal{F}$ and constants in C, at S_0, of all such models, is the same as that of $\mathcal{I}_i$. Thus, the models of $\mathcal{D}^i$ constitute a subset of $Mod^i_{\mathcal{D}}$. To isolate one of such models, we fix an arbitrary infinite object domain Δ (such that $adom(\mathcal{I}_i) \subseteq \Delta$), and arbitrarily extend the partial interpretation of constants over the constants outside C, satisfying $\mathcal{D}_{uno}$. Notice that this can always be done, as Δ is infinite and the set of constant symbols countable. With Δ and the denotation of all constants fixed, $\mathcal{D}^i$ has complete information, i.e., yields a single model $\mathcal{M}_i$, thus, by Theorem 4, we can check whether $\mathcal{D}^i \models \Phi$, i.e., whether $\mathcal{M}^i \models \Phi$ (notice that, as it turns out from Procedure 1, to perform the check, one does not even need to know the interpretation of constants outside C). This, by the discussion above, is equivalent to checking whether for all models $\mathcal{M} \in Mod^i_{\mathcal{D}}$, it is the case that $\mathcal{M} \models \Phi$. Therefore, if the set of interpretations $\Gamma = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ is given, we can check whether $\mathcal{D} \models \Phi$.

It remains to explain how such a set of interpretations $\Gamma = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ can be obtained. To this end, observe that, by Lemma 3, it follows that $|adom(\mathcal{I}_i)| \leq \sum_{F \in \mathcal{F}} a_F \cdot b + |C| \doteq b'$. Based on this, the set Υ of interpretations $\mathcal{I}_i$ can be obtained by: *(i)* fixing a set O of b' arbitrary objects; *(ii)* generating a set Υ' of all the finitely many interpretations of $\mathcal{F}$ and C over O, such that $\mathcal{D}_{uno}$ is enforced on C and for every interpretation $\mathcal{I}' \in \Upsilon'$, $\mathcal{I}' \models \mathcal{D}_0$; *(iii)* for any set $\Upsilon'' \subseteq \Upsilon'$ of isomorphic interpretations, removing from Υ' all but one of such interpretations (in fact, this step is not needed to our purposes, but avoids useless redundancies). The resulting Υ' is the set of desired interpretations $\mathcal{I}_1, \ldots, \mathcal{I}_n$, which we rename simply as Υ.

Now, observe that, by the way it is defined, Υ contains, up to object renaming, all possible interpretations of $\mathcal{F}$ and C over a set of b' distinct objects, that satisfy $\mathcal{D}_0$ and $\mathcal{D}_{uno}$ (on C). Thus, since for a generic model $\mathcal{M}$ of $\mathcal{D}$, the interpretation $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}})$ contains at most b' distinct objects (by the boundedness of $\mathcal{D}$), it turns out that there exists an interpretation $\mathcal{I}_i \in \Upsilon$ such that $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \mathcal{I}_i$. Therefore, the cell $Mod^i_{\mathcal{D}}$ such that $\mathcal{M} \in Mod^i_{\mathcal{D}}$, is characterized by some interpretation $\mathcal{I}_i \in \Upsilon$, namely the interpretation at S_0 shared, up to object renaming, by the models of the cell itself. On the other hand, because any $\mathcal{I}_i \in \Upsilon$ enforces $\mathcal{D}_{uno}$ and is such that $\mathcal{I} \models \mathcal{D}_0$, it follows that there exists some model $\mathcal{M}$ of $\mathcal{D}$ such that $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \mathcal{I}_i$. Therefore, every interpretation of Υ characterizes some cell $Mod^i_{\mathcal{D}}$, specifically, that of the models $\mathcal{M}$ such that $\tilde{\mathcal{I}}_{\mathcal{M}}(S_0^{\mathcal{M}}) \sim \mathcal{I}_i$. Therefore, Υ is indeed the set of desired interpretations. This concludes the proof. □

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

8. Computational complexity

In this section, we assess the computational complexity of verifying $\mu\mathcal{L}_p$ formulas over a bounded situation calculus basic action theory $\mathcal{D}$. In particular we show that the constructive techniques we have used for proving decidability are, in fact, optimal with respect to worst case computational complexity. We make the assumption that, for a basic action theory $\mathcal{D}$, the maximum number of distinct objects occurring in the state of any situation, dominates the input size of $\mathcal{D}$ itself, and that there exists a bound $\bar{a}_F$ on the maximum arity of fluents. This is a reasonable assumption, analogous to that, typical in databases, that the size of the database provides a higher bound on the size of the input along all dimensions, and that, in practical cases, there exists an upper bound on the arity of relations. We exploit the constructive techniques introduced for showing decidability to get an exponential time upper-bound.

Theorem 20. *Verifying $\mu\mathcal{L}_p$ formulas over a situation calculus basic action theory bounded by b, with complete information on the initial situation, can be done in time exponential in b.*

Proof. This is a consequence of Procedure 1 and the complexity of $\mu\mathcal{L}_p$ model checking. Firstly, consider Procedure 1 and observe that, by Lemma 4, at any iteration, the number m of distinct objects occurring, overall, in the interpretations of states (i.e. $|adom(Q)|$ of Lemma 4) is bounded by $2b' + N$, where $b' = \sum_{F \in \mathcal{F}} b \cdot a_F$, a_F is the arity of fluent F, and N is the maximum number of parameters in action types. Since we assume $|F|$ and N bounded by b, and a_F bounded by a constant,

it turns out that m is polynomial in b. Now, observe that, with m distinct objects and a_F bounded by a constant, one can obtain a number of interpretations of $\mathcal{F}$ and C that is at most exponential in m, i.e., in (a polynomial of) b. Then, because in Procedure 1 every state is associated with exactly one interpretation, and since no state is visited more than once, we have that the while-loop (lines 6–20) terminates after, at most, an exponential number of iterations.

As to each iteration, by our assumptions, we have that any loop inside the while-loop ends after at most exponentially many iterations. Indeed, for any action type with at most N parameters, we have at most m^N possible assignments, thus $m^N \leq m^{b'}$, which gives an exponential bound, as both m and b' are polynomial with respect to b. Now, observe that the dominant operation in the while-loop is checking whether two interpretations are isomorphic. Since also this check can be performed in exponential time with respect to b (the problem is in NP), we obtain, overall, an exponential time-bound for Procedure 1.

Now, recall that propositional μ-calculus model checking is polynomial with respect to the sizes of the input transition system and the input formula [40]. As to the transition system, the check is performed on the one returned by Procedure 1, which has size at most exponential in b (i.e., as many interpretations as one can obtain with at most m objects, plus a quadratic number of transitions wrt it). As to the formula, say Φ, we first rewrite it (in polynomial time) into its equivalent domain-independent version Φ', and then "propositionalize" it, by quantifier elimination, using only the values that occur, overall, in the active domains of the interpretations of the states of the input transition system. This step can be done, again, in exponential time, and returns a quantifier-free formula exponentially larger than the original one, but equivalent to it, on the obtained finite transition system. Thus, since μ-calculus model checking is polynomial wrt the size of both the transition system and the formula, we obtain that, overall, the check requires time at most exponential wrt b. □

Such an exponential bound is, in fact, tight, as we can show the EXPTIME-hardness of the problem by reduction from acceptance in a polynomial-space bounded alternating Turing machine.

Theorem 21. *Verifying $\mu\mathcal{L}_p$ formulas over bounded situation calculus basic action theories with complete information on the initial situation is EXPTIME-hard.*

Proof. We show a reduction from polynomial-space bounded alternating Turing machines, whose acceptance problem is EXPTIME-complete [18]. A (one-tape) *Alternating Turing Machine* (ATM) [18] is a tuple $M = (Q, \Gamma, \delta, q_0, g)$ where

- Q is the finite set of states;
- Γ is the finite tape alphabet;
- $\delta : Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is called the transition table (L shifts the head left and R shifts the head right);
- $q_0 \in Q$ is the initial state;
- $g : Q \rightarrow \{and, or, accept\}$ specifies the type of each state.

If M is in a state $q \in Q$ with $g(q) = accept$ then that configuration is said to be *accepting*. A configuration with $g(q) = and$ is said to be accepting if *all* configurations reachable in one step are accepting. A configuration with $g(q) = or$ is said to be accepting when there *exists* some configuration reachable in one step which is accepting. (The latter is the type of all states in a Nondeterministic Turing Machine.) M is said to accept an input string w if the initial configuration of M (where the state of M is q_0, the head is at the left end of the tape, and the tape contains w) is accepting. An ATM is said to be *polynomial-space-bounded* if it scans at most a number of tape cells that is polynomially-bounded by the size of the input.

Following [75] (Chap. 4), we can axiomatize the ATM using the following fluents:

- $transTable(q, c, q', c', m, s)$. This is a situation-independent predicate (i.e., with a trivial successor-state-axioms preserving its content forever) describing the ATM's transition table δ: when in state q scanning tape symbol c, the machine enters state q', overwrites c with tape symbol c', and moves its tape head in the direction m, which is one of L (left) or R (right).
- $gType(q, t, s)$. This is a situation-independent predicate assigning (once and for all) a type $t \in \{and, or, accept\}$ to the state q of the ATM.
- $cell(i, c, s)$. This means that tape cell $i \in [0, \ldots, \ell]$ contains the symbol $c \in \Gamma \cup \{blank\}$ in situation s. Notice that in every situation the number of facts of the form $cell(i, \gamma, s)$ is fixed and determined by the maximal length of the tape of the bounded ATM, ℓ. Initially, the first cells contains the input word w while the others are *blank*.
- $state(q, s)$. This means that in situation s, the machine's state is q. Initially, we have $state(q_0, S_0)$, where q_0 is the initial state of the ATM.
- $scan(i, s)$. This means that the machine's head is scanning tape cell $i \in [0, \ldots, \ell]$ in situation s. Initially, the head is scanning tape cell 0. In any situation, there will only be one fact of the form $scan(i, s)$.

We need just one action type $trans(q', c', m)$, meaning that the machine makes a transition from the current configuration to a new configuration where the state is q', tape symbol c' is written, and the tape head moves in direction m, whose precondition axiom is as follows:

$$Poss(trans(q', c', m), s) \equiv \exists q, i, c.\, state(q, s) \wedge scan(i, s) \wedge cell(i, c, s) \wedge transTable(q, c, q', c', m, s)$$

The successor state axioms for the fluents that can change are as follows:

$$state(q, do(a, s)) \equiv \exists c, m.a = trans(q, c, m) \vee state(q, s) \wedge \neg\exists q', c, m.a = trans(q', c, m) \wedge q' \neq q$$

$$\begin{aligned} scan(i, do(a, s)) \equiv\ & \exists q, c, i'.a = trans(q, c, L) \wedge scan(i', s) \wedge \\ & (i' = 0 \supset i = i') \wedge (i' \neq 0 \supset i = i' - 1) \vee \\ & \exists q, c.a = trans(q, c, R) \wedge scan(i', s) \wedge i = i' + 1 \vee \\ & scan(i, s) \wedge \neg\exists q, c, m.a = trans(q, c, m) \end{aligned}$$

$$cell(i, c, do(a, s)) \equiv \exists q, m.a = trans(q, c, m) \wedge scan(i, s) \vee cell(i, c, s) \wedge \neg\exists q, c', m.a = trans(q, c', m) \wedge scan(i, s) \wedge c' \neq c$$

For initial situation description, assuming the input $w = c_0 \ldots c_i$, we have:

$$\begin{aligned} & state(q_0, S_0), scan(0, S_0), \\ & cell(0, c_0, S_0), \ldots, cell(i, c_i, S_0), \\ & cell(j, blank, S_0), \text{ for } j \in [i, \ldots, \ell] \end{aligned}$$

Acceptance of the ATM is defined using the following $\mu\mathcal{L}_p$ formula Φ:

$$\begin{aligned} & \mu Z.\, (\exists q.state(q) \wedge gType(q, accept) \vee \\ & \quad (\exists q.state(q) \wedge gType(q, and)) \wedge [-]Z \vee \\ & \quad (\exists q.state(q) \wedge gType(q, or)) \wedge \langle - \rangle Z \end{aligned}$$

Then we have that $\mathcal{D} \models \Phi$ if and only if M accepts w. Notice that in any situation there is exactly one fact of the form $gType(q, t, s)$. Notice also that the above condition does not require quantification across situations. □

9. Checking boundedness

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

First notice that we can determine in a situation s whether every executable action a if performed next does not exceed the bound (i.e. in $do(a, s)$). We can capture the notion of a fluent F being bounded at the next step by the formula:

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

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

To check that the theory is bounded by b it is sufficient to verify that the theory entails the temporal formula:

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

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

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

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

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

Lemma 5.

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

Proof. By induction on situations. □

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

- $\mathcal{D}'_{S_0} = \{\phi[\vec{F}/\vec{F}'] \mid \phi \in \mathcal{D}_{S_0}\}$.
- $\mathcal{D}'_{SS} = \{F'(\vec{x}, do(a, s)) \equiv \Phi[\vec{F}/\vec{F}'](\vec{x}, a, s) \wedge \mathit{NextOrigBounded}_b[F/F'](s) \mid F(\vec{x}, do(a, s)) \equiv \Phi(\vec{x}, a, s) \in \mathcal{D}_{SS}\}$
- $\mathcal{D}'_{ap} = \{\mathit{Poss}(A(\vec{x}), s) \equiv \Psi[\vec{F}/\vec{F}'](\vec{x}, a, s) \wedge \mathit{NextOrigBounded}_b[F/F'](s) \mid \mathit{Poss}(A(\vec{x}), s) \equiv \Psi(\vec{x}, a, s) \in \mathcal{D}_{AP}\}$

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

Lemma 6.

$$\mathcal{D} \models \mathit{AGNextOrigBounded}_b(S_0) \text{ iff } \mathcal{D}' \models \mathit{AGNextOrigBounded}_b[\vec{F}/\vec{F}'](S_0).^{15}$$

Proof. By Lemma 5, it is immediate to see that $\mathcal{D} \models \mathit{AGNextOrigBounded}_b(S_0)$ implies $\mathcal{D}' \models \mathit{AGNextOrigBounded}_b[\vec{F}/\vec{F}'](S_0)$. For the opposite direction, suppose that $\mathcal{D}' \models \mathit{AGNextOrigBounded}_b[\vec{F}/\vec{F}'](S_0)$, but $\mathcal{D} \models \mathit{AGNextOrigBounded}_b(S_0)$ does not hold. This means that there exists a model of $\mathcal{D}$ and a situation S where $\neg\mathit{NextOrigBounded}_b(S)$ holds, though in all previous situations $s < S$ we have that $\mathit{NextOrigBounded}_b(s)$ holds. Now by Lemma 5, we can construct a model for $\mathcal{D}'$ such that the truth values of F are replicated in F' as long as $\mathit{NextOrigBounded}_b$ holds in the previous situation. So in S, we must have $\neg\mathit{NextOrigBounded}_b[\vec{F}/\vec{F}'](S)$, which contradicts the assumption that $\mathcal{D}' \models \mathit{AGNextOrigBounded}_b[\vec{F}/\vec{F}'](S_0)$. □

By Lemma 6, since $\mathcal{D}'$ is bounded by b if $\mathcal{D}'_{S_0}$ is, it follows that:

Theorem 22. *Given a basic action theory whose initial situation description is bounded by b, then checking whether the entire theory is bounded by b is decidable.*

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

Theorem 23. *Given a FO description of the initial situation $\mathcal{D}_0$ and a bound b, it is undecidable to check whether all models of $\mathcal{D}_0$ are bounded by b.*

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

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

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

Nonetheless in many cases we know by construction that the initial situation is bounded. In such cases the proof technique of Theorem 22 provides an effective way to check if the entire theory is bounded.

10. Related work

Besides the situation calculus [65,75], many other formalisms for reasoning about actions have been developed in AI, including the event calculus [55,80,81], the features and fluents framework [77], action languages such as $\mathcal{A}$ [44] and $\mathcal{C}+$ [47], the fluent calculus [87], and many others. In most of these, the focus is on addressing problems in the representation of action and change, such as the frame problem. Some attention has also been paid to specifying and verifying general temporal properties, especially in the context of planning. The Planning Domain Definition Language (PDDL) [66] has been

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

developed for specifying planning domains and problems, and a recent version supports the expression of temporal constraints on the plan trajectory [46]. Approaches such as those in TLPlan [3], in TALplanner [56], or in planning via model checking [68] support planning with such temporal constraints. Within the situation calculus, temporal constraints for planning have been studied in, e.g., [11,7]. All these planning-related approaches are essentially propositional and give rise to transition systems that are finite-state. One interesting attempt to interpret first-order linear temporal logic simultaneously as a declarative specification language and procedural execution language is that of MetateM [8], though verification is not addressed.

Most work on verification has been done in computer science, generally focusing on finite-state systems and programs. Many logics have been developed to specify temporal properties of such systems and programs, including linear-time logics, such as Linear Temporal Logic (LTL) [69] and Property-Specification Language (PSL) [39], and branching time logics such as Computation Tree Logic (CTL) [20] and CTL* [41], the μ-calculus [40,15], which subsumes the previous two, as well as Propositional Dynamic Logic (PDL) [42], which incorporates programs in the language. Model checking (and satisfiability) in these propositional modal logics is decidable [6], but such logics can only represent finite domains and finite state systems. Practical verification systems, e.g., [52,19], have been developed for many such logics, based on model checking techniques [6].

In AI, verification by model checking has become increasingly popular in the autonomous agents and multi-agent systems area. There, many logics have been proposed that additionally deal with the informational and motivational attitudes of agents [72,70,90,93,23,82]. Some recent work has been specifically concerned with formalizing multi-agent knowledge/belief and their dynamics [89,51]. Moreover, various Belief–Desire–Intention (BDI) agent programming languages have been developed that operationalize these mental attitudes [71,13,24,25]. Verification is important in this area as agent autonomy makes it crucial to be able to guarantee that the system behaves as required [43]. Furthermore, one generally wants to ensure that the agents' mental states as well as their behaviors evolve in a way that satisfies certain properties. Agent logics can be used to specify such properties. Much of the verification work in this area focuses on the model checking of BDI programs. For instance, [12] shows how to use the SPIN model checker [52] to verify properties of finite-state AgentSpeak programs. [36,43] compile BDI programs and agent properties to verify into Java and use JPF [92] to model check them. [63] develops MCMAS, a symbolic model checker specifically for multi-agent systems. [2] develops a theorem proving-based verification framework for BDI programs that uses a PDL-like logic.

In the situation calculus, there is also some previous work on verification. Perhaps the first such work is [34], where verification of possibly non-terminating Golog [58] programs is addressed, though no effective techniques are given. Focusing on the propositional situation calculus (where fluents have only the situation as argument), [86] presents decidable verification techniques. In [48], these techniques are generalized to a one-object-argument fluents fragment of the situation calculus, and in [49] to theories expressed in two-object-argument fragment. Techniques for verification resorting to second-order theorem proving with no decidability guarantees are presented in [82,83], where the CASLve verification environment for multi-agent ConGolog [26] programs is described. In [21], *characteristic graphs* for programs are introduced to define a form of regression over programs to be used as a pre-image computation step in (sound) procedures for verifying Golog and ConGolog programs inspired by model checking. Verification of programs over a two-variable fragment of the situation calculus is shown to be decidable in [22]. [54] establishes conditions for verifying loop invariants and persistence properties. Finally, [32,78] propose techniques (with model-checking ingredients) to reason about infinite executions of Golog and ConGolog programs based on second-order logic exploiting fixpoint approximates.

More recently, work closely related to ours [27,28,31,30] has shown that one obtains robust decidability results for temporal verification of situation calculus action theories under the assumption that in every situation the number of object tuples forming the extension of each fluent is bounded by a constant. In particular, [27] introduced bounded situation calculus basic action theories; this work, however, assumes standard names for the object domain and, more significantly, disallows quantification across situations in the verification language. In the present paper, which is a direct extension of [27], both of these limitations are removed. In [28] an extended language with an explicit knowledge operator was considered, while in [31] online executions (i.e., executions where the agent only performs actions that it knows are executable) and progression are studied; like [27], these papers also assume standard names and rule out quantification across situations from the verification language. [30] addresses verification over online executions with sensing in bounded situation calculus theories, adopting as verification language a first-order variant of Linear Temporal Logic (FO-LTL), again without quantification across situations.

One may think of bounded action theories as related to certain classes of action theories that ensure that the progression of the theory remains first-order representable [61,91], for instance, local-effect action theories [62], where only fluent instances involving the arguments of an action may be affected by it. However, it should be clear that bounded action theories need not be local-effect, as actions in a bounded action theory may affect fluent instances involving objects that are not arguments of the action, as long as the extension of fluents remains bounded. Nor are local-effect action theories necessarily bounded. Nonetheless, [31,30] show that the progression of a bounded action theory is always first-order representable. It would be interesting to study further the relationship between such classes of theories.

The work in this paper is also closely related to [10]. There, an ad-hoc formalism for representing actions and change is developed with the purpose of capturing data-aware artifact-centric processes. This formalism describes action preconditions and postconditions in first-order logic, and induces *genericity* [1] – there called *uniformity* – on the generated transition system. Intuitively genericity requires that if two states are isomorphic they induce the "same" transitions (modulo isomor-

phism). This means, in particular, that the system is essentially *Markovian* [75]. As verification language, [10] considers FO-CTL, a first-order variant of CTL that allows for quantifying across states *without requiring object persistence*, as, instead, we do here. Their results imply that one can construct a finite-state transition system over which the FO-CTL formula of interest can be verified. However, differently from our case, such a transition system depends also on the number of variables in the formula. While bounded situation calculus action theories enjoy genericity, it is easy to see that, without assuming object persistence, we immediately lose the possibility of abstracting to a finite transition system independently from the formula to verify. This is true even if we drop completely fixpoints. Indeed, assume that we have an action that replaces an object in the active domain by one in its parameters. Then, without persistence, for any bound n over the number of objects in a candidate finite abstraction, we can write a (fixpoint-free) formula saying that there exists a finite run with more than n distinct objects:

$$\begin{aligned}&\exists x_1.\textsc{live}(x_1) \wedge \langle - \rangle (\exists x_2.\textsc{live}(x_2) \wedge x_2 \neq x_1 \wedge \\ &\quad \langle - \rangle (\exists x_3 \textsc{live}(x_3) \wedge x_3 \neq x_1 \wedge x_3 \neq x_2 \wedge \\ &\qquad \cdots \\ &\qquad\quad \langle - \rangle (\exists x_{n+1} \textsc{live}(x_{n+1}) \wedge x_{n+1} \neq x_1 \wedge \cdots \wedge x_{n+1} \neq x_n)))\end{aligned}$$

Obviously, this formula is false in the finite abstraction, while true in the original transition system, where objects are not "reused". Notice that the formula belongs also to FO-CTL and this limitation applies to [10] as well. This observation shows that the persistence condition is crucial to obtain an abstraction that is independent from the formula.

It is interesting to observe that while dropping persistence is certainly a valuable syntactic simplification, the deep reason behind it is that generic transition systems, including those generated by situation calculus basic action theories, are essentially unable to talk about objects that are not in the current active domain. If some object that is in the active domain disappears from it and reappears again, after some steps, the basic action theory will treat it essentially as a fresh object (i.e., an object never seen before). Hence, any special treatment of such objects must come from the formula we are querying the transition system with: for example, we may isolate runs with special properties and only on those do verification. The fact that FO-CTL can drop persistence while maintaining decidability of verification over generic transition systems tells us that FO-CTL is not powerful enough to isolate interesting runs to be used as a further assumption for verification. Recently, this intuition has been formally proven for the entire μ-calculus, by showing that the two notions of bisimulation induced by enforcing or relaxing persistence collapse for generic transition systems [17]. On the basis of this result, the paper shows the decidability of verification of μ-calculus properties without enforcing persistence against bounded situation calculus action theories. Interestingly this contrasts with the undecidability of FO-LTL when persistence is not enforced [17,5].

The results in this paper are relevant not only for AI, but also for other areas of computer science (CS). There is some work in CS 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., [16]. There has recently been some attention paid in the field of business processes and services to include data into the analysis of processes [53,45,38]. Interestingly, while we have verification tools that are quite good for dealing with data and processes separately, when we consider them together, we obtain infinite-state transition systems, which resist classical model checking approaches to verification. Only lately has there been some work on developing verification techniques that can deal with infinite-state processes [37,4,9,5,10]. In particular, the form of controlled quantification across situations in our $\mu\mathcal{L}_p$ language, which requires object persistence in the active domain, is inspired by the one in [5], which in turn extends the verification logic presented in [27]. There, the infinite-state data-aware transition systems (with complete information) to verify are defined using an ad-hoc formalism based on database operations, and the decidability results are based on two conditions over the transition systems, namely run-boundedness and state-boundedness. The latter is analogous to our situation-boundedness. In this paper, we make the idea of boundedness flourish in the general setting offered by the situation calculus, detailing conditions needed for decidability, allowing for incomplete information, and exploiting the richness of the situation calculus for giving sufficient conditions for boundedness that can easily be used in practice. Such results can find immediate application in the analysis of data-aware business processes and services.

11. Conclusion

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

A future research direction of particular interest is a more systematic investigation of specification patterns for obtaining boundedness. This includes patterns that provide bounded persistence and patterns that model bounded/fading memory.

These questions should be examined in light of different approaches that have been proposed for modeling knowledge, sensing, and revision in the situation calculus and related temporal logics [79,35,84,89]. This work has already started. In particular, as mentioned earlier, the approach of this paper has been extended in [31,30] to allow verification of temporal properties over *online executions* of an agent, where the agent may acquire new information through sensing as it executes and only performs actions that are feasible according to its beliefs. In that work, the agent's belief state is modeled meta-theoretically, as an action theory that is progressed as actions are performed and sensing results are obtained. In [28], temporal epistemic verification is tackled within a language-theoretic viewpoint, where the situation calculus is extended with a knowledge modality [79]. The form of boundedness studied in that case requires that the number of object tuples that the agent thinks may belong to any given fluent be bounded. In [31,30], instead, it is only required that number of distinct tuples entailed to belong to a fluent is bounded, while the number of tuples that are in the extension of a fluent in some model of the theory need not be bounded. More work is needed to fully reconcile these meta-theoretic and language-theoretic approaches.

Finally, an important topic to address is verification of agent programs [34] expressed in a high-level language, like Golog [58] or ConGolog [26], which are based on the situation calculus. Some cases where verification of ConGolog programs is decidable are identified in [22]. More recently, the framework presented here has been extended to support verification of ConGolog programs, without recursive procedures, over bounded situation calculus action theories [29]. Note that this extension is not immediate, as a temporal property may hold over all executions of a program without holding over all branches of the situation tree. However it can be shown that the whole program configuration, formed by the situation and the remaining part of the program to execute, is bounded (if we disallow recursion), and hence the approach presented in this paper can be applied to show decidability of verification [29].

Acknowledgements

The authors acknowledge the support of: Sapienza Università di Roma, under the research project *Immersive Cognitive Environments*; Ripartizione Diritto allo Studio, Università e Ricerca Scientifica of Provincia Autonoma di Bolzano–Alto Adige, under project *Verification and Synthesis from Components of Processes That Manipulate Data*; and the National Science and Engineering Research Council of Canada under grant RGPIN-2015-03756 *Specification, Verification, and Synthesis of Autonomous Adaptive Agents*.

References

[1] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison Wesley, 1995.
[2] N. Alechina, M. Dastani, F. Khan, B. Logan, J.-J. Meyer, Using theorem proving to verify properties of agent programs, in: Specification and Verification of Multi-Agent Systems, Springer, 2010, pp. 1–33.
[3] F. Bacchus, F. Kabanza, Planning for temporally extended goals, Ann. Math. Artif. Intell. 22 (1–2) (1998) 5–27.
[4] B. Bagheri Hariri, D. Calvanese, G. De Giacomo, R. De Masellis, P. Felli, Foundations of relational artifacts verification, in: Proc. of BPM, 2011, pp. 379–395.
[5] B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, M. Montali, Verification of relational data-centric dynamic systems with external services, in: Proc. of PODS, 2013, pp. 163–174.
[6] C. Baier, J.-P. Katoen, K. Guldstrand Larsen, Principles of Model Checking, MIT Press, 2008.
[7] J.A. Baier, S.A. McIlraith, Planning with temporally extended goals using heuristic search, in: Proc. of ICAPS, 2006, pp. 342–345.
[8] H. Barringer, M. Fisher, D.M. Gabbay, G. Gough, R. Owens, MetateM: an introduction, Form. Asp. Comput. 7 (5) (1995) 533–549.
[9] F. Belardinelli, A. Lomuscio, F. Patrizi, Verification of deployed artifact systems via data abstraction, in: Proc. of ICSOC, 2011, pp. 142–156.
[10] F. Belardinelli, A. Lomuscio, F. Patrizi, Verification of agent-based artifact systems, J. Artif. Intell. Res. 51 (2014) 333–376.
[11] M. Bienvenu, C. Fritz, S.A. McIlraith, Planning with qualitative temporal preferences, in: Proc. of KR, 2006, pp. 134–144.
[12] R.H. Bordini, M. Fisher, C. Pardavila, M. Wooldridge, Model checking AgentSpeak, in: Proc. of AAMAS, 2003, pp. 409–416.
[13] R.H. Bordini, J.F. Hubner, M. Wooldridge, Programming Multi-Agent Systems in AgentSpeak Using Jason, Wiley, 2007.
[14] C. Boutilier, R. Reiter, M. Soutchanski, S. Thrun, Decision-theoretic, high-level agent programming in the situation calculus, in: Proc. of AAAI/IAAI, 2000, pp. 355–362.
[15] J. Bradfield, C. Stirling, Modal mu-calculi, in: Handbook of Modal Logic, vol. 3, Elsevier, 2007, pp. 721–756.
[16] O. Burkart, D. Caucal, F. Moller, B. Steffen, Verification of infinite structures, in: Handbook of Process Algebra, Elsevier, 2001, pp. 545–623.
[17] D. Calvanese, G. De Giacomo, M. Montali, F. Patrizi, On first-order μ-calculus over situation calculus action theories, in: Proc. of KR, 2016.
[18] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer, Alternation, J. ACM 28 (1) (1981) 114–133.
[19] A. Cimatti, E.M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, NuSMV 2: an opensource tool for symbolic model checking, in: Proc. of CAV, 2002, pp. 359–364.
[20] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching-time temporal logic, in: Proc. of Logics of Programs, Workshop, 1981, pp. 52–71.
[21] J. Claßen, G. Lakemeyer, A logic for non-terminating Golog programs, in: Proc. of KR, 2008, pp. 589–599.
[22] J. Claßen, M. Liebenberg, G. Lakemeyer, B. Zarrieß, Exploring the boundaries of decidable verification of non-terminating Golog programs, in: Proc. of AAAI, 2014, pp. 1012–1019.
[23] P.R. Cohen, H.J. Levesque, Intention is choice with commitment, Artif. Intell. 42 (2–3) (1990) 213–261.
[24] M. Dastani, 2apl: a practical agent programming language, Auton. Agents Multi-Agent Syst. 16 (3) (2008) 214–248.
[25] F.S. de Boer, K.V. Hindriks, W. van der Hoek, J.C. Meyer, A verification framework for agent programming with declarative goals, J. Appl. Log. 5 (2) (2007) 277–302.
[26] G. De Giacomo, Y. Lespérance, H.J. Levesque, ConGolog, a concurrent programming language based on the situation calculus, Artif. Intell. 121 (1–2) (2000) 109–169.
[27] G. De Giacomo, Y. Lespérance, F. Patrizi, Bounded situation calculus action theories and decidable verification, in: Proc. of KR, 2012, pp. 467–477.
[28] G. De Giacomo, Y. Lespérance, F. Patrizi, Bounded epistemic situation calculus theories, in: Proc. of IJCAI 2013, 2013, pp. 846–853.

[29] G. De Giacomo, Y. Lesperance, F. Patrizi, S. Sardina, Verifying ConGolog programs on bounded situation calculus theories, in: Proc. of AAAI, 2016.
[30] G. De Giacomo, Y. Lespérance, F. Patrizi, S. Vassos, LTL verification of online executions with sensing in bounded situation calculus, in: Proc. of ECAI, 2014, pp. 369–374.
[31] G. De Giacomo, Y. Lespérance, F. Patrizi, S. Vassos, Progression and verification of situation calculus agents with bounded beliefs, in: Proc. of AAMAS, 2014, pp. 141–148.
[32] G. De Giacomo, Y. Lespérance, A.R. Pearce, Situation calculus based programs for representing and reasoning about game structures, in: Proc. of KR, 2010, pp. 445–455.
[33] G. De Giacomo, H.J. Levesque, Projection using regression and sensors, in: Proc. of IJCAI, 1999, pp. 160–165.
[34] G. De Giacomo, E. Ternovskaia, R. Reiter, Non-terminating processes in the situation calculus, in: Proc. of the AAAI'97 Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control, 1997, pp. 18–28.
[35] R. Demolombe, M. del Pilar Pozos Parra, A simple and tractable extension of situation calculus to epistemic logic, in: Proc. of ISMIS, 2000, pp. 515–524.
[36] L.A. Dennis, M. Fisher, M.P. Webster, R.H. Bordini, Model checking agent programming languages, Autom. Softw. Eng. 19 (1) (2012) 5–63.
[37] A. Deutsch, R. Hull, F. Patrizi, V. Vianu, Automatic verification of data-centric business processes, in: Proc. of ICDT, 2009, pp. 252–267.
[38] M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede, Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley, 2005.
[39] C. Eisner, D. Fisman, A Practical Introduction to PSL. Integrated Circuits and Systems, Springer, 2006.
[40] E.A. Emerson, Model checking and the mu-calculus, in: Descriptive Complexity and Finite Models, AMS, DIMACS, 1996, pp. 185–214.
[41] E.A. Emerson, J.Y. Halpern, "Sometimes" and "not never" revisited: on branching versus linear time (preliminary report), in: Proc. of POPL, 1983, pp. 127–140.
[42] M.J. Fischer, R.E. Ladner, Propositional dynamic logic of regular programs, J. Comput. Syst. Sci. 18 (2) (1979) 194–211.
[43] M. Fisher, L.A. Dennis, M.P. Webster, Verifying autonomous systems, Commun. ACM 56 (9) (2013) 84–93.
[44] M. Gelfond, V. Lifschitz, Representing action and change by logic programs, J. Log. Program. 17 (2/3&4) (1993) 301–321.
[45] C.E. Gerede, J. Su, Specification and verification of artifact behaviors in business process models, in: Proc. of ICSOC, 2007, pp. 181–192.
[46] A. Gerevini, D. Long, Preferences and soft constraints in PDDL3, in: Proc. of ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning, 2006, pp. 46–54.
[47] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, H. Turner, Nonmonotonic causal theories, Artif. Intell. 153 (1–2) (2004) 49–104.
[48] Y. Gu, I. Kiringa, Model checking meets theorem proving: a situation calculus based approach, in: Proc. of 11th International Workshop on Nonmonotonic Reasoning, Action, and Change, 2006.
[49] Y. Gu, M. Soutchanski, Decidable reasoning in a modified situation calculus, in: Proc. of IJCAI, 2007, pp. 1891–1897.
[50] M. Hennessy, R. Milner, On observing nondeterminism and concurrency, in: Proc. of ICALP, 1980, pp. 295–309.
[51] A. Herzig, Belief change operations: a short history of nearly everything, told in dynamic logic of propositional assignments, in: Proc. of KR, 2014, pp. 141–150.
[52] G.J. Holzmann, The model checker SPIN, IEEE Trans. Softw. Eng. 23 (5) (1997) 279–295.
[53] R. Hull, Artifact-centric business process models: brief survey of research results and challenges, in: Proc. of OTM 2008 Confederated International Conferences, 2008, pp. 1152–1163.
[54] R.F. Kelly, A.R. Pearce, Property persistence in the situation calculus, Artif. Intell. 174 (12–13) (2010) 865–888.
[55] R.A. Kowalski, M.J. Sergot, A logic-based calculus of events, New Gener. Comput. 4 (1) (1986) 67–95.
[56] J. Kvarnström, P. Doherty, TALplanner: a temporal logic based forward chaining planner, Ann. Math. Artif. Intell. 30 (1–4) (2000) 119–169.
[57] H.J. Levesque, G. Lakemeyer, The Logic of Knowledge Bases, MIT Press, 2001.
[58] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin, R.B. Scherl, GOLOG: a logic programming language for dynamic domains, J. Log. Program. 31 (1997) 59–84.
[59] L. Libkin, Elements of Finite Model Theory, Springer, 2004.
[60] L. Libkin, Embedded finite models and constraint databases, in: Finite Model Theory and Its Applications, Springer, 2007, pp. 257–338.
[61] F. Lin, R. Reiter, How to progress a database, Artif. Intell. 92 (1–2) (1997) 131–167.
[62] Y. Liu, H.J. Levesque, Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions, in: Proc. of IJCAI, 2005, pp. 522–527.
[63] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: a model checker for the verification of multi-agent systems, in: Proc. of CAV, 2009, pp. 682–688.
[64] A. Marrella, M. Mecella, S. Sardiña, SmartPM: an adaptive process management system through situation calculus, IndiGolog, and classical planning, in: Proc. of KR, 2014, pp. 1–10.
[65] J. McCarthy, P.J. Hayes, Some philosophical problems from the standpoint of artificial intelligence, Mach. Intell. 4 (1969) 463–502.
[66] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL—the planning domain definition language, Tech. Rep. CVC TR98003/DCS TR1165, Yale Center for Computational Vision and Control, 1998.
[67] F. Pirri, R. Reiter, Some contributions to the metatheory of the situation calculus, J. ACM 46 (3) (1999) 261–325.
[68] M. Pistore, P. Traverso, Planning as model checking for extended goals in non-deterministic domains, in: Proc. of IJCAI, 2001, pp. 479–484.
[69] A. Pnueli, The temporal logic of programs, in: Proc. of FOCS, 1997, pp. 46–57.
[70] A. Rao, M. Georgeff, An abstract architecture for rational agents, in: Proc. of KR, 1992, pp. 439–449.
[71] A.S. Rao, AgentSpeak(l): BDI agents speak out in a logical computable language, in: Proc. of Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, 1996, pp. 42–55.
[72] A.S. Rao, M.P. Georgeff, Modeling rational agents within a BDI-architecture, in: Proc. of KR, 1991, pp. 473–484.
[73] R. Reiter, The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression, in: Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, Academic Press, 1991, pp. 359–380.
[74] R. Reiter, Natural actions, concurrency and continuous time in the situation calculus, in: Proc. of KR, 1996, pp. 2–13.
[75] R. Reiter, Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press, 2001.
[76] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Prentice Hall, 2010.
[77] E. Sandewall, Features and Fluents, Oxford University Press, New York, 1994.
[78] S. Sardiña, G. De Giacomo, Composition of ConGolog programs, in: Proc. of IJCAI, 2009, pp. 904–910.
[79] R.B. Scherl, H.J. Levesque, Knowledge, action, and the frame problem, Artif. Intell. 144 (1–2) (2003) 1–39.
[80] M. Shanahan, Solving the Frame Problem – A Mathematical Investigation of the Common Sense Law of Inertia, MIT Press, 1997.
[81] M. Shanahan, The event calculus explained, in: Artificial Intelligence Today, Springer, 1999, pp. 409–430.
[82] S. Shapiro, Y. Lespérance, H. Levesque, The cognitive agents specification language and verification environment, in: Specification and Verification of Multi-Agent Systems, Springer, 2010, pp. 289–315.
[83] S. Shapiro, Y. Lespérance, H.J. Levesque, The cognitive agents specification language and verification environment for multiagent systems, in: Proc. of AAMAS, 2002, pp. 19–26.
[84] S. Shapiro, M. Pagnucco, Y. Lespérance, H.J. Levesque, Iterated belief change in the situation calculus, Artif. Intell. 175 (1) (2011) 165–192.
[85] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pac. J. Math. 5 (2) (1955) 285–309.

[86] E. Ternovskaia, Automata theory for reasoning about actions, in: Proc. of IJCAI, 1999, pp. 153–159.
[87] M. Thielscher, From situation calculus to fluent calculus: state update axioms as a solution to the inferential frame problem, Artif. Intell. 111 (1–2) (1999) 277–299.
[88] J. van Benthem, Modal Logic and Classical Logic, Bibliopolis, 1983.
[89] H. van Ditmarsch, W. van der Hoek, B. Kooi, Dynamic Epistemic Logic, Springer, 2008.
[90] B. van Linder, W. van der Hoek, J.C. Meyer, Formalising abilities and opportunities of agents, Fundam. Inform. 34 (1–2) (1998) 53–101.
[91] S. Vassos, F. Patrizi, A classification of first-order progressable action theories in situation calculus, in: Proc. of IJCAI, 2013, pp. 1132–1138.
[92] W. Visser, K. Havelund, G.P. Brat, S. Park, F. Lerda, Model checking programs, Autom. Softw. Eng. 10 (2) (2003) 203–232.
[93] M. Wooldridge, Reasoning about Rational Agents, MIT Press, 2000.